A. Related Work

Activation and Gradient Variance Much previous work that has addressed the training stability of deep neural networks has focused on controlling the magnitude of intermediate computations in the forward and backward pass—in other words, the activation and gradient variance. The problem of controlling gradient variance is also called the "exploding/vanishing gradient problem" and has been studied in many contexts.

Theoretically, several works have characterized information propagation for various settings. Poole et al. (2016); Raghu et al. (2017); Schoenholz et al. (2016) use various notions of expressivity to characterize the dynamics of deep, wide, randomly initialized feedforward networks with various activation functions, as a function of initialization statistics and other hyperparameters.

Pennington et al. (2017; 2018) focus more specifically on a notion of "dynamical isometry", which characterizes a worst-case setting of requiring information propagation for any input—essentially requiring the model as a whole to be well-conditioned, i.e. have singular values close to 1.

This mean field theory analysis has been extended to randomly initialized networks of many different architectures (including CNNs, RNNs, residual networks, etc.), characterizing when they exhibit stable forward and backward dynamics and providing recommendations for initializations (Yang & Schoenholz, 2017; Chen et al., 2018a; Xiao et al., 2018; Yang et al., 2019).

A parallel line of work (Hanin, 2018; Hanin & Rolnick, 2018) treat the case of deep ReLU nets (either feedforward or residual) in more depth and precisely analyze the finite-width setting, characterizing the initialization and network widths necessary to avoid the exploding/vanishing gradient problem.

Empirical Solutions in Deep Neural Networks Empirically, these theoretical insights have been realized in several ways. First, many initialization schemes have been proposed to control activation and gradient variances at initialization. These include simple "local" heuristics such as Xavier and He initialization (Glorot & Bengio, 2010; He et al., 2015b) for single weight matrices in isolation, as well as a range of more sophisticated "global" recommendations for an entire network such as those recommended above (Hanin & Rolnick, 2018; Xiao et al., 2018) and more (Zhang et al., 2019).

Second, imposing constraints for worst-case propagation, namely orthogonality on the weight matrices (even beyond initialization), has been explored extensively in both the RNN setting (Arjovsky et al., 2016) as well as for deep networks, particularly in settings sensitive to instability such as GANs (Miyato et al., 2018).

Finally, many architectural solutions have been proposed such as the classical highway and residual connections (Srivastava et al., 2015; He et al., 2015a; 2016). Architectural solutions to the exploding/vanishing gradient problem have been extensively studied in the RNN setting due to very long potential computation graphs, and some of the most popular solutions (Hochreiter & Schmidhuber, 1997; Chung et al., 2014) have been adopted for the deep network case, particularly for very unstable models such as Transformers (Parisotto et al., 2020; Xu et al., 2020). Our proposed architecture for Transformers is closely related to the DenseNet convolutional network (Huang et al., 2017), which is usually viewed as a distinct architecture involving extra skip connections, but can also be seen as an instance of a general architecture framework (Section 2.1) using concatenation operations. Although initially proposed as a heuristic to induce "strengthen feature propagation and encourage feature reuse", our insights shed additional light on why these models may have been so effective.

Amplification effect A drawback of the previous approaches is that they analyze statistics locally per layer, instead of the output of the entire model as a whole. Liu et al. (2020) consider the question of how a model's output will change after small parameter perturbations, which they call an "amplification effect". They introduce the following quantity:

$$\mathrm{Var}\left[\mathcal{F}(x_0, W) - \mathcal{F}(x_0, W^*)\right] \tag{6}$$

where $\mathcal{F}$ is a model depending on an input x_0 and parameter W, and W^* is a perturbed parameter. They argue that this quantity scales differently in depth for a pre-norm vs. post-norm placement, which explains the stability of pre-norm transformers over post-norm transformers.

This notion of amplification (Liu et al., 2020) can be seen as an informal and special case of our sensitivity framework. In this sense, sensitivity can be viewed as an operator $\mathbb{S}$ taking in a function (e.g. a network $\mathcal{F}$) and distribution over its inputs (e.g. parameters W), and returning a scalar that measures the amount of "amplification" the network has.

We contrast the technical differences in the definitions in more detail below. Equation (6) is not formally defined in several ways: First, the Var function is not defined, and it is not obvious what it means on a vector input since (6) is supposed to describe a scalar. Second, the source of randomness is not clear – even if W is randomly initialized, what is the distribution for which results hold? Finally and most importantly, what is the distribution of the noise $W^* - W$ and how is it normalized appropriately (otherwise, Eq. (6) depends on the noise magnitude)?

Sensitivity (Definition 1) formalizes these issues. First, the variance term more accurately measures the variance of scalar activations, or the expected squared length of the whole activation vector (normalized appropriately). Second, sensitivity must be a function of the initialization distribution as well as the function, as in Definition 1. Making this dependence explicit allows statements such as Theorem 2 which depends on the initialization distribution. The formal derivations of even the special cases considered in Liu et al. (2020) also require appropriate assumptions on the initialization distribution (Appendix C).

Finally, Definition 1 says that the distribution for the perturbation *must be proportional to the initialization distribution*. We remark that the obvious alternative, to have the perturbation be i.i.d. (e.g., $\mathcal{N}(0,1)$ for each parameter), has several drawbacks and our main results (Propositions 2 to 4 and Theorem 1) would not hold. For example, it can be shown from a modification of Proposition 5 that Eq. (6) simply reduces to $\|\nabla_W \mathcal{F}\|^2$, so that "amplification" is essentially just the gradient norm. Additionally, this alternative loses the desired composition and invariance properties. As two examples: (i) for a feedforward linear network, the sensitivity is simply the depth, but equation (6) is the sum of hidden layer sizes, which depends on architecture details (ii) the modified sensitivity depends heavily on the exact initialization distribution (since the perturbation is absolutely instead of relative to the distribution), and for example loses the invariance property in Proposition 3.

