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—i.e., 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:

\[
\text{Var} \left[ \mathcal{F}(x_0, W) - \mathcal{F}(x_0, W^*) \right]
\]

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 \( \mathcal{S} \) 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 begin by writing out the expression from the definition of sensitivity:

\[ S_{\theta}(\theta) = \frac{1}{d} \mathbb{E}_{\theta \sim p(\theta)} \left[ \left\| \nabla_{\theta} f(\theta) \right\|^2 \right] \]

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

**Proposition 5 (Gradient characterization).**

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

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

**Proof.** Let \( f(u,v) \) 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:

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

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

\[ = \mathbb{E}_{u,v,\hat{u}} \left[ \lim_{\delta \to 0} \left\| \frac{f(u + \delta \hat{u}, v) - f(u, v)}{\delta} \right\|^2 \right] \]

\[ = \mathbb{E}_{u,v,\hat{u}} \left\| (\nabla u f) \cdot \hat{u} \right\|^2 \]
Here we denote $\nabla_u f = \frac{\partial f}{\partial u} \bigg|_{u,v}$ to be the Jacobian (dimension $m \times n$) of the partial derivative of $f$ with respect to its first argument, evaluated at $u, v$. Simplifying further,

$$
\begin{align*}
= & \mathbb{E}_{u,v,\tilde{u}} [ \tilde{u}^T (\nabla_v f)^T (\nabla_u f) \tilde{u} ] \\
= & \mathbb{E}_{u,v,\tilde{u}} \text{tr} [ \tilde{u}^T (\nabla_v f)^T (\nabla_u f) \tilde{u} ] \\
= & \mathbb{E}_{u,v,\tilde{u}} \text{tr} [ (\nabla_v f)^T (\nabla_u f) \tilde{u} \tilde{u}^T ] \\
= & \text{tr} \mathbb{E}_{u,v} [ (\nabla_v f)^T (\nabla_u f) ] \mathbb{E}_{\tilde{u}} [ \tilde{u} \tilde{u}^T ]
\end{align*}
$$

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

$$
\sum_{u_k \in u} \mathbb{E}_u [u_k^2] \mathbb{E}_{u,v} [ ||\nabla_u f||^2 ] = \sum_{u_k \in u} \mathbb{V}[u_k] \mathbb{V}[\nabla_u f].
$$

To sanity check, here $u_k$ has dimension 1 and $\nabla_u f$ has dimension $m$.

\[ \square \]

The above proof used the following more general trick:

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

**Proof.**

$$
\begin{align*}
\mathbb{V}[f^T g] & = \mathbb{E}||f^T g||^2 \\
& = \mathbb{E} \text{tr} (f^T g g^T f) \\
& = \mathbb{E} \text{tr} (f f^T g g^T) \\
& = \text{tr} (\mathbb{E}[f f^T] \mathbb{E}[gg^T])
\end{align*}
$$

\[ \square \]

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

$$
\begin{align*}
\mathbb{V}_p[u] \cdot \mathbb{S}_{u,p}[u] & = \sum_{u_k} \mathbb{V}[u_k] \cdot \mathbb{V}[\nabla_{u_k} u] \\
& = \sum_{u_k} \mathbb{V}[u_k] \\
& = \mathbb{V}_p[u].
\end{align*}
$$
• **Sum** By Proposition 1 and the assumption that $\nabla f, \nabla g$ are uncorrelated,

$$
\mathbb{V}_p[f(u, v) + g(u, v)] \cdot S_{u, p}[f(u, v) + g(u, v)] = \sum_{u_k} \mathbb{V}[u_k] \mathbb{V}[\nabla u_k f + \nabla u_k g]
$$

$$
= \sum_{u_k} \mathbb{V}[u_k] \mathbb{V}[\nabla u_k f] + \sum_{u_k} \mathbb{V}[u_k] \mathbb{V}[\nabla u_k g]
$$

$$
= \mathbb{V}_p[f(u)] \cdot S_{u, p}[f(u, v)] + \mathbb{V}_p[g(u)] \cdot S_{u, p}[g(u, v)]
$$

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

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

(8)

Let us consider just the first term for now. This can be simplified since $g$ is uncorrelated and constant variance, $\mathbb{E}[gg^T] = \sigma^2 I$ for some $\sigma$.

$$
\sum_{u_k} \mathbb{V}[u_k] \cdot \mathbb{E}[\nabla u_k (fg)] = \sum_{u_k} \mathbb{V}[u_k] \cdot \mathbb{V}[\nabla u_k f)g]
$$

$$
= \sum_{u_k} \mathbb{V}[u_k] \cdot \text{tr} (\mathbb{E}[(\nabla u_k f)^T (\nabla u_k f)] \cdot \mathbb{E}[gg^T])
$$

$$
= \sum_{u_k} \mathbb{V}[u_k] \cdot \text{tr} (\mathbb{E}[(\nabla u_k f)^T (\nabla u_k f)] \cdot \sigma^2 I)
$$

