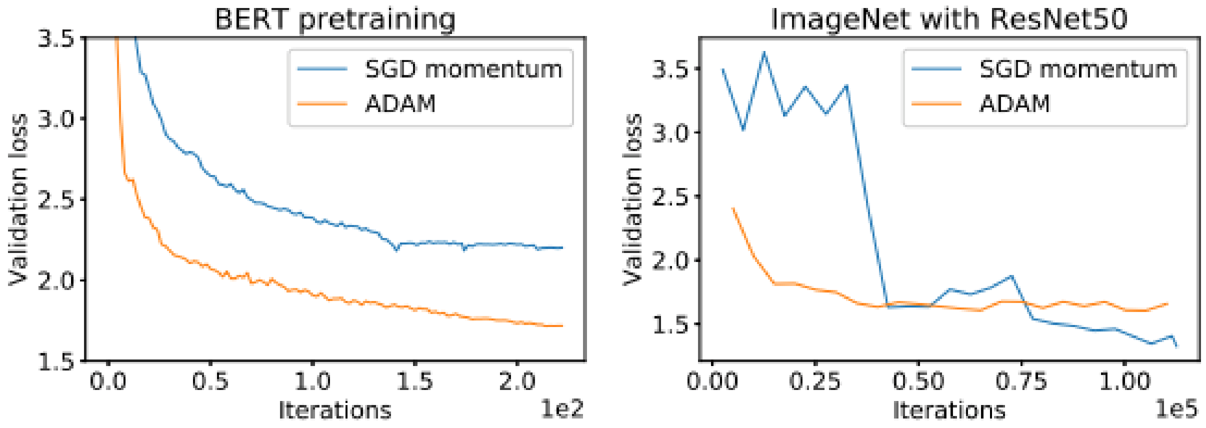
Understanding the Difficulty of Training Transformers

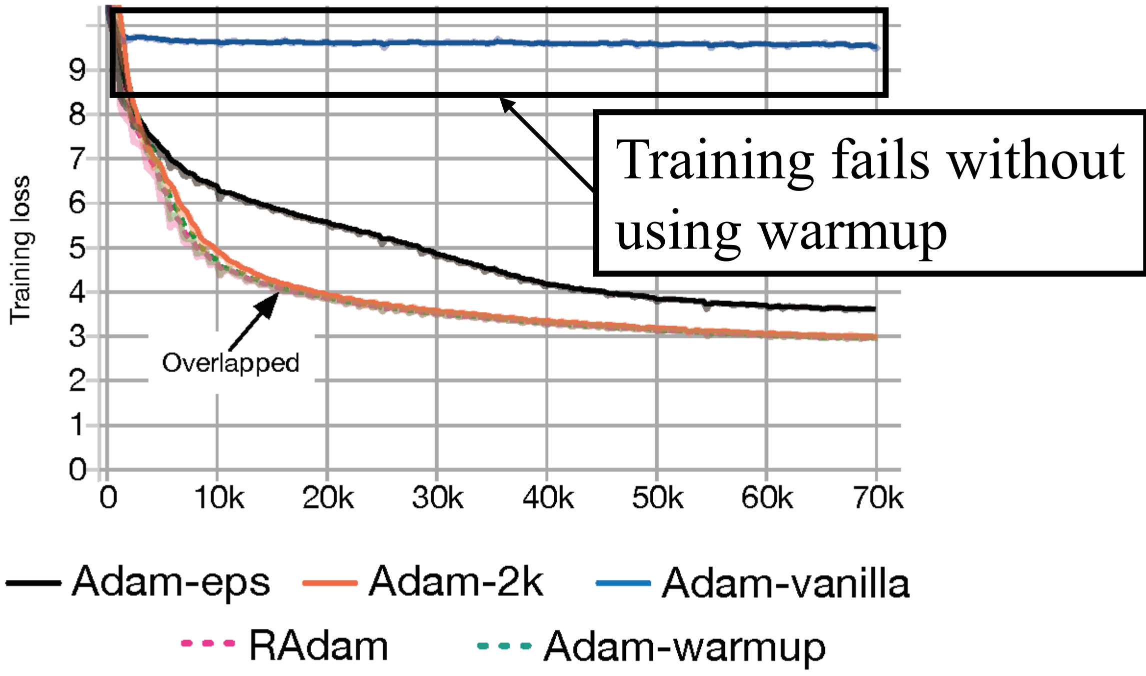
Liyuan Liu[†‡] Xiaodong Liu[‡] Jianfeng Gao[‡] Weizhu Chen[§] Jiawei Han[†]

† University of Illinois at Urbana-Champaign
‡ Microsoft Research
§ Microsoft Dynamics 365 AI



(a) (b)

Although SGD is the canonical algorithm for conventional NNs, it fails to train Transformer effectively.



