SUPPLEMENTAL MATERIAL FOR "EVOLVING DEEP NEURAL NETWORK'S WEIGHTS AT IMAGENET SCALE"

Anonymous authors
Paper under double-blind review

S1 CORRESPONDENCE BETWEEN DARWINIAN EVOLUTION AND NEURAL NETWORK OPTIMIZATION USING DIFFERENTIAL EVOLUTION

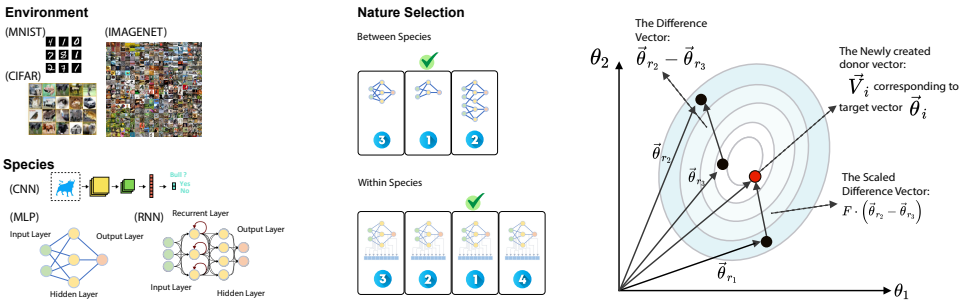

Figure S1: **Conceptual illustration of our proposed Darwinian evolution on neural networks using DE.** (Left) In analogy to Darwinian evolution, the dataset provides the environment where different types of neural networks strike to survive. (Middle) The evolution (natural selection and inheritance) applies to different network architectures and trainable weights in the same architect. Pretrained neural networks are the primordial ancestors for the DE to evolve and to select the 'elite' solution. (Right) An illustration of DE mutation.

The theory of evolution (Smith, 1993), supported by evidence from genetics, paleontology, and geology, describes how living beings originate from primordial soup, make up the first species (primordial ancestor, also known as last universal common ancestor (LUCA)), survive under environmental pressure, and evolve in nature over long periods. Since Charles Darwin's book "On the Origin of Species" was published in 1859 (Darwin, 2004), the theory has been expanding to describe the necessary conditions, ingredients, and mechanisms before the start of and the transition to evolution. In Fig. S1, we illustrate Darwinian Evolution's analogy to the neural network optimization problem using differential evolution (DE). The neural network architecture and the dataset play the role of the species and environment, respectively. Different architectures specialize in different functions, for example, convolutional neural networks (e.g., ResNet (He et al., 2015), MobileNet (Howard et al., 2017)) and recurrent neural networks (e.g., LSTM (Hochreiter & Schmidhuber, 1997), GRU (Chung et al., 2014)) capture translation invariances and temporal dependencies underlie the data by implementing the inner workings of the visual cortex and memory. The trainable parameters can be interpreted as the genetic traits within the architect framework that affects survival and breeding. On the other side, the complexity of the dataset can be interpreted as the complexity of the environment, scaling from simple MNIST(LeCun et al., 1998) to big data ImageNet(Deng et al., 2009). Survival fitness can be defined as the loss function of the learning tasks. This work focuses on ANN training by considering the evolution of a population of trainable parameters in the pre-defined architecture (i.e., single organism) rather than evolving a population of different architectures (Real et al., 2017; Lu et al., 2020).

A detailed description of DE is provided in the Method section of the main text. In Fig. S1, an illustration of the DE's mutation strategy S_1 in 2D search space is shown, with the contour lines indicating fitness functions in the search space. The figure shows how DE simulates mutation using any three candidate solutions in the population. Various improvements have been on the ordinary DE by adding adaptive learning (i.e., SADE, SHADE) of mutation & recombination, trigonometric and sinusoidal mutations (Sin-F) for scaling factor F.

S1.1 CONCEPT OF PRIMORDIAL SOUP AND ANCESTOR

Primordial soup theory (Taylor, 2005), the Miller-Urey experiment (Miller, 1953), and others (Kasting, 1993) studied and simulated the conditions of early Earth for the first life, which arose from non-living matters, give rise to other species through evolution. This work examined neuro-evolution that begins with the primordial ancestor, ADAM-trained neural networks as the first species. Neuro-evolution that starts from the primordial ancestor is found to be empirically superior to the primordial soup, randomly initialized neural networks, as shown in Fig. S2. Hence, the results align with the Primordial soup theory, showing the importance of forming a species before neuro-evolution takes place.

