IT'S BETTER TO BE SECOND: SECOND-ORDER RESIDUALS PREVENT VANISHING GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The residual connection has long been viewed as crucial for the stable training of deep architectures, from multi-layered perceptrons and ResNets to the transformer that modern large language models (LLMs) are based upon. The common understanding is that it is necessary to preserve gradient flow and speed up training convergence. However, is that really true? In this work, we prove that residual connections do not prevent vanishing (or exploding) gradients. Furthermore, they can lead to inefficient parameter usage by blowing up the output norm over depth. Instead, we show that using second-order residuals, where $\mathbf{x}_{t+1}$ is a function of two previous layers $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, can provably prevent vanishing gradients regardless of model depth. Second-order residuals can be substituted into modern deep architecture for little overhead cost, leading to comparable or better downstream performance on several tasks (synthetic multihop retrieval and multitask parity, as well as large-scale language modeling).

1 INTRODUCTION

Ask any group of people what they think the most important ingredient in cooking is, and the likeliest answer you will receive is the common table salt — a simple ingredient so critical to flavor, it is used in every cuisine around the world. And if you were to ask any group of deep learning practitioners what they think the most important ingredient in model design is, a common answer might be the *residual connection* — a simple idea so critical to effective learning, it is found in almost all common architectures of the last decade, e.g. ResNets (He et al., 2016), transformers (Vaswani et al., 2017), and diffusion models (Ho et al., 2020; Song et al., 2020).

The residual connection is defined as

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

the core idea being that the current layer's input $\mathbf{x}_t$ is added to its own output $\mathbf{f}_t(\mathbf{x}_t)$ before being passed onto the next layer. Early conceptions of this idea were explored here and there in literature, but its first significant usage came with the long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997), where the cell state is updated additively via a residual connection.

What motivates the use of residuals? A central hypothesis underpinning modern machine learning is the importance of *depth*. Composing multiple learnable nonlinear modules creates models that can express more complex functions and richer representations (Bengio, 2009). This can be seen across different model classes. For example, deeper convolutional neural networks can compose increasingly abstract representations from earlier low-level features, which are then useful for downstream discriminative tasks (Zeiler & Fergus, 2014). Depth scaling is also necessary for transformers to perform well on longer contexts, e.g. state tracking and graph connectivity cannot be expressed by fixed-depth transformers (Wang et al., 2024b; Merrill & Sabharwal, 2025).

Depth, however, is both a boon and a curse. Deeper models are also harder to train, because gradients that are backpropagated across an increasing number of layers are exponentially more likely to vanish (or explode) (Glorot & Bengio, 2010). This problem came to a head by the early 2010s, as practitioners pushed model depths to regimes that no longer supported training convergence or led to test-time performances that trailed behind shallower models. It was around this time that the residual connection became popularized by ResNets (He et al., 2016) and highway networks (Srivastava et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2015). The key insight was that adding the layer's input to its output introduced a constant to the gradient norm. This preserved gradient flow by reducing the reliance on what the layer $\mathbf{f}(\cdot)$ learnt. Concretely, one can observe that the $+\mathbf{x}_t$ term in (1) creates an identity term, $\mathbf{I}$, in the Jacobian.

Since then, residuals have become a mainstay in almost all major architectural designs, including the transformer that modern large language models (LLM) are based on. They are commonly used in *every* layer of a deep network, connecting each layer to the next. The use of residuals coexists alongside other design choices that control gradient and output norms, such as normalization (Ioffe & Szegedy, 2015; Zhang & Sennrich, 2019).

In this work, we revisit the longstanding assumption that residual connections are required to train neural networks at depth. First, we show that despite empirical success, using residuals can lead to problems. We prove that, contrary to common belief, **residual connections cannot prevent vanishing** (or exploding) **gradient norms** — indeed, they do not guarantee to solve the very issue they were meant to solve! We also reiterate the idea that residuals lead to increasing output norms, which partially explains recent empirical observations that deeper layers learn less effectively and contribute little to model performance (Gromov et al., 2024; Csordás et al., 2025).

Can we, then, connect model layers in a different way? Is it possible to replace such a central component of modern architecture with an alternative that not only produce functional models, but ones that can scale gracefully with depth? Remarkably, we show that it is. We model the interlayer connectivity of a deep neural network as a difference equation and show that the standard residual corresponds to a first-order difference equation. By replacing it with a **second-order difference equation**, we design a system that **provably prevents gradient norms from vanishing** to 0.

We demonstrate that the second-order connection works surprisingly adequate in practice: on synthetic multi-hop and parity tasks, our approach **far outperforms** the standard residual, and on large-scale language modeling and downstream evaluations, our approach is **comparable** to the standard residual. We believe that our work opens the door to exploring model connectivity in novel ways, imbuing deep networks with desired gradient properties that the standard residual may not possess.

Our contributions can be summarized as follows:

1. We prove that the standard residual connection can lead to both vanishing and exploding gradient norms. We also outline other failure modes of the residual. (Section 3)
2. We model layer connectivity as a second-order difference equation and show that doing so provably leads to non-vanishing gradients. (Section 4)
3. We evaluate our approach on small-scale synthetic tasks, as well as large-scale language modeling and downstream zero-shot evaluations on understanding benchmarks. (Section 5)

2 Related Work

Residual Connections The first widely-used model class to adopt the residual connection is the LSTM (Hochreiter & Schmidhuber, 1997), where gradient propagation is a concern because of the model's recurrent structure. The use of residuals became widespread alongside convolutional neural networks, which were often trained to depths exceeding 100. The most prominent residual networks are ResNets (He et al., 2016), highway networks (Srivastava et al., 2015), and DenseNets (Huang et al., 2017) — indeed, the name is popularized by the former.

Recent works have begun exploring alternatives to the standard residual. Pagliardini et al. (2024) and Heddes et al. (2025) generalize connections to weighted averages of all previous blocks. Ji et al. (2025) propose that residuals can be eliminated altogether and stable gradient benefits can be obtained through sound initialization. Zhu et al. (2024) propose hyper-connections, a generalization of residuals where the strength (weights) of all input-output connections are themselves learnt and the model can even learn multiple sets of weights in any given layer. Our proposal is mathematically different to each of these (we considers connections to two previous layers only) and comes with provable guarantees against gradient vanishing.

Orthogonal Directions More distantly, other avenues of research relate to the control of gradient or output norms as depth scales. One central direction is normalization: both the type of normalization

(e.g. batch norm (Ioffe & Szegedy, 2015) vs RMSNorm (Zhang & Sennrich, 2019)) and where normalization appears in the architecture. While the original transformer uses a post-norm strategy (Vaswani et al., 2017), most modern LLMs are pre-norm (Grattafiori et al., 2024; Yang et al., 2025). More recently, others have proposed more complicated normalization schemes (Wang et al., 2024a; Xie et al., 2023; Zhuo et al., 2025; Sun et al., 2025). Other relevant directions include studying hyperparameter transfer across depth (depth μP) (Yang et al., 2023) and learning on a fixed-norm hypersphere (Loshchilov et al., 2024).