Removing the warmup phrase results in more serious consequences.

Transformer requires non-trivial efforts

What Complicates Transformer Training?

Gradients Vanishing

Unbalanced gradients can hamper the training from the beginning and has been long regarded as the major reason destabilize model training.

Recent study shows that, even after introducing residual connections, the Transformer network still suffers from gradient vanishing.

Surprisingly, we find gradient vanishing is not the direct reason

Fixing the gradient vanishing issue alone cannot stabilize training.

Unbalanced gradients are largely handled by adaptive optimizers.

Fixing Initialization Stabilizes the Transformer Training

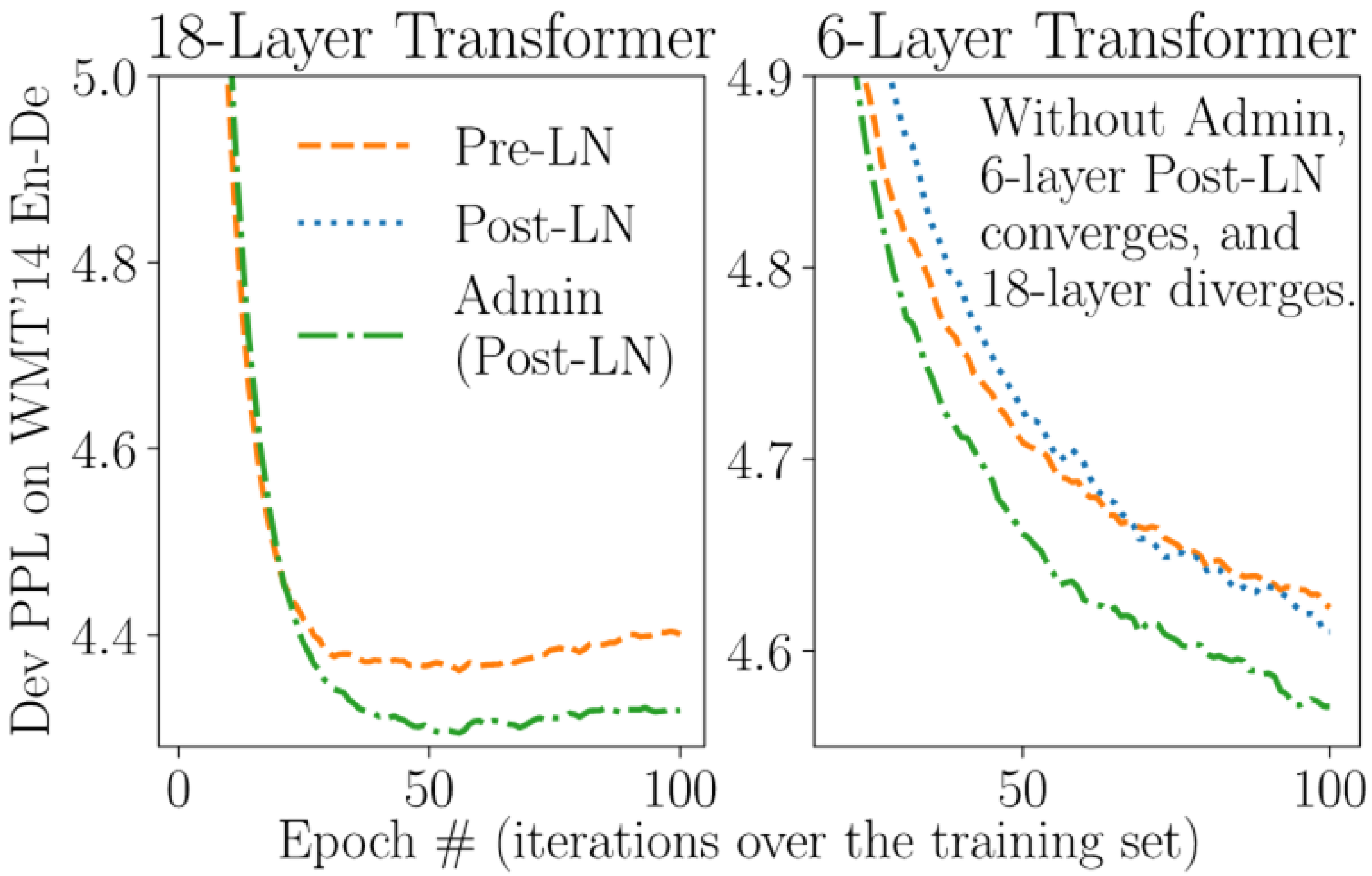
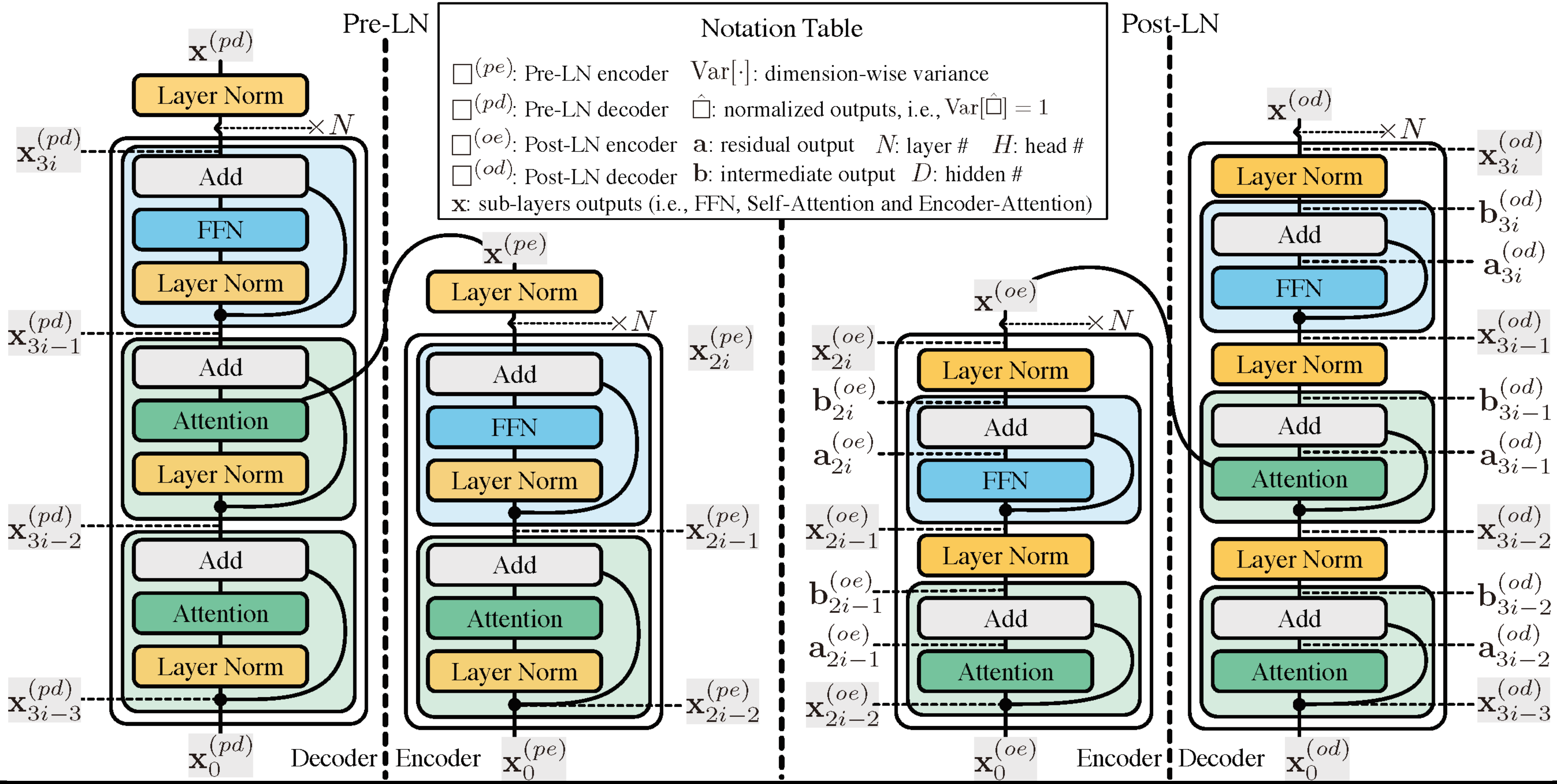