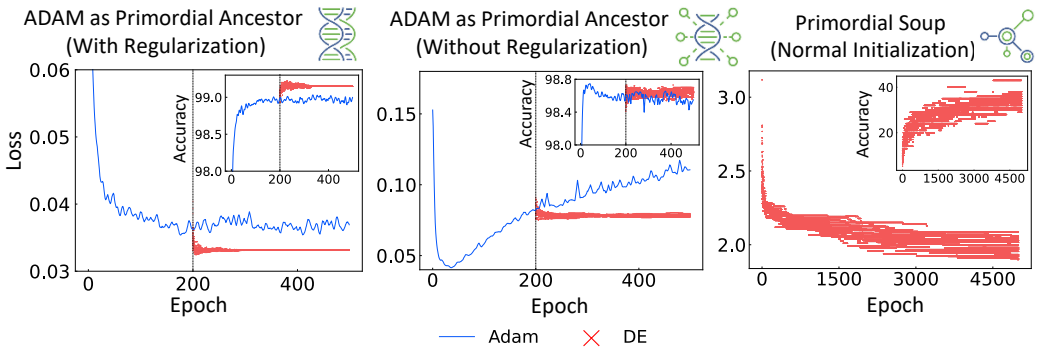

Figure S2: **The starting point of neuro-evolution: primordial ancestor v.s. primordial soup.** The left and middle figures illustrate how the losses and accuracies change over the epoch for DNNs trained by Adam (blue curves) and DE (red markers) using Adam as the primordial ancestor, with and without regularization. It is observed that Adam requires regularization to handle overfitting, while DE does not require. The right figure evolves a neural network from the primordial soup using random initialization, which has difficulty in convergence.

S2 SMALL DATASET AND LENETS

We carried out experiments in small datasets, MNIST, Fashion MNIST, CIFAR-10, and CIFAR-100, and we trained models (i.e., LeNet1, LeNet5, MLP, RNN) from scratch using BP with early stopping and used the DNNs from the ending epochs (the last ps epochs) as the initial population for ordinary DE. The hyperparameters of DE being examined are summarized in Table S2. The ordinary DE with strategy S_1 and crossover are used to evolve DNNs. The moderate size of datasets and DNNs allows us to examine different configurations of evolving DNNs in a controllable way.

In Fig.S3, LeNet1 and LeNet5 architectures are evolved in the environment (dataset) with different complexities, which provide different amounts of training data generated using data augmentation. The evolution of DE is observed to have positive impacts on the ADAM-trained neural network. Moreover, datasets and models with higher complexity are observed to achieve better performance. Both the LeNet1 and LeNet5 evolved using the proposed Darwinian evolution framework are found to be better than the LeNet5 trained using BP by LeCun et al. in 1998 (Lecun et al., 1998). The result is summarized in Table S1, with the error curve shown in Figure S4. The middle panel of Fig. S3 shows the degrees of improvement on top of the BP-based approach under different degrees of $L2$ regularization. It is observed that the best results are achieved with a regularization of 0.0001 compared to no regularization at all, and all regularization parameters lead to increased model accuracy. The use of regularization reduces the impact of over-fitting in BP-based optimizers. During the Darwinian evolution, it is observed that the trait of preventing over-fitting is inherited without an explicit $L2$ regularization in the fitness function. To assess the out-of-distribution robustness of the neural network trained using the proposed framework, two datasets with common corruptions are used, namely MNIST-C and CIFAR-10-C. The types of corruption are summarized at the right panel of Fig. S3.