B. Sensitivity Proofs

This section proves all results from Section 3 of the main body. For completeness, we restate and prove all results here.

Definition 2 (Sensitivity)**.** Define $\mathbb{V}_p[f(\theta)]$ to be the second moment $\frac{1}{d}\mathbb{E}_{\theta\sim p(\theta)}[\|f(\theta)\|_2^2]$ normalized by the dimension d of $f(\theta)$.

Let $f(u,v)$ be a d-dimensional function of some (vector) inputs and $p(u,v)$ be a distribution over these inputs such that each (scalar) input is independent.

The *sensitivity* of f with respect to u and p, denoted $\mathbb{S}_{u,p}[f]$, is

$$\frac{1}{d}\frac{1}{\mathbb{V}_p[f(u,v)]}\lim_{\delta\to 0}\frac{1}{\delta^2}\mathbb{E}_{\substack{(u,v)\sim p(u,v)\\ \tilde{u}\sim p(u)}}\left[\|f(u+\delta\tilde{u},v)-f(u,v)\|_2^2\right]$$

Proposition 5 (Gradient characterization)**.**

$$\mathbb{S}_{u,p}[f(u,v)] = \mathbb{V}_p[f(u,v)]^{-1}\sum_{u_k}\mathbb{V}_p(u_k)\mathbb{V}_p(\nabla_{u_k} f) \tag{7}$$

where the sum is over individual (scalar) parameters $\theta_k \in \theta$.

Proof. Let $f(u,v), u, v$ have dimensions m, n, p respectively. We ignore the factor of $\frac{1}{m}$ appearing in Definition 2, since they cancel in equation (7).

We begin by writing out the expression from the definition of sensitivity:

$$\mathbb{S}_{u,p}[f(u,v)]\cdot\mathbb{V}_p[f(u,v)] = \lim_{\delta\to 0}\frac{1}{\delta^2}\mathbb{E}_{\substack{(u,v)\sim p(u,v)\\ \tilde{u}\sim p(u)}}\left[\|f(u+\delta\tilde{u},v)-f(u,v)\|_2^2\right]$$

Moving the limit inside, the term inside the expectation becomes a directional derivative

$$\begin{aligned}
&= \mathbb{E}_{\substack{(u,v)\sim p(u,v)\\ \tilde{u}\sim p(u)}} = \mathbb{E}_{u,v,\tilde{u}}\left[\lim_{\delta\to 0}\left\|\frac{f(u+\delta\tilde{u},v)-f(u,v)}{\delta}\right\|_2^2\right]\\
&= \mathbb{E}_{u,v,\tilde{u}}\,\|(\nabla_u f)\cdot\tilde{u}\|^2
\end{aligned}$$

Here we denote $\nabla_u f = \frac{\partial f}{\partial u}\big|_{u,v}$ to be the Jacobian (dimension $m \times n$) of the partial derivative of f with respect to its first argument, evaluated at u, v. Simplfying further,

$$
\begin{aligned}
=&\mathbb{E}_{u,v,\tilde{u}}\left[\tilde{u}^T(\nabla_u f)^T(\nabla_u f)\tilde{u}\right] \\
=&\mathbb{E}_{u,v,\tilde{u}}\mathbf{tr}\left[\tilde{u}^T(\nabla_u f)^T(\nabla_u f)\tilde{u}\right] \\
=&\mathbb{E}_{u,v,\tilde{u}}\mathbf{tr}\left[(\nabla_u f)^T(\nabla_u f)\tilde{u}\tilde{u}^T\right] \\
=&\mathbf{tr}\mathbb{E}_{u,v,\tilde{u}}\left[(\nabla_u f)^T(\nabla_u f)\tilde{u}\tilde{u}^T\right] \\
=&\mathbf{tr}\mathbb{E}_{u,v}\left[(\nabla_u f)^T(\nabla_u f)\right]\mathbb{E}_{\tilde{u}}\left[\tilde{u}\tilde{u}^T\right]
\end{aligned}
$$

Because $\tilde{u}$ is drawn from the same distribution as u and by the assumption that the distribution is independent over every parameter, $\mathbb{E}_{\tilde{u}}\left[\tilde{u}\tilde{u}^T\right]$ is diagonal, and this simplifies to the sum

$$
\begin{aligned}
&\sum_{u_k \in u} \mathbb{E}_u[u_k^2]\mathbb{E}_{u,v}\left[\|\nabla_{u_k} f\|^2\right] \\
=&\sum_{u_k \in u} \mathbb{V}[u_k]\mathbb{V}\left[\nabla_{u_k} f\right].
\end{aligned}
$$

To sanity check, here u_k has dimension 1 and $\nabla_{u_k} f$ has dimension m.

□

The above proof used the following more general trick:

Lemma 1. *If f and g are independent w.r.t. $p(\cdot)$, then $\mathbb{V}[f^T g] = \mathbf{tr}\left(\mathbb{E}[ff^T]\mathbb{E}[gg^T]\right)$.*

Proof.

$$
\begin{aligned}
\mathbb{V}[f^T g] &= \mathbb{E}\|f^T g\|_F^2 \\
&= \mathbb{E}\mathbf{tr}\left(f^T g g^T f\right) \\
&= \mathbb{E}\mathbf{tr}\left(f f^T g g^T\right) \\
&= \mathbf{tr}\left(\mathbb{E}[ff^T]\mathbb{E}[gg^T]\right)
\end{aligned}
$$

□

Proposition 6 (Composition rules)**.** *Sensitivity satisfies several local composition rules, including:*

- ***Identity*** $\mathbb{S}[u] = 1$.
- ***Sum*** *Given $u \sim p$, suppose $\nabla f(u)$ and $\nabla g(u)$ are uncorrelated. Then $\mathbb{S}[f(u) + g(u)] = \mathbb{S}f\frac{\mathbb{V}f}{\mathbb{V}[f+g]} + \mathbb{S}g\frac{\mathbb{V}g}{\mathbb{V}[f+g]}$.*
- ***Product*** *For disjoint sets of parameters u, v, if $f(u)$ is uncorrelated and constant variance given $\mathbb{S}[f(u)g(v)] = \mathbb{S}[f(u)] + \mathbb{S}[g(v)]$.*
- ***Chain rule*** $\mathbb{S}[f \circ g] = \mathbb{S}[f]\mathbb{S}[g]$.