Table of Contents

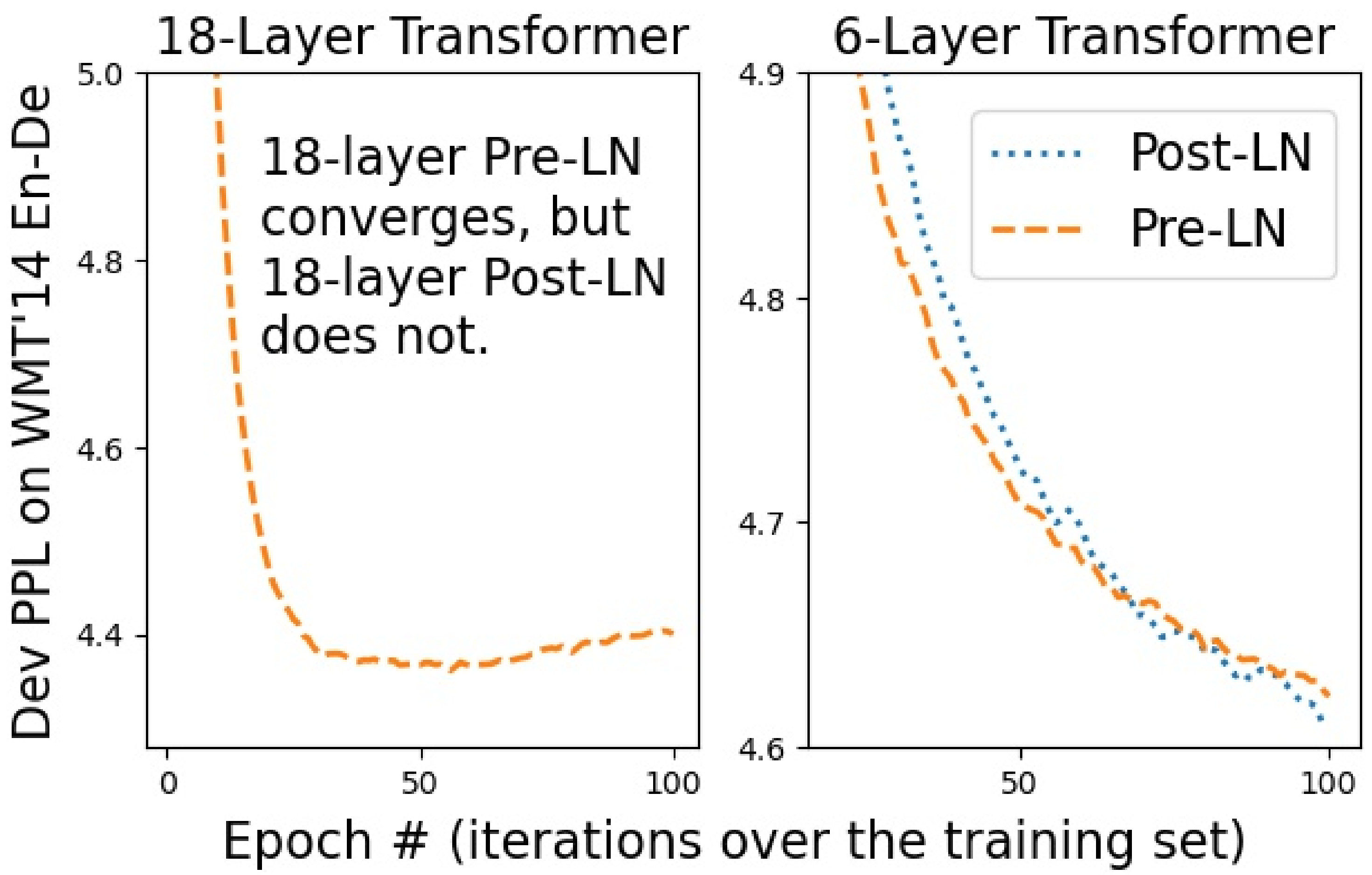


Difference between the Pre-LN and the Post-LN:

Pre-LN variants are more robust.