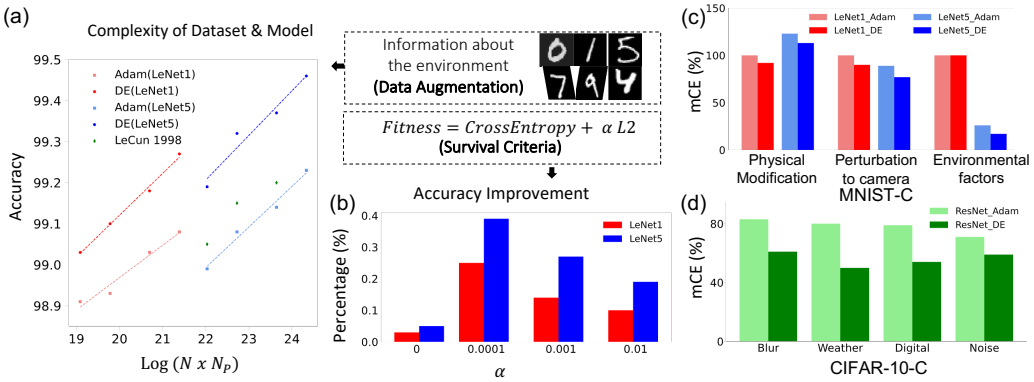

Figure S3: **The impact of environment (dataset and loss function) on the DNNs.** The left figure shows the relationships between the accuracy and the complexity of the dataset (MNIST with data augmentation) and model (LeNets). The middle bottom figure illustrates the influence of regularization. The two figures at the right show the performance of ADAM and DE on the corrupted MNIST-C and CIFAR-10-C.

Table S1: **Test accuracy of LeNets on MNIST and CIFAR.** (M) MNIST, (F.M) Fashion MNIST, (C) CIFAR

	LeCun's (Lecun et al., 1998) Acc	Adam Acc	DE Acc	Params
LeNet1(M)	98.30	98.78	99.03	3,246
LeNet5(M)	99.05	98.95	99.20	62,006
LeNet5(F.M)	na	89.27	89.43	62,006
LeNet5(C-10)	na	60.16	61.36	62,006
LeNet5(C-100)	na	24.47	26.73	62,006

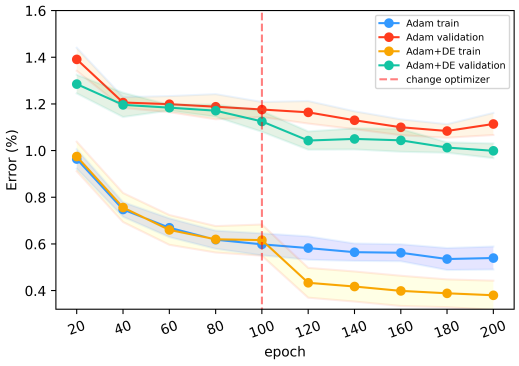

Figure S4: **Average classification error with standard errors across epochs during the training process.** LeNet-1 model consistently converges across 10 runs on MNIST.

S2.1 Darwinian evolution does not overfit

In nature, species evolve for better fit but not overfit (e.g., giraffes can reach higher leaves with the evolved long necks, but not over-long necks (Wang et al., 2022)). In this work, we studied whether

evolving neural networks using DE will lead to overfitting that gradient descent methods suffer. Generally, DE is found to have no overfitting issue, as shown in Fig. S2. Comparing the primordial ancestor trained with and without regularization, it is also observed that the quality of the primordial ancestor has a significant impact on DNNs, demonstrating the effect of descent in the same lineage. In the same lineage, offspring receive the genetic traits from parents through inheritance with slight variation and modification. In Fig. S2, the lineage trained using BP with regularization performs better, as the use of regularization reduces the impact of over-fitting in BP-based optimizers. During the Darwinian evolution, it is observed that the trait of preventing over-fitting is inherited without an explicit $L2$ regularization in the fitness function.

S2.2 Performance on Different Datasets and Models

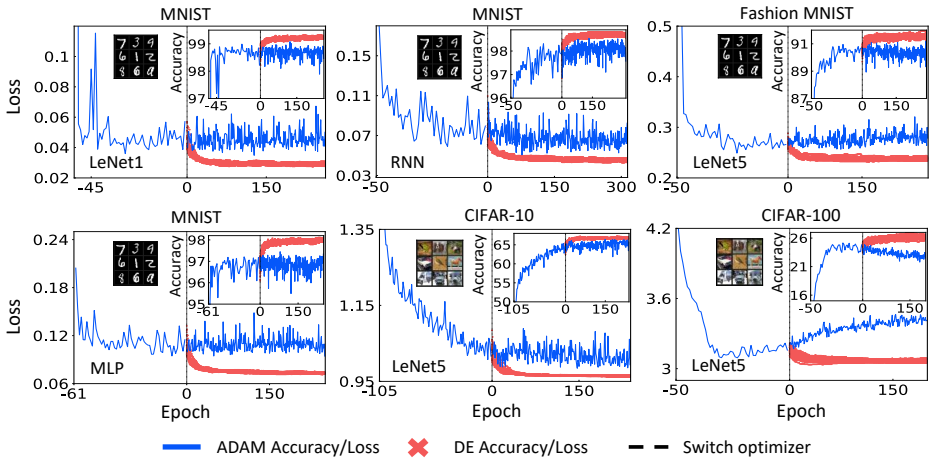

Figure S5: **Generalization of DE on different datasets and deep learning models.** Datasets include MNIST, Fashion MNIST, CIFAR-10, and CIFAR-100, while models include: LeNet1, LeNet5, MLP, and RNN. The blue lines represent Adam optimizer, and the red points represent the DE optimizer.