Proof. We prove each composition rule in turn.

- **Identity** First, note that $\mathbb{V}_p[u] = \mathbb{E}\|u\|_2^2 = \sum_{u_k} \mathbb{V}[u_k]$. Second, note that for any u_k, $\nabla_{u_k} u$ is a basis vector (in particular, has norm 1). By Proposition 5,

$$
\begin{aligned}
\mathbb{V}_p[u] \cdot \mathbb{S}_{u,p}[u] &= \sum_{u_k} \mathbb{V}[u_k] \cdot \mathbb{V}[\nabla_{u_k} u] \\
&= \sum_{u_k} \mathbb{V}[u_k] \\
&= \mathbb{V}_p[u].
\end{aligned}
$$

- **Sum** By Proposition 1 and the assumption that $\nabla f, \nabla g$ are uncorrelated,

$$\begin{aligned}\mathbb{V}_p[f(u,v)+g(u,v)]\cdot\mathbb{S}_{u,p}[f(u,v)+g(u,v)] &= \sum_{u_k}\mathbb{V}[u_k]\mathbb{V}[\nabla_{u_k}f+\nabla_{u_k}g] \\ &= \sum_{u_k}\mathbb{V}[u_k]\mathbb{V}[\nabla_{u_k}f+\sum_{u_k}\mathbb{V}[u_k]\nabla_{u_k}g] \\ &= \mathbb{V}_p[f(u)]\cdot\mathbb{S}_{u,p}[f(u,v)]+\mathbb{V}_p[g(u)]\cdot\mathbb{S}_{u,p}[g(u,v)]\end{aligned}$$

- **Product** We simplify using Proposition 5.

$$\mathbb{V}[fg]\cdot\mathbb{S}[fg]=\sum_{u_k}\mathbb{V}[u_k]\cdot\mathbb{E}[\nabla_{u_k}(fg)]+\sum_{v_k}\mathbb{V}[v_k]\cdot\mathbb{E}[\nabla_{v_k}(fg)] \tag{8}$$

Let us consider just the first term for now. This can be simplified since g is a not a function of u. By the assumption that g is uncorrelated and constant variance, $\mathbb{E}[gg^T]=\sigma^2 I$ for some σ.

$$\begin{aligned}\sum_{u_k}\mathbb{V}[u_k]\cdot\mathbb{E}[\nabla_{u_k}(fg)] &= \sum_{u_k}\mathbb{V}[u_k]\cdot\mathbb{V}[(\nabla_{u_k}f)g] \\ &= \sum_{u_k}\mathbb{V}[u_k]\cdot\mathbf{tr}\left(\mathbb{E}[(\nabla_{u_k}f)^T(\nabla_{u_k}f)]\cdot\mathbb{E}[gg^T]\right) \\ &= \sum_{u_k}\mathbb{V}[u_k]\cdot\mathbf{tr}\left(\mathbb{E}[(\nabla_{u_k}f)^T(\nabla_{u_k}f)]\cdot\sigma^2 I\right) \\ &= \sum_{u_k}\mathbb{V}[u_k]\cdot\sigma^2\mathbf{tr}\left(\mathbb{E}[(\nabla_{u_k}f)^T(\nabla_{u_k}f)]\right) \\ &= \sum_{u_k}\mathbb{V}[u_k]\cdot\sigma^2\mathbb{V}[\nabla_{u_k}f]\end{aligned}$$

Also by the previous Lemma,

$$\begin{aligned}\mathbb{V}[fg] &= \mathbf{tr}\left(\mathbb{E}[f^T f]\mathbb{E}[gg^T]\right) \\ &= \sigma^2\mathbf{tr}\left(\mathbb{E}[f^T f]\right) \\ &= \sigma^2\mathbb{V}[f].\end{aligned}$$

Therefore

$$\begin{aligned}\frac{\sum_{u_k}\mathbb{V}[u_k]\cdot\mathbb{E}[\nabla_{u_k}(fg)]}{\mathbb{V}[fg]} &= \frac{\sum_{u_k}\mathbb{V}[u_k]\cdot\sigma^2\mathbb{V}[\nabla_{u_k}f]}{\sigma^2\mathbb{V}[f]} \\ &= \frac{\sigma^2\mathbb{V}[f]\mathbb{S}[f]}{\sigma^2\mathbb{V}[f]} \\ &= \mathbb{S}[f].\end{aligned}$$

Note that this is the first term of (8) after dividing both sides by $\mathbb{V}[fg]$. An analogous calculation can be made for the second term by symmetry. Adding these terms up, we have

$$\mathbb{S}[fg]=\mathbb{S}[f]+\mathbb{S}[g]$$

as desired.

- **Chain rule** We show a simplified case for clarity. Write Proposition 1 in vectorized form as $\mathbb{V}[f]\cdot\mathbb{S}[f]=\mathbb{V}[u]\mathbb{V}[\nabla_u f]$. Applying this to $f\circ g$,

$$\begin{aligned}\mathbb{V}[f(g(u))]\cdot\mathbb{S}[f(g(u))] &= \mathbb{V}[u]\cdot\mathbb{V}[\nabla_u(f(g(u)))] \\ &= \mathbb{V}[u]\cdot\mathbb{V}(\nabla f)(g(u))\cdot(\nabla g)(u)\end{aligned}$$

$$= \frac{\mathbb{V}[g(u)]\mathbb{V}(\nabla f)(g(u)) \cdot \mathbb{V}[u](\nabla g)(u)}{\mathbb{V}[g(u)]}$$
$$= \frac{\mathbb{S}[f] \cdot \mathbb{V}[f]\mathbb{S}[g]\mathbb{V}[g]}{\mathbb{V}[g]}$$

Dividing through gives $\mathbb{S}[f \circ g] = \mathbb{S}[f] \cdot \mathbb{S}[g]$.