These works are orthogonal to our work in the sense that they intervene on aspects of model design that do not relate to layer connectivity. As such, they can be used alongside our approach.

3 Failure Modes of the Standard Residual Connection

Let us begin by reviewing the residual connection, a ubiquitous component of modern deep architecture such as transformers (Vaswani et al., 2017) and diffusion models (Ho et al., 2020; Song et al., 2020). We denote network activations as $\mathbf{x}_t \in \mathbb{R}^d$ where t indexes the network layers. The standard residual function is

$$\mathbf{x}_{t+1} = \mathbf{f}_t(\mathbf{x}_t) + \mathbf{x}_t \tag{2}$$

where $\mathbf{f}_t : \mathbb{R}^d \to \mathbb{R}^d$ is a (nonlinear) transformation. For example, $\mathbf{f}_t$ in a standard decoder-only transformer will alternate between the self-attention function and the feedforward network (FFN) within each transformer block.

Why has the residual connection proven so useful in building deep neural networks? Observe that the Jacobian of this function is

$$\nabla_{\mathbf{x}_t} \mathbf{x}_{t+1} = \nabla_{\mathbf{x}_t} \mathbf{f}_t + \mathbf{I}. \tag{3}$$

(3) provides some general intuition: the presence of a constant identity term $\mathbf{I}$ in the Jacobian prevents gradients from vanishing even if $\nabla_{\mathbf{x}_t} \mathbf{f}_t \approx \mathbf{0}$. The risk of vanishing gradients grows exponentially as model depth increases:

$$\nabla_{\mathbf{x}_0} \mathbf{x}_{t+1} = \prod_{i=0}^{t} \nabla_{\mathbf{x}_i} \mathbf{g}_i + \mathbf{I}, \tag{4}$$

explaining why the residual connection's ability to preserve gradient flow has proven critical to effective learning at depth.

Our analysis and understanding of the residual connection generally stop here: a gradient-preserving technique that is indispensable in practice. But is it really?

In our work, we shall re-examine this fundamental assumption. First, we show that the explanation above is not fully accurate — the residual connection can, in fact, lead both to vanishing and exploding gradients. Beyond the theoretical concerns, we reiterate a practical issue: allowing activations to grow can lead to inefficient parameter usage in later layers.

3.1 What can go wrong with the residual connection?

Let us first take a closer look at the derivative $\nabla_{\mathbf{x}_0} \mathbf{x}_T$ (for an arbitrary depth T). To build some intuition, we consider a linearization of (2). Formally:

Definition 3.1. We define the **linear first-order residual** as a recurrence

$$\mathbf{x}_{t+1} = \mathbf{W}_t \mathbf{x}_t + \alpha \cdot \mathbf{x}_t \tag{5}$$

for $t \in \mathbb{N}_0$, where $\alpha \in \mathbb{R}$ and each $\mathbf{W}_t \in \mathbb{R}^{d \times d}$.

The dynamics of this system is fully determined by three parameters: the coefficient α, the collection of linear matrices $\{\mathbf{W}_t\}_{t=0}^{T}$, as well as the initial condition $\mathbf{x}_0 \in \mathbb{R}^d$. The standard residual corresponds to the case where $\alpha = 1$. With a bit of straightforward manipulation, we will see that this system can either vanish or explode depending on our choice of parameters.

Proposition 3.2. *For the recurrence defined by (5), there exists some choice of* $\{\mathbf{W}_t\}_{t=0}^T$ *such that* $\forall \mathbf{x}_0 \in \mathbb{R}^d, \forall \alpha \in \mathbb{R}$*:*

1. $||\nabla_{\mathbf{x}_0} \mathbf{x}_T||_2 \in \Omega(2^T)$
2. $||\nabla_{\mathbf{x}_0} \mathbf{x}_T||_2 = 0 \qquad \forall T > 0$

See Appendix A.1 for the proof. Case 1 shows that **gradient norms can grow exponentially** and Case 2 shows that **they can also vanish to 0**. The key idea behind the proof is simply that we can choose $\mathbf{W}_t$ to control how $\mathbf{W}_t \mathbf{x}_t$ aligns with $\mathbf{x}_t$: if the two vectors cancel each other out then the output becomes $\mathbf{0}$ and the system vanishes, whereas if the two vectors have the same direction then the output is doubled and system will grow exponentially.

Since $\{\mathbf{W}_t\}_{t=0}^T$ represent linear approximations of the model's parameters at each layer, Proposition 3.2 implies that regardless of initial condition $\mathbf{x}_0$, a nonlinear model with standard residual connections can fail to converge during training. Not only so, these training pathologies cannot be avoided by simply changing α or even by setting learnable or layer-dependent α_t. Note that this will still be true even in the general nonlinear setting, since (5) is just a special case of (2).

Of course, parameters $\{\mathbf{W}_t\}_{t=0}^T$ are learnt and not arbitrarily chosen. In practice, models do not always take gradient steps towards unstable learning regimes. However, residual connections can create problems even beyond the worst-case scenarios painted in Proposition 3.2. In pre-norm architectures (i.e. inputs to $\mathbf{f}_t$ in (2) are normalized), the residual norm $||\mathbf{x}_t||_2$ grows with depth. As a result, $\mathbf{f}_t(\mathbf{x}_t)$ becomes much smaller than $\mathbf{x}_t$ on deeper layers of the network and later layers make minimal contributions to the output. This asymmetry has been previously noted (Wang et al., 2024a; Xie et al., 2023) and contributes to empirical observations that deeper layers contribute little to model performance (Gromov et al., 2024; Csordás et al., 2025).

Switching to a post-norm architecture (i.e. $\mathbf{x}_{t+1}$ is normalized immediately after (2)) is not the solution either, because the gradient of the normalization function, e.g.

$$\nabla_{\mathbf{x}} \text{RMSNorm}(\mathbf{x}) = \nabla_{\mathbf{x}} \left(\frac{\sqrt{d}}{||\mathbf{x}||_2} \mathbf{x} \right) = \frac{\sqrt{d}}{||\mathbf{x}||_2} \left(\mathbf{I} - \frac{\mathbf{x}\mathbf{x}^\top}{||\mathbf{x}||_2^2} \right) \tag{6}$$

appears T times in a T-layer network. This term shrinks as activations $\mathbf{x}_t$ grow, creating the same problem of exponentially decaying gradients that residuals were meant to solve in the first place.

3.2 What kind of inter-layer connections should our model possess?

We have seen that the standard residual causes model output $\mathbf{x}_t$ to accumulate over depth, leading to inefficient parameter usage on the later layers. It also provides no guarantees on gradient norms in either direction and can lead to both vanishing and exploding gradients.