S2.3 Impact of Scaling Factor and Crossover Rate

Our comprehensive analysis has revealed the crucial significance of selecting appropriate parameters, namely F and Cr, shown in Figs. S6 and S7. Hence, an adaptive approach is employed to enhance the performance of the DE optimizer to evolve DNN on ImageNet.

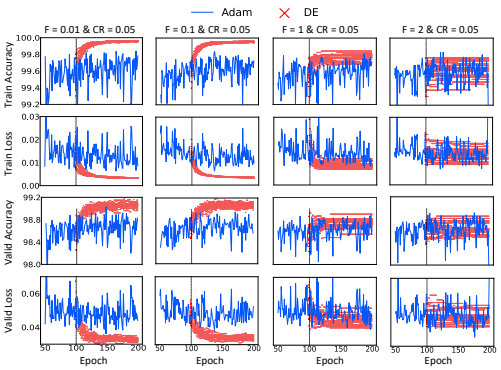

Figure S6: The impact of the mutation factor F. Comparison between DE (blue curves) and Adam (red marker) for RNN models, with fixed crossover rate and different F.

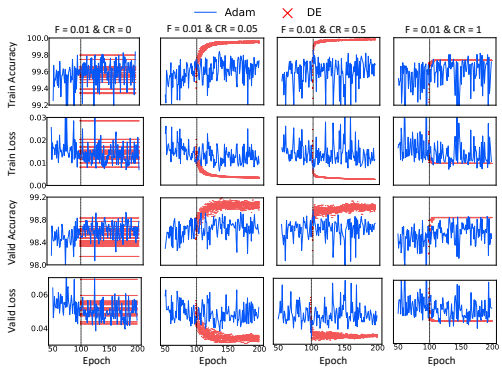

Figure S7: The impact of the crossover rate CR. Comparison between DE (blue curves) and Adam (red marker) for RNN models, with fixed mutation factor and different CR.

S2.4 HYPERPARAMETS USED

Table S2: Hyper-parameters for DE.

Name	Symbol	Value/ Range
Scaling factor	F	[0.01, 0.1, 1, 2]
Crossover rate	Cr	[0, 0.05, 0.5, 1]
Mutation strategy	S	S_1
# of generations	Gen	200-300
Batch size	bs	50000
Population size	ps	20
BP algorithm		ADAM
Update scope		whole NN
Regularization	α	[0, 10^{-4}, 10^{-3}, 10^{-2}]

Table S3: Hyper-parameters in SHADE-tri-ensin

Name	Symbol	Value/ Range
Scaling factor	F	0-0.2
Crossover rate	Cr	0-0.3
Mutation strategies	S	S_1 - S_4, S_{Trigo}
# of generations	Gen	200
Batch size	bs	2048
Population size	ps	10
BP algorithm		SGD pretraining
Update scope		exempt FC
Archive size	H	5

S2.5 ROBUSTNESS AGAINST NOISE

To verify the robustness and generalization of our approach, we trained the model using MNIST data and CIFAR-10 data and tested it on the MNIST-C and CIFAR10-C datasets, respectively. Table S4 shows the LeNet1 and LeNet5's performance on MNIST-C. Table S5 shows the LeNet1 and LeNet5's performance on CIFAR10-C. Table S6 shows the ResNet performance on CIFAR10-C.

Table S4: **Performance on MNIST-C.** The error and mean corruption error (mCE) of ADAM and DE for LeNet models trained on MNIST. Models that perform the best for each type of corruption is colored blue. Overall, LeNet5 trained by DE is found to be less susceptible to corruptions.