□

Proposition 7 (Invariance rules)**.** *Sensitivity is invariant to the following transformations:*

- ***Normalization*** *If* $y = \mathsf{norm}(x)$ *then* $\mathbb{S}y = \mathbb{S}x$.
- ***Reparameterization*** *Consider the function* $g(\theta') = f(c\theta')$ *with distribution* $q(\theta') = p(\frac{1}{c}\theta')$. *Then* $\mathbb{S}_{\theta',q}g(\theta') = \mathbb{S}_{\theta,p}f(\theta)$.

Proof. The invariance rules follow straightforwardly from the definition and alternate characterization.

- **Normalization** Note that norm is dividing by a constant, which scales the left and right sides of Definition 2 equally through the variance terms, so does not change the sensitivity.
- **Reparameterization** We use Proposition 5, and show that each individual term in the sum is invariant to reparameterization. Let $x = cy$, define $g(y) = f(x) = f(cy)$ so these functions are equal. Then $f'(x) = f'(cy) = \frac{1}{c}cf'(cy) = \frac{1}{c}g'(y)$ and $x^2(\nabla f(x))^2 = y^2(\nabla g(y))^2$, as desired.

□

Proposition 8 (Layer decomposition)**.** *Suppose that* $x \oplus y = \alpha_i x + \beta_i y$ *is any weighted residual function where* α_i, β_i *are independent of* x, y *(but can depend on depth* i*). Also suppose that the module* $y_i = F_i(\hat{x}_i, \theta_i)$ *satisfies* $\mathbb{S}_{\hat{x}_i}F_i = 1$. *Then*

$$\mathbb{S}[\hat{x}_i] = \sum_{j \leq i} \rho_j, \qquad \rho_i = \frac{\beta_i^2 \mathbb{V}[y_i]}{\mathbb{V}[x_i]}\mathbb{S}[F_i] \tag{9}$$

Proof. By the definition of $\hat{x}_i$ and normalization, $\hat{x}_i = \frac{x_i}{\sqrt{\mathbb{V}[x_i]}}$. By the assumption about the combination $\oplus$, we can write

$$\hat{x}_i = \gamma \hat{x}_{i-1} + \frac{\beta_i}{\sqrt{\mathbb{V}[x_i]}} y_i \tag{10}$$

for some γ.

Take the variance of both sides of (10) to get

$$\mathbb{V}[\hat{x}_i] = \gamma^2 \mathbb{V}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]$$
$$1 = \gamma^2 + \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i].$$

Take the sensitivity of both sides of (10) to get

$$\mathbb{S}[\hat{x}_i] = \frac{\gamma^2 \mathbb{V}[\hat{x}_{i-1}]}{\mathbb{V}[\hat{x}_i]}\mathbb{S}[\hat{x}_{i-1}] + \frac{\frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]}{\mathbb{V}[\hat{x}_i]}\mathbb{S}[y_i]$$
$$= \gamma^2 \mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]\mathbb{S}[y_i]$$

$$= \left(1 - \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]\right)\mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]\mathbb{S}[y_i]$$
$$= \mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]}\mathbb{V}[y_i]\left(\mathbb{S}[y_i] - \mathbb{S}[\hat{x}_{i-1}]\right)$$

Finally, by the chain rule for sensitivities we have

$$\mathbb{S}_{\theta_i,\theta_{<i}}[y_i] = \mathbb{S}_{\theta_i,\theta_{<i}}[F_i(\hat{x}_i,\theta_i)] = (\mathbb{S}_{\hat{x}_i}F_i)\cdot\mathbb{S}_{\theta_{<i}}\hat{x}_i + (\mathbb{S}_{\theta_i}F_i)\cdot\mathbb{S}_{\theta_i}\theta_i.$$

By the assumption that $\mathbb{S}_{\hat{x}_i}F_i = 1$, we have $\mathbb{S}[y_i] - \mathbb{S}[\hat{x}_{i-1}] = \mathbb{S}F_i$. Therefore

$$\mathbb{S}[\hat{x}_i] = \mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2\mathbb{V}[y_i]}{\mathbb{V}[x_i]}\mathbb{S}[F_i],$$

which immediately gives the result. □

Theorem 2. *For any architecture that preserves activation variance through the depth of the network, the gradient variance at depth i is proportional to $\mathbb{S}x_i - \mathbb{S}x_{i-1}$.*

Proof. We start from Proposition 5 and decompose the sum into parameters u_k before layer i and v_k in layer k.

$$\mathbb{S}[x_i] = \sum_{u_k}\mathbb{V}[u_k]\frac{\mathbb{V}[\nabla_{u_k}x_i]}{\mathbb{V}[x_i]} + \sum_{v_k}\mathbb{V}[v_k]\frac{\mathbb{V}[\nabla_{v_k}x_i]}{\mathbb{V}[x_i]}.$$

Let us examine the first term. We can expand the fraction as

$$\frac{\mathbb{V}[\nabla_{u_k}x_i]}{\mathbb{V}[x_i]} = \frac{\mathbb{V}[\nabla_{u_k}x_{i-1}\nabla_{x_{i-1}}x_i]}{\mathbb{V}[x_i]}$$
$$= \frac{\mathbb{V}[\nabla_{u_k}x_{i-1}]}{\mathbb{V}[x_{i-1}]}\frac{\mathbb{V}[x_{i-1}]}{\mathbb{V}[x_i]}\mathbb{V}[\nabla_{x_{i-1}}x_i]$$
$$= \frac{\mathbb{V}[\nabla_{u_k}x_{i-1}]}{\mathbb{V}[x_{i-1}]}\mathbb{S}_{x_{i-1}}[x_i].$$