In this work, we focus on the vanishing gradient problem. Vanishing gradients are, in some sense, a bigger challenge for learning than exploding gradients. Large gradient norms are constrained by model depth and can be mitigated by normalization and learning rate, allowing learning to still take place. However, when gradients vanish the model simply cannot learn. This is true for both the 1-layer and the 100-layer network.

To approach this problem, we will consider inter-layer connections from an architecture-agnostic perspective, abstracting away specific modules or functionality. Observe that an interconnected network, like (2), can be described as a **difference equation** — a recurrence relation that governs how output $\mathbf{x}_t$ grows with depth t. Specifically, the standard residual (2) is a **first-order** non-autonomous difference equation, i.e. where $\mathbf{x}_{t+1}$ is a function of t and $\mathbf{x}_t$ only. In this work, we will relax this constraint and allow residual connections that take the form of higher-order difference equations. In the next section, we will show that second-order recurrences can *provably* prevent gradients from vanishing whilst controlling output growth at the same time.

4 SECOND-ORDER RESIDUAL CONNECTIONS

A second-order difference equation describes $\mathbf{x}_{t+1}$ as a function of t, $\mathbf{x}_t$, and $\mathbf{x}_{t-1}$. Again, let us consider a linearization of the system to build some intuition.

Definition 4.1. We define the **linear second-order residual** as a recurrence

$$\mathbf{x}_{t+1} = \mathbf{W}_t \mathbf{x}_t + \alpha \cdot \mathbf{x}_t + \beta \cdot \mathbf{x}_{t-1} \tag{7}$$

for $t \in \mathbb{N}_0$, where $\alpha, \beta \in \mathbb{R}$ and each $\mathbf{W}_t \in \mathbb{R}^{d \times d}$.

Unlike a first-order recurrence, the second-order recurrence does not suffer from vanishing gradients. To show this formally, let us first rewrite (7) using a change of variables. Let $\mathbf{z}_0 = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_{-1} \end{bmatrix}$. We define the recurrence

$$\mathbf{z}_{t+1} = \begin{bmatrix} \mathbf{W}_t + \alpha \mathbf{I} & \beta \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{z}_t \tag{8}$$

for $t \in \mathbb{N}_0$, where α, β, and $\{\mathbf{W}_t\}_{t=0}^T$ are identically defined as in (7). It is clear that (7) and (8) describe the same recurrence relation if the initial conditions $(\mathbf{x}_{-1}, \mathbf{x}_0)$ are identical. (8) conveniently allows us to describe our desired property:

Proposition 4.2. *For the recurrence defined by (8), if $|\beta| \geq 1$, then $\forall \alpha \in \mathbb{R}$, $\forall \mathbf{x}_{-1}, \mathbf{x}_0 \in \mathbb{R}^d$:*

$$\inf_{\{\mathbf{W}_t\}_{t=0}^T} ||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_2 = 1 \tag{9}$$

See Appendix A.2 for the proof. Proposition 4.2 shows that **gradients do not vanish for any collection of $\{\mathbf{W}_t\}_{t=0}^T$**, i.e. regardless of what transformations are being applied to $\mathbf{x}_t$ on any layer. This statement is true for all possible initializations $(\mathbf{x}_{-1}, \mathbf{x}_0)$, with the only requirement being that $|\beta| \geq 1$.

What exactly has changed here? Here is an intuitive picture: earlier in the first-order recurrence, observe that if $\mathbf{W}_t \mathbf{x}_t$ cancels out $\mathbf{x}_t$ then $\mathbf{x}_{t+1} = \mathbf{0}$. Since the next layer $\mathbf{x}_{t+2}$ only depends on $\mathbf{x}_{t+1}$, the system cannot recover from vanishing. This is another reason why vanishing gradients are a bigger problem than exploding gradients — gradients can vanish (irrevocably) at any single layer, whereas large norms can be mitigated by future layers $\mathbf{W}_{>t}$.

In the second-order recurrence, even if $\mathbf{x}_{t+1} = \mathbf{0}$, the next layer $\mathbf{x}_{t+2}$ takes $\mathbf{x}_t$ as input and so does not vanish. Importantly, $\mathbf{x}_t \neq \mathbf{0}$, otherwise $\mathbf{W}_t \mathbf{x}_t$ could not have canceled out $\mathbf{x}_t$ *(unless $\mathbf{x}_{-1} = \mathbf{x}_0 = \mathbf{W}_t = \mathbf{0}$ and the entire system is trivially $\mathbf{0}$)*. By taking two previous layers as input, the second-order system safeguards against vanishing: any layer with output $\mathbf{0}$ has no contribution towards the next layer and thus prevents the next layer from vanishing.

We conclude with a couple of remarks about the usefulness of Proposition 4.2 towards the actual models that we care about building.

Remark 4.3. Although (7) and (8) are linear approximations (where we have substituted the nonlinear $\mathbf{f}_t$ with $\mathbf{W}_t$), Proposition 4.2 still holds true for the general nonlinear case. See the proof in Appendix A.2.

Naturally, one might ask if Proposition 4.2 is sufficient, since the gradient that we actually care about optimizing is $\nabla_{\mathbf{W}_t} L(\mathbf{x}_T)$, i.e. the gradient of network parameters with respect to some loss objective L, and not $\nabla_{\mathbf{z}_0} \mathbf{z}_T$. Corollary 4.4 shows that indeed it is:

Corollary 4.4. *For the recurrence defined by (8), if $|\beta| \geq 1$, then $\forall \alpha \in \mathbb{R}$, $\forall \mathbf{x}_{-1}, \mathbf{x}_0 \in \mathbb{R}^d$:*

$$\inf_{\{\mathbf{W}_t\}_{t=0}^{T}} ||\nabla_{\mathbf{W}_t} L(\mathbf{x}_T)||_2 = ||\nabla_{\mathbf{x}_T} L(\mathbf{x}_T)||_2 \cdot ||\mathbf{x}_t||_2 \tag{10}$$

for all $0 \leq t < T$ and any continuous loss objective $L : \mathbb{R}^d \to \mathbb{R}$.

See Appendix A.3 for the proof, which is just a straightforward application of the chain rule. Corollary 4.4 implies that so long as the loss objective L itself does not have a vanishing gradient with respect to the model's final output $\mathbf{x}_T$, and that the model's intermediate outputs $\mathbf{x}_t$ do not vanish, then the gradient with respect to parameters will not vanish during optimization.

One might also question whether, in preventing gradients from vanishing, we have erred the other direction by making gradient norms significantly larger. Second-order residual connections can still lead to exponentially growing gradient norms. However, if we impose additional conditions on α and β, we can show that **the upper bound on gradient norms of the second-order system will still be smaller than that of the standard first-order system**.