	Error				mCE			
	LeNet1_Adam	*LeNet1_DE*	*LeNet5_Adam*	*LeNet5_DE*	*LeNet1_Adam*	*LeNet1_DE*	*LeNet5_Adam*	*LeNet5_DE*
Shot Noise	2.7%	2.4%	3.0%	**2.1%**	100%	92%	114%	**79%**
Impulse Noise	13.6%	14.2%	11.3%	**9.6%**	100%	104%	83%	**70%**
Glass Blur	13.1%	12.2%	8.9%	**6.3%**	100%	93%	68%	**48%**
Motion Blur	20.1%	15.6%	10.8%	**8.9%**	100%	78%	54%	**44%**
Shear	3.5%	3.1%	3.1%	**2.5%**	100%	89%	88%	**74%**
Scale	11.2%	**7.9%**	12.4%	8.1%	100%	**71%**	111%	73%
Rotate	9.6%	8.7%	9.5%	**7.9%**	100%	90%	99%	**83%**
Brightness	84.9%	84.8%	8.2%	**4.2%**	100%	100%	10%	**5%**
Translate	44.9%	43.0%	43.3%	**42.0%**	100%	96%	96%	**94%**
Stripe	32.2%	31.8%	15.1%	**8.2%**	100%	99%	47%	**26%**
Fog	81.9%	82.4%	22.0%	**18.2%**	100%	101%	27%	**22%**
Spatter	5.2%	4.9%	3.1%	**2.6%**	100%	95%	59%	**49%**
Dotted Line	4.8%	4.6%	4.0%	**3.6%**	100%	96%	84%	**76%**
Zigzag	16.0%	15.4%	13.0%	**11.3%**	100%	96%	81%	**71%**
Canny Edges	22.9%	**20.1%**	40.2%	37.9%	100%	**88%**	176%	165%
Average	24.4%	23.4%	13.9%	**11.6%**	100%	92%	80%	**65%**

Table S5: **Performance on CIFAR10-C.** The error and mean corruption error (mCE) of ADAM and DE for LeNet models trained on CIFAR10. Models that perform the best for each type of corruption is colored blue. In all corruptions, LeNet5 trained by DE is found to be less susceptible to corruptions.

	Error				mCE			
	LeNet1_Adam	*LeNet1_DE*	*LeNet5_Adam*	*LeNet5_DE*	*LeNet1_Adam*	*LeNet1_DE*	*LeNet5_Adam*	*LeNet5_DE*
Gaussian	52.7%	52.3%	44.4%	**43.7%**	100%	99%	84%	**83%**
Shot	51.2%	51.2%	42.5%	**42.3%**	100%	100%	83%	**83%**
Impulse	56.2%	56.1%	48.3%	**47.4%**	100%	100%	86%	**84%**
Defocus	51.3%	49.9%	44.2%	**43.4%**	100%	97%	86%	**85%**
Glass	53.3%	52.1%	48.0%	**46.9%**	100%	98%	90%	**88%**
Motion	53.2%	51.9%	49.0%	**47.4%**	100%	97%	92%	**89%**
Zoom	53.5%	52.1%	48.5%	**46.9%**	100%	97%	91%	**88%**
Snow	52.1%	52.1%	46.4%	**46.0%**	100%	100%	89%	**88%**
Frost	54.9%	55.9%	52.4%	**50.2%**	100%	102%	96%	**92%**
Fog	58.8%	57.6%	56.7%	**54.7%**	100%	98%	96%	**93%**
Brightness	50.8%	49.8%	44.4%	**43.5%**	100%	98%	87%	**86%**
Contrast	67.2%	65.5%	65.3%	**64.0%**	100%	97%	97%	**95%**
Elastic	52.7%	51.7%	45.8%	**45.0%**	100%	98%	87%	**85%**
Pixel	49.7%	49.1%	41.7%	**41.1%**	100%	99%	84%	**83%**
JPEG	49.7%	48.7%	42.0%	**41.1%**	100%	98%	84%	**83%**
Average	53.8%	53.1%	48.0%	**46.9%**	100%	99%	89%	**87%**

Table S6: **Performance of deeper models on CIFAR10-C.** The error and mean corruption error (mCE) of ADAM and DE for deeper models trained on CIFAR10. Models that perform the best for each type of corruption is colored blue. Overall, deeper models trained by DE is found to be less susceptible to corruptions. ResNet and MobileNet are observed to be comparable, where each manages certain corruptions better.