$$
= \sum_{u_k} \mathbb{V}[u_k] \cdot \sigma^2 \text{tr} (\mathbb{E}[(\nabla u_k f)^T (\nabla u_k f)])
$$

$$
= \sum_{u_k} \mathbb{V}[u_k] \cdot \sigma^2 \mathbb{V}[\nabla u_k f]
$$

Also by the previous Lemma,

$$
\mathbb{V}[fg] = \text{tr} (\mathbb{E}[f^T g] \mathbb{E}[gg^T])
$$

$$
= \sigma^2 \text{tr} (\mathbb{E}[f^T f])
$$

$$
= \sigma^2 \mathbb{V}[f].
$$

Therefore

$$
\frac{\sum_{u_k} \mathbb{V}[u_k] \cdot \mathbb{E}[\nabla u_k (fg)]}{\mathbb{V}[fg]} = \frac{\sum_{u_k} \mathbb{V}[u_k] \cdot \sigma^2 \mathbb{V}[\nabla u_k f]}{\sigma^2 \mathbb{V}[f]}
$$

$$
= \frac{\sigma^2 \mathbb{V}[fg]}{\sigma^2 \mathbb{V}[f]}
$$

$$
= S[fg].
$$

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

$$
S[fg] = S[f] + S[g]
$$
as desired.

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

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

$$
= \mathbb{V}[u] \cdot \mathbb{V}(\nabla f)(g(u)) \cdot (\nabla g)(u)
$$
\[
\begin{align*}
\mathbb{V}[g(u)]\mathbb{V}(\nabla f)(g(u)) \cdot \mathbb{V}[u](\nabla g)(u) \\
\mathbb{V}[g(u)]
\end{align*}
\]

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

\[
\text{Proposition 7 (Invariance rules). Sensitivity is invariant to the following transformations:}
\]

- **Normalization** If \( y = \text{norm}(x) \) 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 \( f'(x) = f'(cy) = \frac{1}{c}f'(cy) = \frac{1}{c}g'(y) \) and \( x^2(\nabla f(x))^2 = y^2(\nabla g(y))^2 \), as desired.

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

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

(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

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

(10)

for some \( \gamma \).

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

\[
\mathbb{S}[\hat{x}_i] = \gamma^2 \mathbb{V}[\hat{x}_{i-1}] \mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]} \mathbb{V}[y_i] \mathbb{S}[y_i] \\
= \gamma^2 \mathbb{S}[\hat{x}_{i-1}] + \frac{\beta_i^2}{\mathbb{V}[x_i]} \mathbb{V}[y_i] \mathbb{S}[y_i]
\]
which immediately gives the result.

For any architecture that preserves activation variance through the depth of the network, the gradient variance at depth \( i \) is proportional to \( S_{x_i} - S_{x_{i-1}} \).

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

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

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

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

Therefore

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

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

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

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

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

\( \square \)
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 $S[F_i] = 1$.
- 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 $S[\hat{x}_i] = N$.

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 $S[\hat{x}_i] = \frac{N}{2}$.

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

$$y_i = F_i(x_{i-1}),$$
$$x_i = x_{i-1} + y_i$$

(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 $S[\hat{x}_i] = \frac{N}{2}$.

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 $\alpha, \beta$.

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 $S[\hat{x}_i] = (1 - \alpha^2) \cdot N$.
We make two observations. AdMin (Liu et al., 2020) defines \(x\) The Catformer does not use an additive combination function, so Proposition 8 does not apply. However, we can still use the \(\rho\) Therefore \(\rho_i = \frac{\beta^2}{\alpha^2 + \beta^2}\) and \(S[\hat{x}_i] = \frac{\beta^2}{\alpha^2 + \beta^2} \cdot N\).

**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 \(\beta\) 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 \(E[\alpha] = \sigma(2)\) and \(E[\beta] = \sigma(-2)\) where \(\sigma\) 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 \(x \odot y = \sqrt{i-1}x + \sqrt{i}y\). This was in fact chosen to be variance-preserving, so that

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

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

**C.5. AdMin**

AdMin (Liu et al., 2020) defines \(x \odot y = w_i x + y\) 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 \(S[\hat{x}_i] = H_{N+1} - 1\).

**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

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

**Proof.** For shorthand let \(z = \text{concat}[x, y]\). Using Proposition 5, we have

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

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

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

\[
= \sum_{u_k} \mathbb{V}(u_k) [\mathbb{V}(\nabla u_k x) + \mathbb{V}(\nabla u_k y)]
\]

\[
= S[x] \cdot \mathbb{V}[x] + S[y] \cdot \mathbb{V}[y].
\]

Dividing by \(\mathbb{V}[z] = \mathbb{V}[x] + \mathbb{V}[y]\) gives the result. \(\Box\)
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 $\delta$ and calculating via Definition 1. Results line up closely with the theoretical calculations, shown in Fig. 3.

![Graphs showing empirical sensitivities for different methods](image)

Figure 3. Empirical sensitivities: $S[\hat{x}_i]$ 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)
Catformer: Designing Stable Transformers via Sensitivity Analysis

**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 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


Catformer: Designing Stable Transformers via Sensitivity Analysis