Proposition 4.5. *Let $||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,F}$ be the gradient norm for the first-order recurrence defined by (5) and assume $\alpha = 1$. Let $||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,S}$ be the gradient norm for the second-order recurrence defined by (8) and assume $\alpha = 0$ and $|\beta| = 1$. Then $\forall \mathbf{x}_{-1}, \mathbf{x}_0 \in \mathbb{R}^d$:*

$$\sup_{\{\mathbf{W}_t\}_{t=0}^{T}} ||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,S} \leq \sup_{\{\mathbf{W}_t\}_{t=0}^{T}} ||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,F} \tag{11}$$

See Appendix A.4 for the proof. The rough intuition here in the 1D case is that the fastest the first-order residual can grow is the base-2 exponential series 2^t, whereas the fastest the second-order residual can grow is the Fibonacci sequence, which corresponds to a (slower) base-ϕ exponential series ϕ^t *(where $\phi \approx 1.618$ is the golden ratio)*.

4.1 Practical Implementation

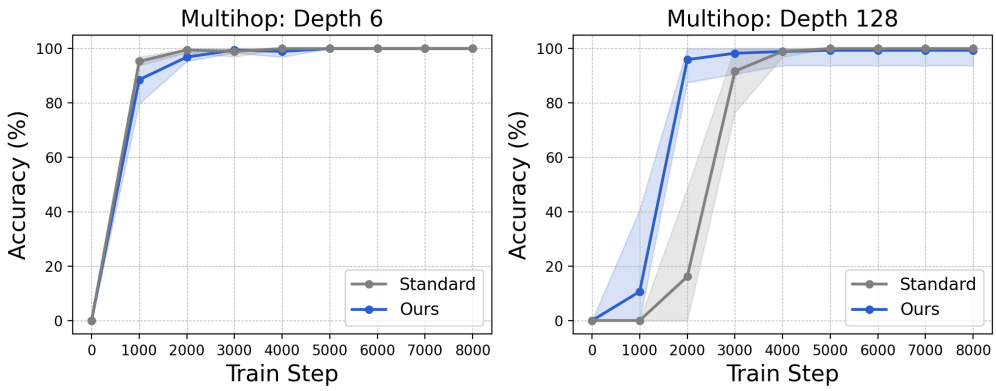
Having established some favorable properties of the second-order residual connection (7), we now provide a concrete instantiation. Our specific formulation is

$$\mathbf{x}_{t+1} = \sqrt{d} \cdot \tanh\left(\frac{\mathbf{f}_t(\mathbf{x}_t)}{\sqrt{d}}\right) - \mathbf{x}_{t-1} \tag{12}$$

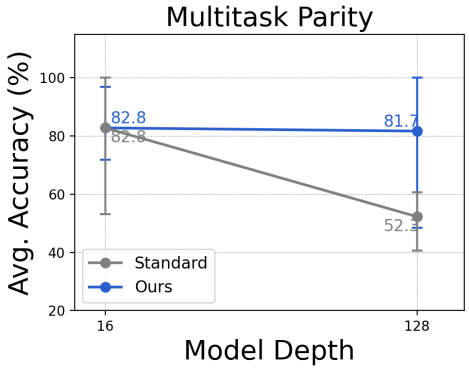
This formulation corresponds to a choice of $\alpha = 0$ and $\beta = -1$, which follows directly from Propositions 4.2 and 4.5. We choose to use $\beta = -1$ instead of $\beta = 1$ to control the output norm, following the discussion in Section 3. $\mathbf{f}_t(\cdot)$ is the model-specific module (e.g. self-attention and the FFN for transformers) and wrapping $\mathbf{f}_t(\mathbf{x}_t)$ with the $\tanh(\cdot)$ function also bounds the output norm. The inner $\frac{1}{\sqrt{d}}$ coefficient prevents the nonlinearity from saturating and the outer $\sqrt{d}$ coefficient ensures that gradient norms are not too small, i.e. the model does not learn too slowly. We initialize $\mathbf{x}_{-1}$ as the input to the model (i.e. the token plus positional embeddings) and $\mathbf{x}_0 = -\mathbf{x}_{-1}$.

Equation (12) can be substituted in a plug-and-play manner into any model in lieu of the standard residual connection, with no additional modifications required. Since growing output norms are not a concern for our approach, we use the older Xavier initialization scheme and initialize all parameters with the same variance (Huang et al., 2020), as opposed to depth-dependent initialization schemes for the standard residual (e.g. Black et al., 2022). We ablate for this in Appendix C.1. In all our experiments, learning rate and other hyperparameters remain unchanged in any pairwise comparison between the standard residual and our approach.

We note that the additional computational cost to implement Equation (12) is negligible. We perform both runtime and memory analyses of both systems in Appendix C.2, including a FLOP analysis.



(a)



(b)

Figure 1: **(a)** Results on the multihop retrieval task, showing test accuracy as a function of training steps. Each line plot shows variance across 10 runs of that model. **(b)** Results on the multitask parity task. Each point on the plot shows the average accuracy across 10 runs of that model, with the error bars denoting the worst and best runs' accuracies.

5 Experiments

We conduct two broad types of experiments: **(i)** small-scale synthetic tasks testing specific skills like multi-hop retrieval and multitask parity, and **(ii)** large-scale language modeling along with zero-shot downstream evaluations. Further experimental details can be found in Appendix B.

5.1 Synthetic Tasks

The goal of these experiments is to verify the second-order residual in a controlled setting with synthetic data, confirming that it can be used to train models that generalize well, especially at higher depths. We consider two tasks below.

Task 1: Multitask Parity The multitask parity problem is a generalization of sparse parity (Barak et al., 2022; Michaud, 2026). The context consists of a sequence of N binary bits (i.e. vocab size $|V| = 2$). The first K bits one-hot encode the task k. Each task chooses a predetermined subset S_k of size L from the remaining $N - K$ bits, and the goal is to compute the parity of this subset of bits. It is known that parity is a difficult problem for transformers (Hahn, 2020) and the multitask version is challenging even on small $\{N, K, L\}$.

To this end, we train standard decoder-only, pre-norm dense transformers on a synthetically generated dataset for the task setting $\{N = 15, K = 5, L = 3\}$. The task k and subsets S_k for each task are uniformly sampled in both train and test datasets. In order to truly test that our approach works well at depth, we compare the second-order residual and the standard residual on two settings: "shallow"

(depth $D = 16$, width $W = 64$) and "deep" (depth $D = 128$, width $W = 256$). All other model details are kept identical. There are therefore 4 settings (2 depths $\times$ 2 residual types). For each setting, we compute 10 runs (on the same dataset but across random model initializations) and measure the average, worst, and best accuracies.

Figure 1**(b)** displays the results of this experiment. On the shallow setting where $D = 16$, both the standard and second-order models achieve similar performance ($\approx 80\%$), with the best run generalizing perfectly. However, as we increase the depth to 128, the **standard residual model experiences a massive drop in accuracy whereas our model retains its accuracy**. We can see that the standard residual does not scale well to large depths, whereas using the second-order residual connection enables the model to train and generalize even as depth scales by a magnitude.

