ODE Perspective on Neural Networks

Wenhao Chai

Jul 26, 2025

The Neural Network (NN) architecture, despite its empirical success, is often viewed as a discrete stack of black-box layers. A growing body of research, however, reframes these powerful models through the lens of continuous-time dynamics, interpreting them as numerical discretizations of Ordinary and Partial Differential Equations (ODEs/PDEs). This perspective provides a powerful theoretical foundation that not only enhances our understanding but also drives practical innovation. This post explores this viewpoint, showing how it has led to architectural improvements like ODE Transformers, provided explanatory frameworks for information flow, and recently culminated in a paradigm shift towards end-to-end, one-step generative modeling with methods like MeanFlows. We argue that this continuous-dynamics view is a key to unlocking the next generation of efficient, interpretable, and powerful models.

1. Neural ODE

Modern neural networks are typically constructed from a sequence of predefined, repeated layers or blocks, where each layer or block applies a transformation to its input. Without loss of generality, we focus on the ResNet [7, 8] family of models that include skip connections, which can be formulated as:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \tag{1}$$

where $\mathbf{h}_t$ denotes the hidden state at layer t, f is a learnable transformation function (typically composed of convolutions, normalization, and non-linearities) with the parameters θ_t. This residual formulation suggests that the evolution of hidden states can be interpreted as a discretization of an underlying continuous transformation.

This formula bears a striking resemblance to the forward Euler method, a fundamental numerical technique for approximating the solution to an ordinary differential equation (ODE):

$$\mathbf{h}_{t+\Delta t} = \mathbf{h}_t + \Delta t \cdot \frac{d\mathbf{h}(t)}{dt} \tag{2}$$

Building upon this intuition, Neural Ordinary Differential Equations (Neural ODEs) [3] propose replacing the discrete sequence of transformations with a continuous dynamical system:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta) \tag{3}$$

where the hidden state $\mathbf{h}(t)$ evolves continuously over time under the dynamics defined by f. The final representation is obtained by solving this ODE from an initial state $\mathbf{h}(0)$ to a target time using an ODE solver. This formulation not only provides a principled approach to modeling depth as a continuous variable but also offers benefits such as adaptive computation and memory efficiency.

Although the formulations appear similar, there is a difference in how parameters are used:

- In ResNet, each layer employs independent parameters $f(\cdot, \theta_t)$, meaning each (θ_t is specific to layer t.
- In Neural ODEs, the dynamics are defined by a single shared function $f(\mathbf{h}(t), t, \theta)$ with shared parameters θ, treating the model as a continuous-time dynamical system.

It is worth noting that scaling the residual branch by a factor such as $1/t$ [5] or learnable [8, 14, 16, 17] further strengthens the connection between ResNets and Neural ODEs. Specifically, this scaling can be interpreted as introducing an explicit step size Δt in the forward Euler discretization. This also connects to the scheduler design in VE-SDE (Variance Exploding Stochastic Differential Equation) [13] in diffusion models, which is constructed to maintain variance stability. In a sense, it aligns with the layer-wise evolution in ResNets that progressively adjusts signal propagation. This implies that ResNet can be viewed as a finite sequence of transformations, each with distinct parameters. Neural ODEs represent a continuous-depth model, effectively modeling an infinitely fine-grained transformation with unified parameters.

While ResNet naturally correspond to a forward Euler discretization of an ODE due to their residual additive form, DenseNet [9] differ substantially. The core design of DenseNet involves concatenating all previous layer outputs, leading to a growing feature dimensionality and non-Markovian dynamics. As such, DenseNet cannot be directly formulated as a standard Neural ODE.

We also briefly introduce two papers: ODE Transformer [11] and OT Transformer [10], providing guidance from the ODE perspective on higher-order solution methods, architectural design, and optimization strategies for Transformers.

Higher-order Solver. Classic Pre-Norm Transformer is mathematically equivalent to a first-order Euler step for solving an ODE. However, the Euler method is prone to truncation error. When stacking many layers (integration steps), this leads to error accumulation and degraded performance especially in deep models. To mitigate this, the authors propose replacing the Euler-based residual block with higher-order ODE solvers. Instead of stacking many first-order approximation layers, it's better to use fewer layers where each layer achieves higher-order precision through multiple internal computations. ODE Transformer uses groups of residual blocks as its basic unit. Within each ODE block, multiple implicit sub-layer computations are performed—analogous to the Runge-Kutta multi-step method. The same parameters are reused across these internal steps, enabling parameter sharing and improving parameter efficiency. For a second-order Runge-Kutta block as an example, we have the formulation:

$$F_1 = f(\mathbf{h}_t, \theta_t) \tag{4}$$

$$F_2 = f(\mathbf{h}_t + F_1, \theta_t) \tag{5}$$

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \frac{1}{2}(F_1 + F_2) \tag{6}$$

Instead of just doing one step like in a normal residual connection (Euler method), it takes a "look ahead", averages the change, and uses that to update $\mathbf{h}_t$.