Therefore

$$\mathbb{S}[x_i] = \mathbb{S}_{x_{i-1}}[x_i]\sum_{u_k}\mathbb{V}[u_k]\frac{\mathbb{V}[\nabla_{u_k}x_{i-1}]}{\mathbb{V}[x_{i-1}]} + \sum_{v_k}\mathbb{V}[v_k]\frac{\mathbb{V}[\nabla_{v_k}x_i]}{\mathbb{V}[x_i]}$$
$$= \mathbb{S}_{x_{i-1}}[x_i]\mathbb{S}[x_{i-1}] + \sum_{v_k}\mathbb{V}[v_k]\frac{\mathbb{V}[\nabla_{v_k}x_i]}{\mathbb{V}[x_i]}.$$

or WLOG let $\mathbb{S}_{x_{i-1}}[x_i] = 1$ (e.g., true for a linear or more generally homogenous layer F_i). We can also consider the normalized $\hat{x}_i$. This yields

$$\mathbb{S}[x_i] - \mathbb{S}[x_{i-1}] = \sum_{v_k}\mathbb{V}[v_k]\mathbb{V}[\nabla_{v_k}\hat{x}_i].$$

Finally, note that for an activation variance-preserving network, the variance $\mathbb{V}[v_k]$ is inversely proportional to the fan-in. In other words, it is inversely proportional to the number of such parameters. Thus the $\sum_{v_k}$ and $\mathbb{V}[v_k]$ cancel, and we are left with

$$\mathbb{V}[\nabla_{v_k}\hat{x}_i] \propto \mathbb{S}[x_i] - \mathbb{S}[x_{i-1}].$$

□

C. Sensitivity Derivations

For most of these derivations, we will apply Proposition 8. We make the following assumptions without loss of generality about the modules:

- the modules satisfy $\mathbb{S}[F_i] = 1$.
- the modules are activation variance preserving, i.e. $\mathbb{V}[y_i] = \mathbb{V}[\hat{x}_{i-1}]$ where $y_i = F_i(\hat{x}_{i-1})$ (equation (2)).
- the modules decorrelate the outputs from inputs, i.e. $\mathbb{E}[\hat{x}_{i-1} \circ y_i] = 0$ (where $\circ$ denotes elementwise product).

are normalized appropriately. For example, these assumptions are satisfied by a linear module. Finally, WLOG also assume the inputs to the network x_0 are normalized so that $\mathbb{V}[x_0] = 1$.

Note that the second assumption immediately implies that $\mathbb{V}[y_i] = 1$. By Proposition 8, to calculate the sensitivities it suffices to calculate

$$\rho_i = \frac{\beta_i^2 \mathbb{V}[y_i]}{\mathbb{V}[x_i]}.$$

Therefore the main quantity to track will be the variances of the activations.

C.1. Feedforward

By the assumption that modules are variance preserving, we have $\mathbb{V}[y_i] = \mathbb{V}[\hat{x}_{i-1}] = 1$. Since $x \oplus y = y$, we have $x_i = y_i$ so $\mathbb{V}[x_i] = 1$. Therefore $\rho_i = 1$ and $\mathbb{S}[\hat{x}_i] = N$.

C.2. Residual

Pre-norm This was analyzed in Section 3.4. The main point is that $\mathbb{V}[x_i] = i + 1$ so $\rho_i = \frac{1}{i+1}$.

Post-norm The residual connection in this architecture is $x_i = \hat{x}_{i-1} + y_i$. By the decorrelating assumption, $\mathbb{V}[x_i] = 2 = 2\mathbb{V}[y_i]$. Finally $\beta_i = 1$, so we get $\rho_i = \frac{1}{2}$ and $\mathbb{S}[\hat{x}_i] = \frac{N}{2}$.

No-norm This network is not technically defined in Eqs. (1) to (3), but we analyze it for completeness. This is instead defined by

$$\begin{aligned} y_i &= F_i(x_{i-1}) \\ x_i &= x_{i-1} + y_i \end{aligned}$$

(which can be viewed as Eqs. (1) to (3), but replacing norm with a no-op). Since F_i is variance preserving, we have $\mathbb{V}y_i = \mathbb{V}x_{i-1}$.

By the decorrelating assumption, $\mathbb{V}[x_i] = 2\mathbb{V}[y_i]$. Finally $\beta_i = 1$, so we get $\rho_i = \frac{1}{2}$ and $\mathbb{S}[\hat{x}_i] = \frac{N}{2}$.

Note that the main difference between the post-norm residual and no-norm residual is that in the former, the variance of x_i is controlled, while in the latter, the variance doubles every layer. However, their sensitivities are the same.

C.3. Weighted Residual

We consider a more general weighted residual of the form $x \oplus y = \alpha x + \beta y$ for constants α, β.

Pre-norm We have

$$\mathbb{V}[x_i] = \alpha^2 \mathbb{V}[x_{i-1}] + \beta^2 \mathbb{V}[y_i]$$

so that $\mathbb{V}[x_i] \to \frac{\beta^2}{1-\alpha^2}$. Therefore $\rho_i \to (1 - \alpha^2)$ and $\mathbb{S}[\hat{x}_i] = (1 - \alpha^2) \cdot N$

Post-norm We have

$$\begin{aligned}\mathbb{V}[x_i] &= \alpha^2\mathbb{V}[\hat{x}_{i-1}] + \beta^2\mathbb{V}[y_i] \\ &= \alpha^2 + \beta^2.\end{aligned}$$

Therefore $\rho_i = \frac{\beta^2}{\alpha^2+\beta^2}$ and $\mathbb{S}[\hat{x}_i] = \frac{\beta^2}{\alpha^2+\beta^2} \cdot N$.