Task 2: Multi-Hop Retrieval We consider a synthetic multi-hop retrieval task as follows. The context is a string of size N of the format `...[answer][ref_1]...[ref_1][ref_2]...[ref_k][answer]`, where the goal is to output `[answer]` at the end. The intended solution is to trace `[answer]` backwards across K reference keys `[ref_k]` to the `[answer]`, and `...` denotes variable-length random substrings in between references. The vocabulary size is 26. The answer and keys are randomly sampled but their lengths L is fixed. The task aims to test the model's ability to perform K-hop retrieval.

Again, we train standard decoder-only, pre-norm dense transformers on a synthetically generated dataset for the task setting $\{N = 165, K = 3, L = 5\}$. We compare the second-order residual and the standard residual on two settings: "shallow" (depth $D = 6$, width $W = 64$) and "deep" (depth $D = 128$, width $W = 256$). Again, all other model details are kept identical and we compute 10 runs (on the same dataset but across random model initializations) per setting.

Figure 1**(a)** displays the results of this experiment. For this task, all settings eventually achieve perfect test accuracy. On the shallow setting, both the standard residual and the second-order residual converge to the correct solution equally fast. However, as we scale depth to 128, **the standard residual converges much more slowly than our approach**. This shows that using the second-order residual connection brings optimization speed gains on this task.

5.2 Large-Scale Language Modeling

The goal of these experiments is to verify that the models using the second-order residual can perform language modeling and understanding at scale, especially at higher depths. To this end, we train standard decoder-only, pre-norm dense transformers on 10B tokens from the FineWeb-Edu (Penedo et al., 2024) dataset, which is around 1.6 times of the Chinchilla-optimal ratio (Hoffmann et al., 2022) for the largest models. We use a context length of 1024 and batch size of 0.5M tokens (Brown et al., 2020).

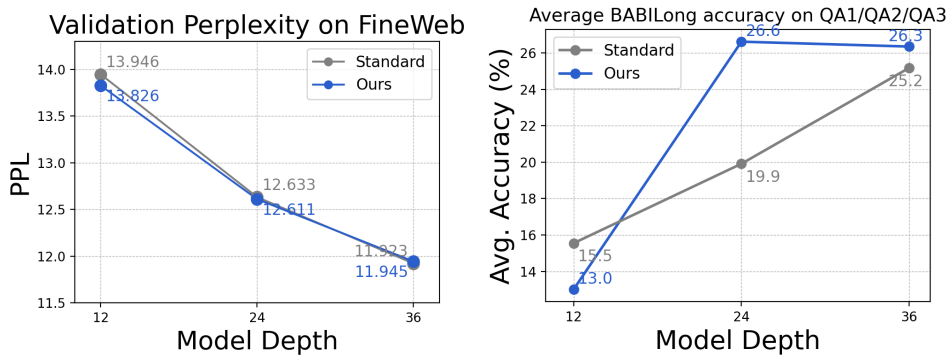
Similarly, since our goal is to verify that our approach works as depth scales, we compare models at depths $D = \{12, 24, 36\}$, all at width $W = 768$ with $H = 12$ heads. For each depth, we run both the standard residual (2) and our second-order residual (12). All other model details are kept identical; the only difference being that we omit the parameter initialization scheme commonly used in transformers to correct for growing residual norms (Black et al., 2022) and initialize all parameters identically *(refer to the discussion in Section 4.1)*.

Main Result: Language Modeling and Understanding Table 1 shows three sets of evaluations: **(i)** (in-distribution) validation perplexity on FineWeb, **(ii)** out-of-distribution zero-shot perplexity on WikiText-103 (Merity et al., 2016) and LAMBADA (Paperno et al., 2016), and **(iii)** zero-shot evaluation on 10 common reasoning and understanding benchmarks. The key result here is that **the second-order residual is comparable to that of the standard residual even as depth scales**, edging out the standard model on validation perplexity for the 12- and 24-layer networks, and slightly trailing on the 36-layer networks. Zero-shot evaluation accuracies also reflect a similar story, with both models performing comparably. Figure 2**(a)** replicates the FineWeb validation perplexities.

Result: State-Tracking Reasoning We also evaluate the same models on the BABILong benchmark, which evaluates the models on a suite of state-tracking reasoning tasks (Kuratov et al., 2024). Figure 2**(b)** plots the zero-shot accuracies on three BABILong tasks (QA1/QA2/QA3), which eval-

Model	FW↓	Wiki↓	LMB↓	AC↑	AE↑	BQ↑	CP↑	HS↑	LB↑	OQ↑	PQ↑	SQ↑	WG↑	Avg.↑
Standard-12D	13.95	68.32	**41.20**	24.2	46.8	56.6	61.0	33.1	27.7	**32.4**	62.6	66.7	**52.5**	46.4
Ours-12D	**13.83**	**40.65**	73.30	24.3	46.6	**61.4**	**66.0**	33.5	26.9	28.8	62.8	**67.7**	48.9	**46.7**
Standard-24D	12.63	35.62	**52.02**	26.5	48.6	**59.3**	66.0	35.8	**30.7**	31.0	63.7	69.5	50.2	**48.1**
Ours-24D	**12.61**	**35.35**	56.84	25.8	48.6	55.2	**70.0**	35.9	28.5	30.8	**64.9**	**71.2**	50.0	**48.1**
Standard-36D	**11.92**	**32.62**	45.37	26.5	50.2	**60.9**	**69.0**	38.1	31.2	31.6	64.9	71.5	51.9	**49.6**
Ours-36D	11.94	32.91	**44.79**	25.9	50.8	59.7	68.0	38.1	31.5	32.4	64.6	71.9	52.2	49.5

Table 1: Validation perplexity *(↓ is better)* on FineWeb, zero-shot perplexity on WikiText and LAMBADA, as well as zero-shot accuracy *(↑ is better)* on 10 common-sense reasoning benchmarks. To avoid clutter, numbers are bolded for the individual reasoning benchmarks only if the difference in accuracy ≥ 1.0.



(a) (b)

Figure 2: **(a)** Validation perplexity on FineWeb. **(b)** Average zero-shot accuracies across 3 tasks (QA1/QA2/QA3) from the BABILong state-tracking reasoning benchmark.

uates the model's ability to track and retrieve the correct answer when there are 1/2/3 supporting facts in the entire context of up to 68 facts. The key takeaway here is that **the second-order residual outperforms the standard residual as depth scales**, displaying higher accuracies on the 24- and 36-layer networks. Despite having comparable (24-) or lower (36-) perplexities on FineWeb, our deeper models out-competes the standard residuals on this benchmark, showing that they may have learnt specific skills better, e.g. state-tracking.