Optimal Transport. Just like in ODE-based methods such as flow matching, introducing a regularization term like optimal transport (OT) is also a natural idea. To address training instability and the non-uniqueness of solutions, the authors introduce the following regularization term:

$$\text{OT Objective} := \frac{\lambda}{2dn} \int_0^T \|f(\mathbf{h}_t, t)\|_F^2 \, dt \tag{7}$$

While not explicitly computing the Wasserstein distance to a linear path, the regularization term resembles the kinetic energy formulation of Wasserstein-2 and encourages hidden state trajectories to evolve smoothly to an optimal transport flow.

2. Universal Transformer

Recently, approaches such as Universal Transformer (UT) [4] which involve recurrently reusing Transformer layers, have become popular again. This resurgence is driven by the ongoing pursuit of new scaling dimensions. While test-time compute has shown promise in scaling with sequence length, works like the Universal Transformer aim to approximate similar benefits by increasing depth through layer reuse. Among the recent developments in recurrently stacked or depth-reused Transformers, the other two notable branches reinterpret the architecture through the lens of Deep Equilibrium Models (DEQ) [2] and Hopfield networks [12].

Recurrent Depth [6]. This work proposes a decoder-only transformer architecture with a recurrent latent block. At inference time, the block is unrolled many times to produce a final latent, which is decoded via a lightweight coda. All recurrent steps share parameters, enabling the model to scale compute depth at test-time without increasing parameters. Gradients are truncated to the final steps during training for memory efficiency. Empirically, increasing recurrence improves performance on reasoning-heavy tasks such as GSM8K and HumanEval,

Mixture-of-Recursions [1]. Furthermode, a lightweight router is integrated into the model. This router operates at the token level to dynamically decide the "recursion depth" for each token—that is, how many times the shared block should be applied. This allows the model to allocate more computational resources to semantically complex tokens and fewer to simpler ones.

Deep Equilibrium Models [2]. DEQ models treat an infinitely deep Transformer with shared parameters across layers as the solution to a fixed-point equation. Instead of explicitly stacking an infinite number of layers, DEQ directly solves for the equilibrium hidden state using iterative methods such as Newton's method. It show that many sequence models naturally converge to a fixed point in their hidden representations, allowing the model to compute gradients via implicit differentiation without backpropagating through every iteration. This formulation enables scaling in depth while only storing the parameters of a single Transformer layer.

Hopfield Networks [12]. A Hopfield network is a dynamic system characterized by an energy function; it performs iterative updates that converge to an energy minimum, corresponding to a stored memory pattern. In the Transformer context, queries are viewed as initial states, and keys serve as memory patterns. The attention update step pulls the query states closer to energy minima defined by those patterns. Thus, each Transformer layer can be interpreted as performing a single step of Hopfield retrieval.

Although these successful approaches may not all explicitly follow ODE modeling, we are seeing an increasing number of successful cases with layer-wise shared parameters at the architectural level, which will help advance research in Neural ODE.

3. Discussion

Overall, neural ODEs serve as a powerful and intriguing explanatory framework, with notable advances in both theoretical foundations and architectural design in recent years. At the same time, it is worth noting that diffusion models and flow matching approaches can, in general, be understood and modeled through the lens of SDEs and ODEs. For a flow matching model, the inference process along the time axis can essentially be viewed as a depth-recurrent model stacking layers over time, where most parameters are shared, and only a small portion (such as time embeddings) are not—though even these may not be strictly necessary [15]. As a result, many techniques commonly used in flow matching, such as distillation, high-order solvers, and controlled generation, may find renewed relevance when brought back into the neural ODE framework.

Acknowledgments

We would like to acknowledge that some of the insights, content, and references in this blog are adapted from this link and the slides from Kaiming He.

References

[1] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level thinking. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[4] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[5] Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

[6] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[10] Kelvin Kan, Xingjian Li, and Stanley Osher. Ot-transformer: a continuous-time transformer architecture with optimal transport regularization. *arXiv preprint arXiv:2501.18793*, 2025.

[11] Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. Ode transformer: An ordinary differential equation-inspired model for sequence generation. *arXiv preprint arXiv:2203.09176*, 2022.

[12] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[13] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[14] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

[15] Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Is noise conditioning necessary for denoising generative models? *arXiv preprint arXiv:2502.13129*, 2025.

[16] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

[17] Xiao Zhang, Ruoxi Jiang, William Gao, Rebecca Willett, and Michael Maire. Residual connections harm generative representation learning. *arXiv preprint arXiv:2404.10947*, 2024.