GTrXL The Gated TransformerXL (GTrXL) (Parisotto et al., 2020) replaces the residual by a GRU-style gate. This is similar to the weighted residual, but where β is a function of x, y. This makes it difficult to analyze (for example, the two parts of the sum are not decorrelated so the variance is difficult to track). However, we can get a sense by approximating this gate with a constant.

For example, Parisotto et al. (2020) increase a bias term inside the gate, which has the effect of setting $\mathbb{E}[\alpha] = \sigma(2)$ and $\mathbb{E}[\beta] = \sigma(-2)$ where σ is the sigmoid function. They also use the pre-norm placement. Thus we would expect the sensitivity to be about $(1 - \sigma(2)^2) \cdot N \approx 0.22N$.

C.4. RescaleNet

RescaleNet (Shao et al., 2020) defines $x \oplus y = \sqrt{\frac{i-1}{i}}x + \sqrt{\frac{1}{i}}y$. This was in fact chosen to be variance-preserving, so that

$$\mathbb{V}[x_i] = \frac{i-1}{i}\mathbb{V}[\hat{x}_{i-1}] + \frac{1}{i}\mathbb{V}[y_i] = 1.$$

Since $\beta_i = \sqrt{\frac{1}{i}}$, we have $\rho_i = \frac{1}{i}$ and $\mathbb{S}[\hat{x}_i] = H_N$.

C.5. AdMin

AdMin (Liu et al., 2020) defines $x \oplus y = w_i x + y$ where w_i is a learnable scalar initialized to $\sqrt{i}$. We have

$$\mathbb{V}[x_i] = i\mathbb{V}[\hat{x}_{i-1}] + \mathbb{V}[y_i] = i + 1$$

and $\beta_i = 1$, so $\rho_i = \frac{1}{i+1}$ and $\mathbb{S}[\hat{x}_i] = H_{N+1} - 1$.

C.6. Catformer

The Catformer does not use an additive combination function, so Proposition 8 does not apply. However, we can still use the composition rules.

We make two observations.

Proposition 9 (Concatenation rule)**.** *Let x, y have dimensions m, n respectively. Then*

$$\mathbb{S}[\mathsf{concat}[x,y]] = \mathbb{S}[f]\frac{\mathbb{V}[f]}{\mathbb{V}[f]+\mathbb{V}[g]} + \mathbb{S}[g]\frac{\mathbb{V}[g]}{\mathbb{V}[f]+\mathbb{V}[g]}$$

Proof. For shorthand let $z = \mathsf{concat}[x, y]$. Using Proposition 5, we have

$$\mathbb{S}[z] \cdot \mathbb{V}[z] = \sum_{u_k} \mathbb{V}(u_k)\mathbb{V}(\nabla_{u_k} z)$$

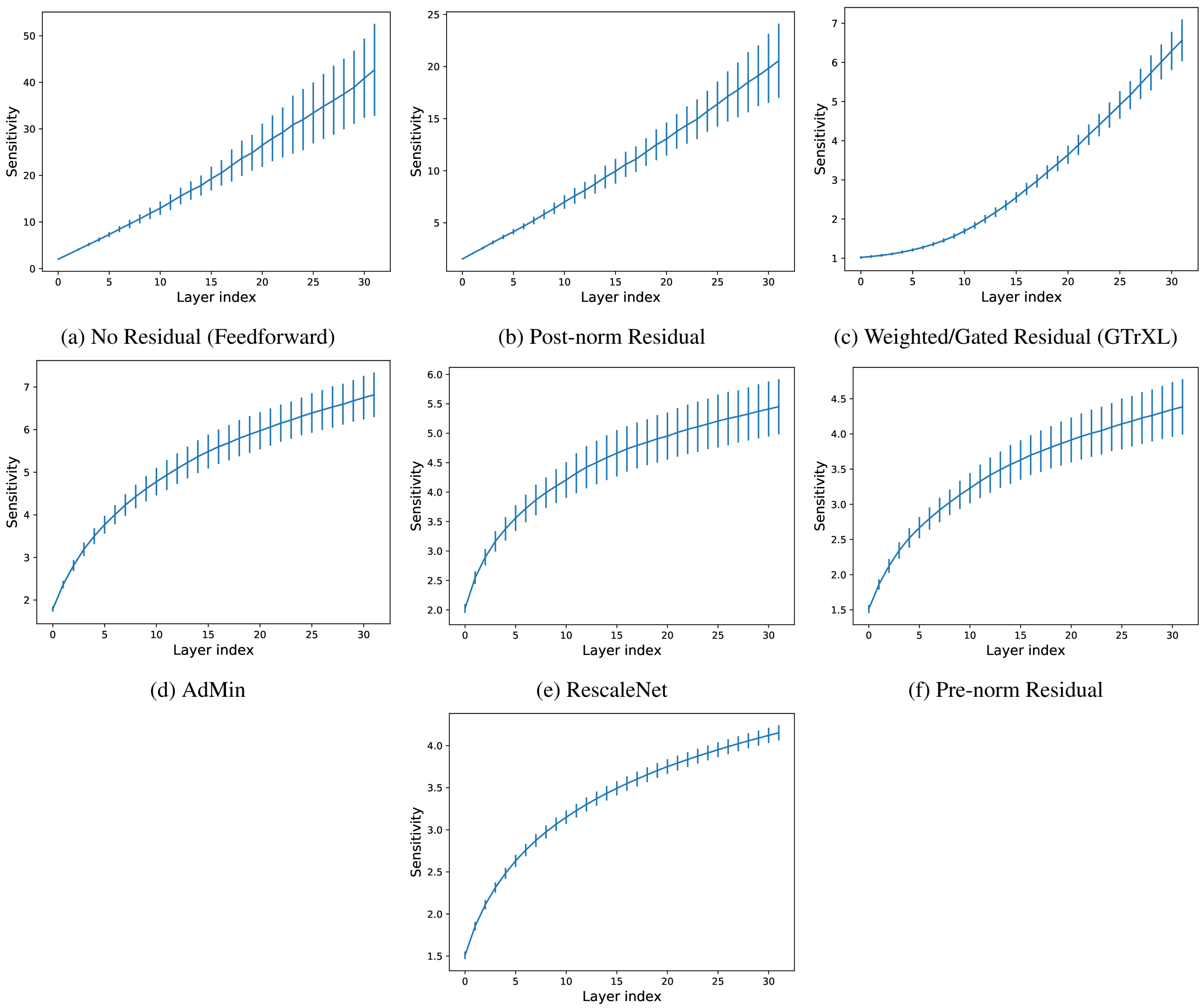
By definition of $\mathbb{V}$ (the squared norm of this vector), by definition of concatenation $\mathbb{V}_{u_k} z = \mathbb{V}_{u_k}[x] + \mathbb{V}_{u_k}[y]$. Similarly, $\mathbb{V}[z] = \mathbb{V}[x] + \mathbb{V}[y]$. Expanding and using Proposition 5 again,