	Error				mCE			
	ResNet_Adam	*ResNet_DE*	*MobileNet_Adam*	*MobileNet_DE*	*ResNet_Adam*	*ResNet_DE*	*MobileNet_Adam*	*MobileNet_DE*
Gaussian	37.5%	**30.9%**	68.6%	68.1%	71%	**59%**	130%	129%
Shot	36.0%	**28.4%**	56.2%	55.6%	70%	**55%**	110%	109%
Impulse	39.9%	**35.8%**	49.2%	49.1%	71%	**64%**	88%	87%
Defocus	40.3%	26.9%	21.7%	**21.2%**	79%	52%	42%	**41%**
Glass	42.4%	**37.8%**	49.6%	48.9%	80%	**71%**	93%	92%
Motion	49.3%	33.8%	31.1%	**30.3%**	93%	64%	58%	**57%**
Zoom	44.0%	30.7%	28.9%	**28.3%**	82%	57%	54%	**53%**
Snow	41.1%	29.6%	22.2%	**21.8%**	79%	57%	43%	**42%**
Frost	45.4%	27.6%	28.5%	**27.9%**	83%	**50%**	52%	51%
Fog	47.4%	30.3%	17.8%	**17.4%**	81%	51%	30%	**30%**
Brightness	39.5%	21.4%	9.3%	**9.2%**	78%	42%	18%	**18%**
Contrast	61.9%	43.7%	33.1%	**32.6%**	92%	65%	49%	**49%**
Elastic	41.6%	28.0%	20.5%	**20.0%**	79%	53%	39%	**38%**
Pixel	36.7%	**24.7%**	26.4%	26.9%	74%	**50%**	53%	54%
JPEG	36.0%	**23.0%**	24.0%	24.0%	72%	**46%**	48%	48%
Average	42.6%	**30.2%**	32.5%	32.1%	79%	**56%**	61%	60%

S3 Additional ResNets Experiment on ImageNet

S3.1 Impact of population size and batch size

Our method is similar to ensemble learning, where typically a larger number of models leads to better performance. Therefore, we increased the number of populations for analysis, as shown in Table S7. Additionally, in evolutionary algorithms, we require a certain number of samples to calculate fitness values for population selection. The mini-batch size of samples also has an impact on the evolution results, as demonstrated in Table S8.

Table S7: **Impact of population size.** The performance of different population size for evolution.

Population size	6	10	20	40
Top-1 (%)	76.542	76.568	76.611	**76.648**

Table S8: **Impact of batch size.** The mini-batch samples for computing fitness value with DE algorithms.

DE batch size	256	512	1024	64000	320000	980000
Top-1 (%)	76.542	**76.568**	76.553	76.551	76.561	76.563

S3.2 Population Initialization

In this paper, the population initialization is obtained by fine-tuning a pre-trained model. To validate the importance of this approach, we replaced the fine-tuning method with randomly generated Gaussian white noise for population initialization, i.e. Ref. (Whitaker & Whitley, 2023). Specifically, we computed the standard deviation of the initially fine-tuned population and used this standard deviation to generate Gaussian noise, which was then added to the pre-trained model. This resulted in a randomly initialized population. From the Table S9, it can be observed that the random noise initialization did not provide any benefits to our method.

S4 Time complexity

Remarkably, if a m-layers network is employed, $\sum_{i=1}^{m} l_i$ parameters are to be optimized, where l_i represents number of i-th layer. Time complexity are thus $\mathcal{O}(n_g \cdot \prod_{i=1}^{m-1} l_i l_{i+1})$ and $\mathcal{O}(n_e \cdot$

Table S9: **Comparison with randomly initialized population.**

Method	Pytorch Benchmark	Random noise	Fine-tuned parents
Top-1 (%)	76.13	76.15	**76.57**

$\sum_{i=1}^{m} l_i$) where n_g and n_e are training samples for gradient-based back-propagation and evolutionary algorithms, respectively.

Step by step, we analyze the algorithmic complexity for both gradient-based back propagation and evolutionary algorithm to train a layered neural network.

We emphasise that our analysis is on training a m-layered neural network, where $\sum_{i=1}^{m} l_i$ parameters are to be optimized. The procedure to produce update parameters are focused.

For feed-forward pass direction, each layer has experienced such process

$$Z_{i+1} \leftarrow M_{i+1,i} \cdot Z_i, \quad Z_{i+1} \leftarrow f(Z_{i+1}), \tag{S1}$$

where $f(*)$ is the activation function and $M_{i+1,i}$ contains the weights going from layer i to $i+1$. Thus, time complexity is the same as feed-forward case, which in total demands $\mathcal{O}(n_g \sum_{i=1}^{m-1} l_i l_{i+1})$ basic operations and $\mathcal{O}(n_g \sum_{i=1}^{m-1} l_{i+1})$ queries to the inverse activation function.