6 Discussion: Tradeoffs and Insights

In our view, it is a surprising result that one can replace the standard residual and still produce functional transformers that learn well, considering how ubiquitious the residual connection has become and how central it is viewed as for good optimization. While the synthetic experiments in Section 5.1 show a significant improvement over the standard residual on large-depth models, large-scale language experiments in Section 5.2 show more modest or mixed results. What is the difference? Is it possible to further iterate upon Equation (12)?

One hypothesis is that Equation (12) creates oscillatory training dynamics at the start of training. This is a tradeoff in exchange for preventing gradients from vanishing and having more bounded output norms. Whether one is more important than the other may depend on the task at hand. In particular, we note that our approach is stronger on the downstream reasoning tasks (Figure 2(**b**)) than it is on validation perplexity. Our approach also outperforms on the synthetic tasks that measure concrete retrieval and reasoning skills. As such, Equation (12) may be more useful for learning discrete (reasoning) skills as opposed to an average metric such as perplexity. Nevertheless, we believe that our work has opened the door towards exploring richer interlayer connectivity beyond the standard residual. We hope to explore this theme in future work.

AI USE STATEMENT

In this work, we have **not** used generative AI tools for any of the tasks with required disclosure under the ICLR 2027 AI Policy for Authors. Namely, we did not use these tools to: generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results.

We have used generative AI tools for some tasks with recommended disclosure: **(i)** analyzing existing literature and asking questions with known answers, **(ii)** generating boilerplate code to run experiments (e.g. training and evaluation loops) and produce result plots (Figures 1 and 2), **(iii)** vetting the paper for grammatical errors and typos *after* the entire draft was written, **(iv)** generating a mock review of the paper to surface areas for improvements *after* the entire draft was written.

We have reviewed all AI-assisted work for accuracy and correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

There are no specific ethical concerns raised by this work beyond those inherent to all machine learning and AI research. Our work complies fully with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We are committed to reproducible research. Section 5 and Appendix B contain all experimental details. In future, we also to publish all the code needed to reproduce our results along with the camera-ready version of our paper.

REFERENCES

Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2 (1):1–127, 2009.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers, 2024. *URL https://arxiv. org/abs/2403.17887*, 2024.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

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*, pp. 770–778, 2016.

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, MohammadHossein Bateni, and Vahab Mirrokni. Deepcrossattention: Supercharging transformer residual connections. *arXiv preprint arXiv:2502.06785*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

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*, pp. 4700–4708, 2017.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pp. 4475–4483. PMLR, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Yiping Ji, James Martens, Jianqiao Zheng, Ziqin Zhou, Peyman Moghadam, Xinyu Zhang, Hemanth Saratchandran, and Simon Lucey. Cutting the skip: Training residual-free transformers. *arXiv preprint arXiv:2510.00345*, 2025.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

Eric J. Michaud. On neural scaling and the quanta hypothesis. `https://ericjmichaud.com/quanta/`, January 2026. Accessed: 2026-05-07.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

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.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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, 2024a.

Mingze Wang et al. Understanding the expressive power and mechanisms of transformer for sequence modeling. *Advances in Neural Information Processing Systems*, 37:25781–25856, 2024b.

Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. Residual: Transformer with dual residual connections. *arXiv preprint arXiv:2304.14802*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

Zhijian Zhuo, Yutao Zeng, Ya Wang, Sijun Zhang, Jian Yang, Xiaoqing Li, Xun Zhou, and Jinwen Ma. Hybridnorm: Towards stable and efficient transformer training via hybrid normalization. *arXiv preprint arXiv:2503.04598*, 2025.

A PROOFS

A.1 PROPOSITION 3.2

Proof. Both cases can be shown as follows:

1. Let $\mathbf{W}_t = (2-\alpha)\mathbf{I}$ for all t. Then[1] :

$$||\nabla_{\mathbf{x}_0}\mathbf{x}_T||_2 = \left|\left|\prod_{i=T-1}^{0}\nabla_{\mathbf{x}_i}\mathbf{x}_{i+1}\right|\right|_2 = \left|\left|\prod_{i=T-1}^{0}2\cdot\mathbf{I}\right|\right|_2 = ||2^T\cdot\mathbf{I}||_2 = 2^T \tag{13}$$

2. Let $\mathbf{W}_0 = -\alpha\mathbf{I}$. Then:

$$||\nabla_{\mathbf{x}_0}\mathbf{x}_T||_2 = ||\nabla_{\mathbf{x}_1}\mathbf{x}_T \circ \nabla_{\mathbf{x}_0}\mathbf{x}_1||_2 = ||\nabla_{\mathbf{x}_1}\mathbf{x}_T \circ \mathbf{0}||_2 = 0 \tag{14}$$

□

A.2 PROPOSITION 4.2

Proof. We can see that

$$\nabla_{\mathbf{z}_t}\mathbf{z}_{t+1} = \begin{bmatrix}\mathbf{W}_t + \alpha\mathbf{I} & \beta\mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix} \tag{15}$$

and

$$\det\left(\nabla_{\mathbf{z}_t}\mathbf{z}_{t+1}\right) = \det\left((\mathbf{W}_t + \alpha\mathbf{I})\circ\mathbf{0} - \beta\mathbf{I}\circ\mathbf{I}\right) = (-\beta)^d, \tag{16}$$

$$\det(\nabla_{\mathbf{z}_0}\mathbf{z}_T) = \prod_{i=0}^{T-1}\det(\nabla_{\mathbf{z}_t}\mathbf{z}_{t+1}) = (-\beta)^{Td}. \tag{17}$$

Now, $|\det\left(\nabla_{\mathbf{z}_0}\mathbf{z}_T\right)| = \sigma_1\cdot\dots\cdot\sigma_d$ is the product of singular values σ_i of $\nabla_{\mathbf{z}_0}\mathbf{z}_T$. If $\beta \geq 1$, then $|\det(\nabla_{\mathbf{z}_0}\mathbf{z}_T)| = |\beta^{Td}| \geq 1$, which implies that the maximal singular value $\sigma_{\max} \geq 1$, and so $||\nabla_{\mathbf{z}_0}\mathbf{z}_T||_2 = \sigma_{\max} \geq 1$. □

Note that we can use the formula $\mathbf{AD} - \mathbf{BC}$ for the determinant of a 2×2 block matrix in (16) since all 4 submatrices commute with each other.

Note also that the linearity of the system is not necessary for the proof to work. Substituting $\mathbf{W}_t\mathbf{x}_t$ with $\mathbf{g}_t(\mathbf{x}_t)$ in (7) only affects the identity of the top left submatrix in (15) *(since* $\mathbf{g}_t$ *only operates on* $\mathbf{x}_t$ *and not* $\mathbf{x}_{t-1}$*)*, which does not matter for the proof to work.