$$\begin{aligned}\mathbb{S}[z] \cdot \mathbb{V}[z] &= \sum_{u_k} \mathbb{V}(u_k)\mathbb{V}(\nabla_{u_k} z) \\ &= \sum_{u_k} \mathbb{V}(u_k)\left[\mathbb{V}(\nabla_{u_k} x) + \mathbb{V}(\nabla_{u_k} y)\right] \\ &= \mathbb{S}[x] \cdot \mathbb{V}[x] + \mathbb{S}[y] \cdot \mathbb{V}[y].\end{aligned}$$

Dividing by $\mathbb{V}[z] = \mathbb{V}[x] + \mathbb{V}[y]$ gives the result. ∎

C.7. Empirical Confirmation

To confirm the theory, we consider several methods (Table 1) and calculate their sensitivities empirically. This is done by randomly initializing a network, picking a perturbation with a small δ and calculating via Definition 1. Results line up closely with the theoretical calculations, shown in Fig. 3.



(a) No Residual (Feedforward)

(b) Post-norm Residual

(c) Weighted/Gated Residual (GTrXL)

(d) AdMin

(e) RescaleNet

(f) Pre-norm Residual

(g) Concatenation (Catformer)

Figure 3. Empirical sensitivities: $\mathbb{S}[\hat{x}_i]$ vs. i for i up to $N = 32$. Error bars are over the randomness in the initialization and the perturbation.

D. Method Details

D.1. Architecture details: controlling parameters

In this section, we describe the method used to control the number of parameters of the Catformer.

We use the following terminology:

d : embedding dimension (output dimension of each module F_i)

e : expansion factor for FF inner layer relative to attention inner layer (n)

a : expansion factor for attn inner layer

D : total depth of network

i : current layer number (indexed from $0, \ldots, D-1$)

One "layer" of a Transformer block is described as a MHA module and then a FF module. Thus, layer i has initial dimension $d + 2id$ (after concatenation). The MHA module brings it to dimension $d + (2i+1)d$, and the FF module brings it to $d + 2(i+1)d$

- Attention layers: QKV each project dim $(d + 2id) \to ad$, final projection matrix brings $ad \to d$
  total parameters of attention in layer i: $ad^2(6i+4)$
- FF layers: initial projection $(d + (2i+1)d) \to ed$, non-linearity, second projection $ed \to d$
  total parameters of feedforward in layer i: $d^2(2i+3)e$
- Average layer depth $i = \frac{D-1}{2}$
- Average attention parameters per layer: $d^2 \cdot a(3D+1)$
- Average feedforward parameters per layer: $d^2 \cdot e(D+2)$
- Average total parameters per layer: $d^2 \cdot (3aD + eD + a + 2e)$

We use $a = 2, e = 4$ for all experiments in Sections 5.1 and 5.2. Note that a stanford Transformer block with a FF expansion factor of 4 (e.g. embedding dimension 512, inner dimension 2048 in the MLP) uses $12n^2$ parameters per layer, where n is the Transformer embedding dimension. For a given model size, the Catformer dimension d can be set appropriately to match this.

E. Experiment Details

E.1. Synthetic Language Modeling

Methods All baseline transformer variants used an embedding dimension of size 512 with an inner dimension in the MLP (equation (5)) of size 2048. The Catformer used approximately equal parameters for each depth using the method described in Appendix D with expansion factors $e_a = 2, e_f = 4$ (note that this expansion factor for the FF layer matches that of the baselines). This equates to dimensions $d = 320, 248, 208$ for depths $N = 2, 4, 6$ respectively.

Training Models were trained with the Adam optimizer (Kingma & Ba, 2015) with learning rate $8e-4$. Batch size 16 was used, and new data was randomly generated according to Section 5.1 for every minibatch. Models were trained for 20000 steps, and the numbers reported in Tables Tables 2 to 4 are the perplexity on fresh "validation" data of size 200. All experiments used 3 seeds for each model; the training data was tied to the seed so that each model was trained on the exact same data for every minibatch.

E.2. DMLab30

We train the transformer agents using a distributed reinforcement learning framework based on R2D2. For distributed training, we leverage 256 actors, a batch size of 64, a replay period of 40, and a min replay size of 5000 and max of 10000. Our transformer agents have torso MLP sizes of 256, 5 heads, and 2 transformer blocks. The memory size is 50 and the transformer value size for Catformer is 32 (scaled to match for TrXL and GTrXL). For each seed, we ran distributed training using 16 TPU v2 chips. We trained each model for 300M steps. We did not tune any of the hyper-parameters specific to R2D2, nor did we tune the dimensions of the transformer torso extensively. We did sweep over gradient norm clipping values, experimenting with values of 40, 100, and no clipping. We ran the same sweeps for all models to ensure a maximally fair comparison.