For back-propagation direction, each layer has experienced such process

$$E_i \leftarrow f'(Z_i - O_i), \quad , D_{i,i-1} \leftarrow E_i \cdot Z_{i-1}, \quad M_{i,i-1} \leftarrow M_{i,i-1} - D_{i,i-1} \tag{S2}$$

where E_{i-1} and $D_{i,i-1}$ are the error terms and adjust matrix. Remarkably, different algorithms are supposed to be employed here and we consider the typical case. Thus, time complexity is $\mathcal{O}(n_g l_i l_{i+1})$ operations, and $\mathcal{O}(n_g l_i)$ queries to the inverse activation function. In total, feed-forward pass algorithm demands $\mathcal{O}(n_g \sum_{i=1}^{m-1} l_i l_{i+1})$ basic operations and $\mathcal{O}(n_g \sum_{i=1}^{m-1} l_{i+1})$ queries to the activation function.

For the differential evolution algorithm, as described by equations in the Method section of the main text, demands $\mathcal{O}(n_e \sum_{i=1}^{m-1} l_i)$, including both the mutation operation and the crossover operation.

S5 Particle Swarm Optimizer

To demonstrate that Darwinian evolutionary theory applies to a wide range of evolutionary algorithms, and the DE algorithm is not a special case. We selected the PSO algorithm (Kennedy & Eberhart, 1995) to replace DE and tested its optimization performance. As shown in Fig. S8, we find that the results for PSO are similar to those for DE in Fig. S2. It further proves the generalization and validity of our theory.

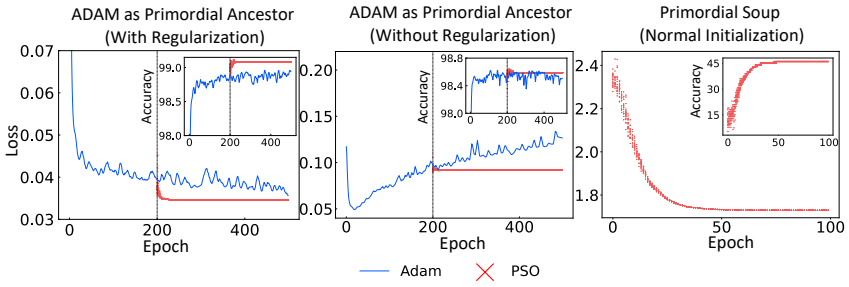

Figure S8: **Other nature-inspired optimizer** To demonstrate the generalization of our proposed method beyond differential evolution, another population-based optimizer, PSO, is used. Similar results in Figure 2 are observed. Hence, it further demonstrates the generalization and validity of the positive impacts of population-based optimizers on deep neural networks.

REFERENCES

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. December 2014.

Charles Darwin. *On the origin of species, 1859*. Routledge, 2004.

J Deng, W Dong, R Socher, L J Li, K Li, and others. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference*, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pp. 770–778, December 2015.

S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. April 2017.

James F Kasting. Earth's early atmosphere. *Science*, 259(5097):920–926, 1993.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pp. 1942–1948. IEEE, 1995.

Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

Y LeCun, L Bottou, Y Bengio, and others. Gradient-based learning applied to document recognition. *Proceedings of the*, 1998.

Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik D Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Multiobjective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 25(2):277–291, 2020.

Stanley L Miller. A production of amino acids under possible primitive earth conditions. *Science*, 117(3046): 528–529, 1953.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pp. 2902–2911. PMLR, 2017.

John Maynard Smith. *The theory of evolution*. Cambridge University Press, 1993.

William R Taylor. Stirring the primordial soup. *Nature*, 434(7034):705–705, 2005.

Shi-Qi Wang, Jie Ye, Jin Meng, Chunxiao Li, Loïc Costeur, Bastien Mennecart, Chi Zhang, Ji Zhang, Manuela Aiglstorfer, Yang Wang, et al. Sexual selection promotes giraffoid head-neck evolution and ecological adaptation. *Science*, 376(6597):eabl8316, 2022.

Tim Whitaker and Darrell Whitley. Sparse mutation decompositions: Fine tuning deep neural networks with subspace evolution. *arXiv preprint arXiv:2302.05832*, 2023.