A.3 COROLLARY 4.4

Proof. By the chain rule, we have:

$$\begin{aligned}
\nabla_{\mathbf{W}_t}L(\mathbf{x}_T) &= \nabla_{\mathbf{z}_{t+2}}L(\mathbf{x}_T)\circ\nabla_{\mathbf{x}_{t+1}}\mathbf{z}_{t+2}\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1}\\
&\quad + \nabla_{\mathbf{z}_{t+2}}L(\mathbf{x}_T)\circ\nabla_{\mathbf{z}_{t+1}}\mathbf{z}_{t+2}\circ\nabla_{\mathbf{x}_{t+1}}\mathbf{z}_{t+1}\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1} &(18)\\
&= \nabla_{\mathbf{z}_{t+2}}L(\mathbf{x}_T)\circ\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} + \begin{bmatrix}\mathbf{W}_t+\alpha\mathbf{I} & \beta\mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}\right)\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1} &(19)\\
&= \nabla_{\mathbf{z}_{t+2}}L(\mathbf{x}_T)\circ\begin{bmatrix}\mathbf{W}_t+\alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1} &(20)\\
&= \nabla_{\mathbf{z}_T}L(\mathbf{x}_T)\circ\nabla_{\mathbf{z}_{t+2}}\mathbf{z}_T\circ\begin{bmatrix}\mathbf{W}_t+\alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1} &(21)\\
&= \begin{bmatrix}\nabla_{\mathbf{x}_T}L(\mathbf{x}_T)\\ \mathbf{0}\end{bmatrix}^\top\circ\nabla_{\mathbf{z}_{t+2}}\mathbf{z}_T\circ\begin{bmatrix}\mathbf{W}_t+\alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\circ\nabla_{\mathbf{W}_t}\mathbf{x}_{t+1} &(22)
\end{aligned}$$

[1]Where necessary, we abuse notation slightly and show a decreasing step size on the product notation to ensure commutation consistent with the chain rule.

We can then bound the spectral norm:

$$||\nabla_{\mathbf{W}_t} L(\mathbf{x}_T)||_2 \geq \sigma_{\min}\left(\begin{bmatrix}\nabla_{\mathbf{x}_T} L(\mathbf{x}_T)\\ \mathbf{0}\end{bmatrix}^\top\right) \cdot \left|\left|\nabla_{\mathbf{z}_{t+2}} \mathbf{z}_T\right|\right|_2 \cdot \sigma_{\min}\left(\begin{bmatrix}\mathbf{W}_t + \alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\right) \cdot \sigma_{\min}\left(\nabla_{\mathbf{W}_t} \mathbf{x}_{t+1}\right) \tag{23}$$

$$= ||\nabla_{\mathbf{x}_T} L(\mathbf{x}_T)||_2 \cdot 1 \cdot 2 \cdot ||\mathbf{x}_t||_2 \tag{24}$$

$$= 2 \cdot ||\nabla_{\mathbf{x}_T} L(\mathbf{x}_T)||_2 \cdot ||\mathbf{x}_t||_2 \tag{25}$$

Note that (23) follows from repeated applications of the identities $||AB||_2 \geq ||A||_2 \cdot \sigma_{\min}(B)$ and $||AB||_2 \geq \sigma_{\min}(A) \cdot ||B||_2$. To get to (24) , the second term follows directly from Proposition 4.2 and the third term can be derived as:

$$\sigma_{\min}\left(\begin{bmatrix}\mathbf{W}_t + \alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\right) = \sqrt{\lambda_{\min}\left(\begin{bmatrix}\mathbf{W}_t + \alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}^\top \begin{bmatrix}\mathbf{W}_t + \alpha\mathbf{I}\\ 2\mathbf{I}\end{bmatrix}\right)} \tag{26}$$

$$= \sqrt{\lambda_{\min}\left((\mathbf{W}_t + \alpha\mathbf{I})^\top(\mathbf{W}_t + \alpha\mathbf{I}) + 4\mathbf{I}\right)} \tag{27}$$

$$= \sqrt{\lambda_{\min}\left((\mathbf{W}_t + \alpha\mathbf{I})^\top(\mathbf{W}_t + \alpha\mathbf{I})\right) + 4} \tag{28}$$

$$= \sqrt{\sigma^2_{\min}(\mathbf{W}_t + \alpha\mathbf{I}) + 4} \tag{29}$$

$$\geq 2 \tag{30}$$

□

A.4 PROPOSITION 4.5

Proof. Let $K = \sup_t ||\mathbf{W}_t||_2$ upper bound the spectral norms of all collections of matrices $\{\mathbf{W}_t\}_{t=0}^T$. K must exist since $T < \infty$.

First we consider our system (8) where $\alpha = 0$ and $|\beta| = 1$. We can see that:

$$||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,S} \leq \prod_{i=0}^{T-1} ||\nabla_{\mathbf{z}_i} \mathbf{z}_{i+1}||_2 \tag{31}$$

$$= \prod_{i=0}^{T-1} \left|\left|\begin{bmatrix}\mathbf{W}_i & \beta\mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\right|\right|_2 \tag{32}$$

$$\leq \prod_{i=0}^{T-1} \left|\left|\begin{bmatrix}||\mathbf{W}_i||_2 & ||\beta\mathbf{I}||_2\\ ||\mathbf{I}||_2 & ||\mathbf{0}||_2\end{bmatrix}\right|\right|_2 \tag{33}$$

$$= \prod_{i=0}^{T-1} \left|\left|\begin{bmatrix}||\mathbf{W}_i||_2 & 1\\ 1 & 0\end{bmatrix}\right|\right|_2 \tag{34}$$

$$= \prod_{i=0}^{T-1} \sqrt{\lambda_{\max}\left(\begin{bmatrix}||\mathbf{W}_i||_2 & 1\\ 1 & 0\end{bmatrix}^\top \begin{bmatrix}||\mathbf{W}_i||_2 & 1\\ 1 & 0\end{bmatrix}\right)} \tag{35}$$

Define the 2×2 matrix

$$\mathbf{A}_i = \begin{bmatrix}||\mathbf{W}_i||_2 & 1\\ 1 & 0\end{bmatrix}^\top \begin{bmatrix}||\mathbf{W}_i||_2 & 1\\ 1 & 0\end{bmatrix} = \begin{bmatrix}||\mathbf{W}_i||_2^2 + 1 & ||\mathbf{W}_i||_2\\ ||\mathbf{W}_i||_2 & 1\end{bmatrix}. \tag{36}$$

Our goal is to upper bound the largest eigenvalue of $\mathbf{A}_i$. The characteristic polynomial of $\mathbf{A}_i$ is:

$$\lambda^2 - \text{Tr}(\mathbf{A}_i)\lambda + \det(\mathbf{A}_i) = \lambda^2 - (||\mathbf{W}_i||_2^2 + 1 + 1)\lambda + ||\mathbf{W}_i||_2^2 + 1 - ||\mathbf{W}_i||_2^2 \tag{37}$$

$$= \lambda^2 - (||\mathbf{W}_i||_2^2 + 2)\lambda + 1 \tag{38}$$

The roots of this quadratic equation are:

$$\lambda = \frac{(||\mathbf{W}_i||_2^2 + 2) \pm \sqrt{(||\mathbf{W}_i||_2^2 + 2)^2 - 4}}{2} \tag{39}$$