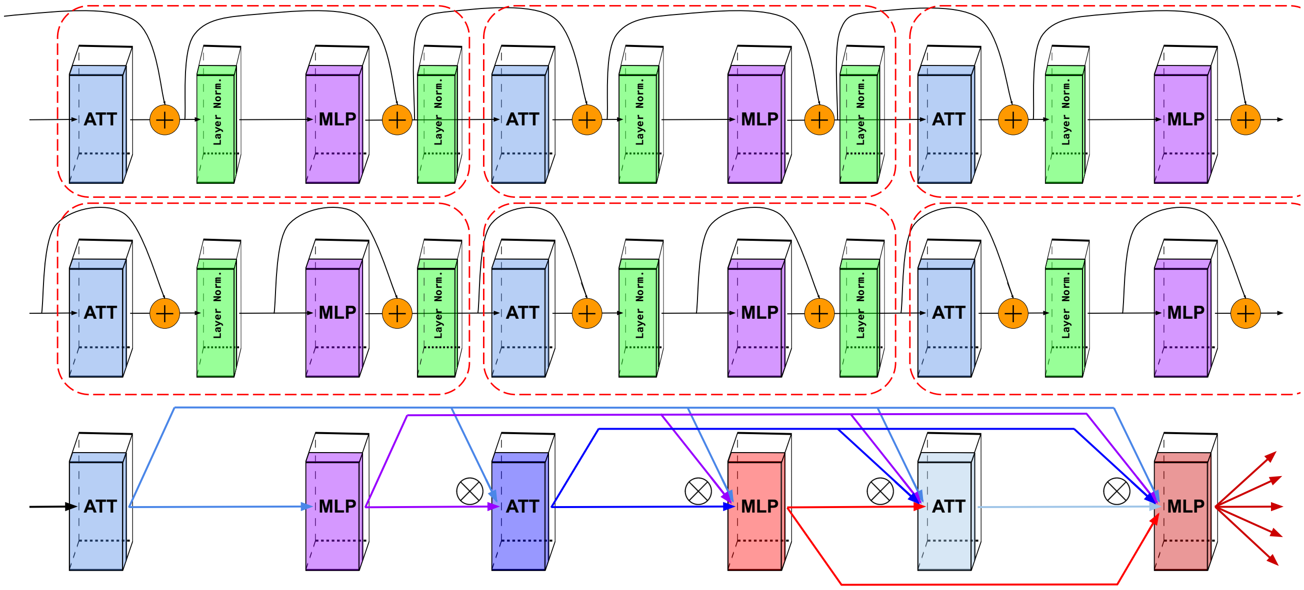


Figure 4. Illustration of the Catformer architecture connections (bottom) vs. other standard pre-norm and post-norm residual architectures.

References

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128. PMLR, 2016.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Chen, M., Pennington, J., and Schoenholz, S. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. In *International Conference on Machine Learning*, pp. 873–882. PMLR, 2018a.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G. F., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 76–86. Association for Computational Linguistics, 2018b. doi: 10.18653/v1/P18-1008. URL https://www.aclweb.org/anthology/P18-1008/.

Choromanski, K., Davis, J. Q., Likhosherstov, V., Song, X., Slotine, J. E., Varley, J., Lee, H., Weller, A., and Sindhwani, V. An ode to an ODE. *CoRR*, abs/2006.11421, 2020a. URL https://arxiv.org/abs/2006.11421.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with Performers. *CoRR*, arXiv:2009.14794, 2020b. URL https://arxiv.org/abs/2009.14794.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

Dao, T., Gu, A., Eichhorn, M., Rudra, A., and Ré, C. Learning fast algorithms for linear transforms using butterfly factorizations. In *International Conference on Machine Learning*, pp. 1517–1527. PMLR, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL `https://arxiv.org/abs/2010.11929`.

Glorot, X. and Bengio, Y. 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.

Hanin, B. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

Hanin, B. and Rolnick, D. How to start training: The effect of initialization and architecture. *arXiv preprint arXiv:1803.01719*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015a.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks, 2016.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Li, J. Universal transforming geometric network. *CoRR*, abs/1908.00723, 2019. URL `http://arxiv.org/abs/1908.00723`.

Likhosherstov, V., Davis, J., Choromanski, K., and Weller, A. CWY parametrization for scalable learning of orthogonal and stiefel matrices. *CoRR*, abs/2004.08675, 2020. URL `https://arxiv.org/abs/2004.08675`.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. Understanding the difficulty of training transformers, 2020.

Luo, H., Zhang, S., Lei, M., and Xie, L. Simplified self-attention for transformer-based end-to-end speech recognition. *CoRR*, abs/2005.10463, 2020. URL `https://arxiv.org/abs/2005.10463`.

Madani, A., McCann, B., Naik, N., Keskar, N. S., Anand, N., Eguchi, R. R., Huang, P., and Socher, R. Progen: Language modeling for protein generation. *CoRR*, abs/2004.03497, 2020. URL `https://arxiv.org/abs/2004.03497`.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pp. 7487–7498. PMLR, 2020.

Pennington, J., Schoenholz, S. S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *arXiv preprint arXiv:1711.04735*, 2017.

Pennington, J., Schoenholz, S., and Ganguli, S. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1924–1932. PMLR, 2018.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. *arXiv preprint arXiv:1606.05340*, 2016.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C., Ma, J., and Fergus, R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioArxiv*, 04 2019. doi: 10.1101/622803.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Shao, J., Hu, K., Wang, C., Xue, X., and Raj, B. Is normalization indispensable for training deep neural network? *Advances in Neural Information Processing Systems*, 33, 2020.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning, 2018.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pp. 5393–5402. PMLR, 2018.

Xu, H., Liu, Q., Xiong, D., and van Genabith, J. Transformer with depth-wise lstm, 2020.

Yang, G. and Schoenholz, S. S. Mean field residual networks: On the edge of chaos. *arXiv preprint arXiv:1712.08969*, 2017.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization, 2019.