Now we aim to upper bound λ. Since $\sup_t ||\mathbf{W}_t||_2 = K$, then

$$\lambda_{\max} \leq \frac{K^2 + 2 + \sqrt{(K^2+2)^2 - 4}}{2} \tag{40}$$

Note that $\lambda_{\max}$ is always real. Substituting $\lambda_{\max}$ back into (35), we have:

$$||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,S} \leq \prod_{i=0}^{T-1} \sqrt{\frac{K^2 + 2 + \sqrt{(K^2+2)^2 - 4}}{2}} \tag{41}$$

$$= \left(\sqrt{\frac{K^2 + 2 + \sqrt{(K^2+2)^2 - 4}}{2}} \right)^T \tag{42}$$

Now let us consider the first-order system (5), which we first define in terms of $\mathbf{z}$ analogously:

$$\mathbf{z}_{t+1} = \begin{bmatrix} \mathbf{W}_t + \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{z}_t, \tag{43}$$

where we can assign a dummy value to $\mathbf{x}_{-1}$.

It suffices to show that the gradient norm $||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,F}$ for the first-order system can exceed (42) for some collection $\{\mathbf{W}_t\}_{t=0}^T$. Consider the case where $||\mathbf{W}_t||_2 = K \cdot \mathbf{I}$ for all t. Then:

$$||\nabla_{\mathbf{z}_0} \mathbf{z}_T||_{2,F} = \left|\left| \prod_{i=T-1}^{0} \nabla_{\mathbf{z}_i} \mathbf{z}_{i+1} \right|\right|_2 \tag{44}$$

$$= \left|\left| \prod_{i=T-1}^{0} \begin{bmatrix} (K+1)\mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \right|\right|_2 \tag{45}$$

$$= \left|\left| \begin{bmatrix} (K+1)^T\mathbf{I} & \mathbf{0} \\ (K+1)^{T-1}\mathbf{I} & \mathbf{0} \end{bmatrix} \right|\right|_2 \tag{46}$$

$$= \sqrt{\lambda_{\max} \left(\begin{bmatrix} (K+1)^T\mathbf{I} & \mathbf{0} \\ (K+1)^{T-1}\mathbf{I} & \mathbf{0} \end{bmatrix}^\top \begin{bmatrix} (K+1)^T\mathbf{I} & \mathbf{0} \\ (K+1)^{T-1}\mathbf{I} & \mathbf{0} \end{bmatrix} \right)} \tag{47}$$

$$= \sqrt{\lambda_{\max} \left(\begin{bmatrix} ((K+1)^{2T} + (K+1)^{2T-2})\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right)} \tag{48}$$

$$= \sqrt{(K+1)^{2T} + (K+1)^{2T-2}} \tag{49}$$

$$= (K+1)^{T-1}\sqrt{1 + (K+1)^2} \tag{50}$$

Finally observe that

$$\left(\sqrt{\frac{K^2 + 2 + \sqrt{(K^2+2)^2 - 4}}{2}} \right)^T \leq (K+1)^{T-1}\sqrt{1 + (K+1)^2} \tag{51}$$

for all $K > 0$ and $T > 0$. □

One interesting observation here is when $K = 1$, e.g. if $\{\mathbf{W}_t\}_{t=0}^T$ are all orthonormal. Then $\lambda_{\max}$ for the second-order system, i.e. (40) above, is simply just $1 + \phi$ where $\phi \approx 1.619$ is the golden ratio. The intuition for the appearance of ϕ is that the fastest the second-order system $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{x}_{t-1}$ can grow is if $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ align perfectly. In the 1D case, such a system corresponds to the Fibonacci sequence, which grows more slowly than 2^x (the first-order system).

B Experimental Details

On all experiments, we use the AdamW optimizer, the GPT2 tokenizer, and weight decay of 0.1. For the large-scale models, we use gradient clipping of 1.0. For the 12-layer models, we use a learning rate of 3×10^{-3} for both methods. For the 24- and 36-layer models, we use a learning rate of 1×10^{-3}. In Table 1, HS denotes HellaSwag, PQ denotes PIQA, AE denotes ARC-Easy, AC denotes ARC-Challenge, WG denotes Winogrande, OQA denotes OpenBookQA, LMB denotes LAMBADA.

C Additional Results

C.1 Ablation for Initialization Scheme

We ablate for initialization scheme below for the 12-layer transformer using the language modeling setting in Section 5.2. In Table 2 below, we report both the validation perplexity on FineWeb and the average zero-shot accuracy (on the same downstream tasks as shown in Table 1):

Model	**FW**↓	**Avg.**↑
Standard-12D, Depth-Dependent Initialization	13.95	46.4
Ours-12D, Uniform Initialization	**13.83**	**46.7**
Ours-12D, Depth-Dependent Initialization	13.87	46.2

Table 2: Validation perplexity on FineWeb and average zero-shot accuracy on 10 common-sense reasoning benchmarks.

As can be seen, the choice of initialization is not the reason for our better performance. However, switching to the uniform variance initialization led to stronger performance on our approach.

C.2 Memory and Runtime Analyses

We compare Equations (2) and (12).

Runtime The first term on the R.H.S. of (12) requires computing an additional $\sqrt{d}\tanh\left(\frac{\cdot}{\sqrt{d}}\right)$ function. This is cheap because it is an element-wise operation, which scale by $O(N)$ compared to $O(N^3)$ of `matmul` operations that dominate the transformer block. The second term on the R.H.S. does not require any additional tensor operations since we are effectively replacing the $+\mathbf{x}_t$ term with $-\mathbf{x}_{t-1}$.

Memory The additional element-wise operation does not incur any additional memory cost. Using the $-\mathbf{x}_{t-1}$ layer does not require additional memory at training time because all layers' activations are already stored in memory in order to compute the backward pass. At inference time we need to store an extra layer's worth of activations, however, this cost is trivial compared to the memory that the KV cache is already using.

Put together, our method incurs minimal runtime and memory overhead. We prove this concretely by doing a FLOP analysis on the 12-layer transformer using the language modeling setting in Section 5.2:

Model	**FLOPs per optimizer step**	**% of Standard**
Standard-12D	11.635T	100
Ours-12D	11.663T	**100.24**

Table 3: FLOPs per optimizer step on the 12-layer model, expressed both as a raw count as well as a percentage compared to the standard residual.

As Table 3 shows, replacing the standard residual with Equation 12 represents only a 0.24% increase in FLOP count.