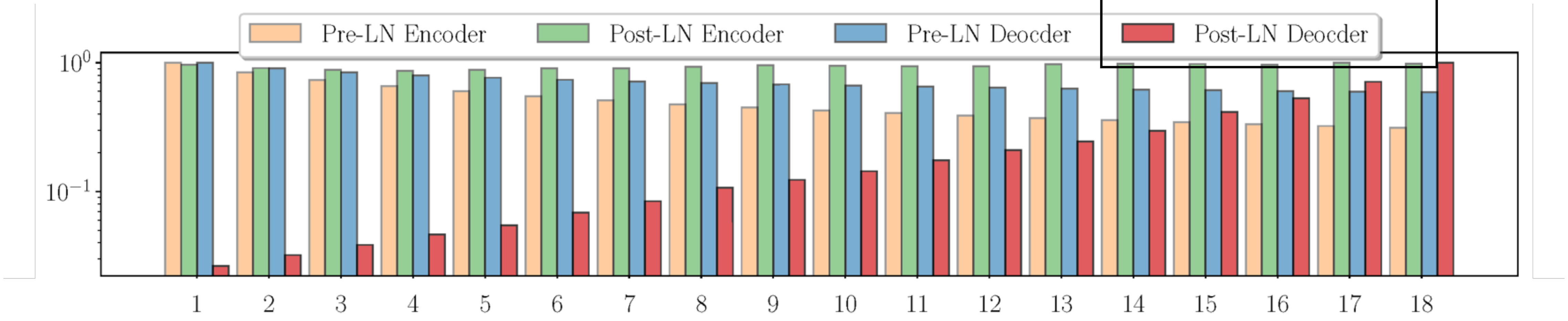
Post-LN variants have a larger potential.

Our study starts from …



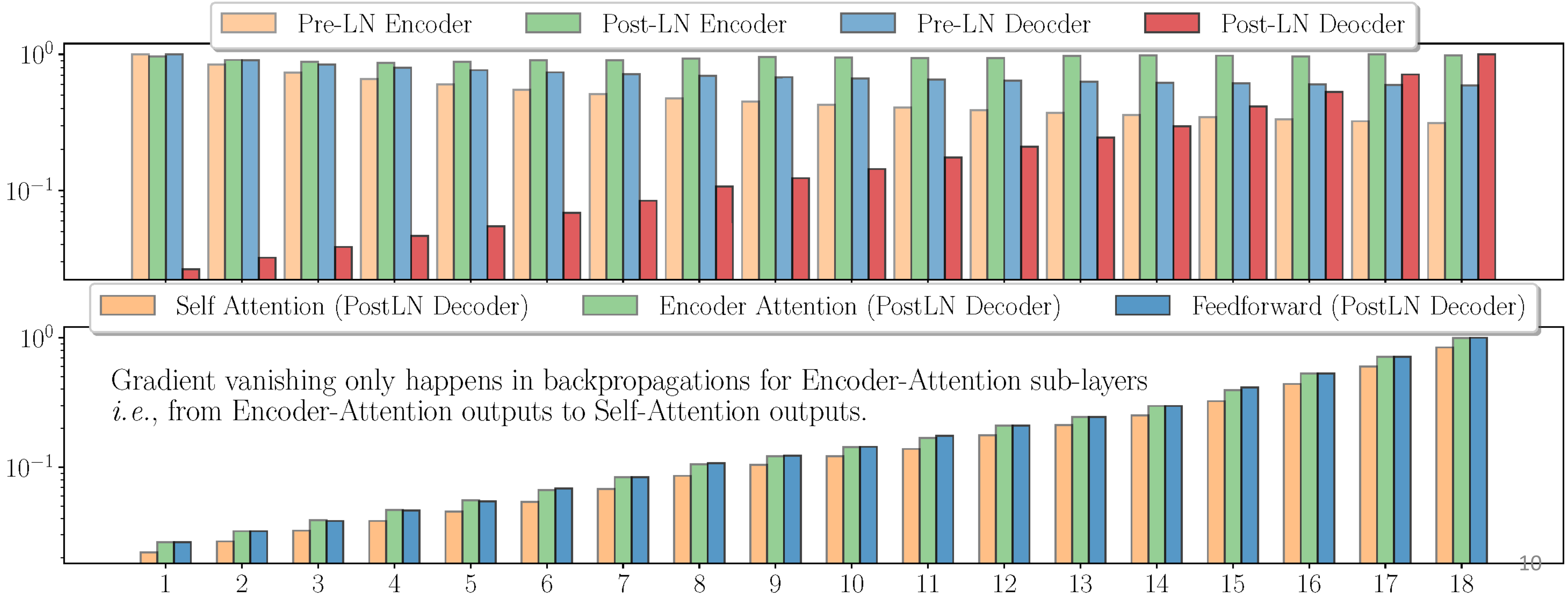
Gradient Vanishing in Transformer

Since Post-LN suffers from gradient vanishing and is not stable, it is long believed that the instability comes from gradient vanishing。



Fixing the gradient vanishing issue alone cannot stabilize training.

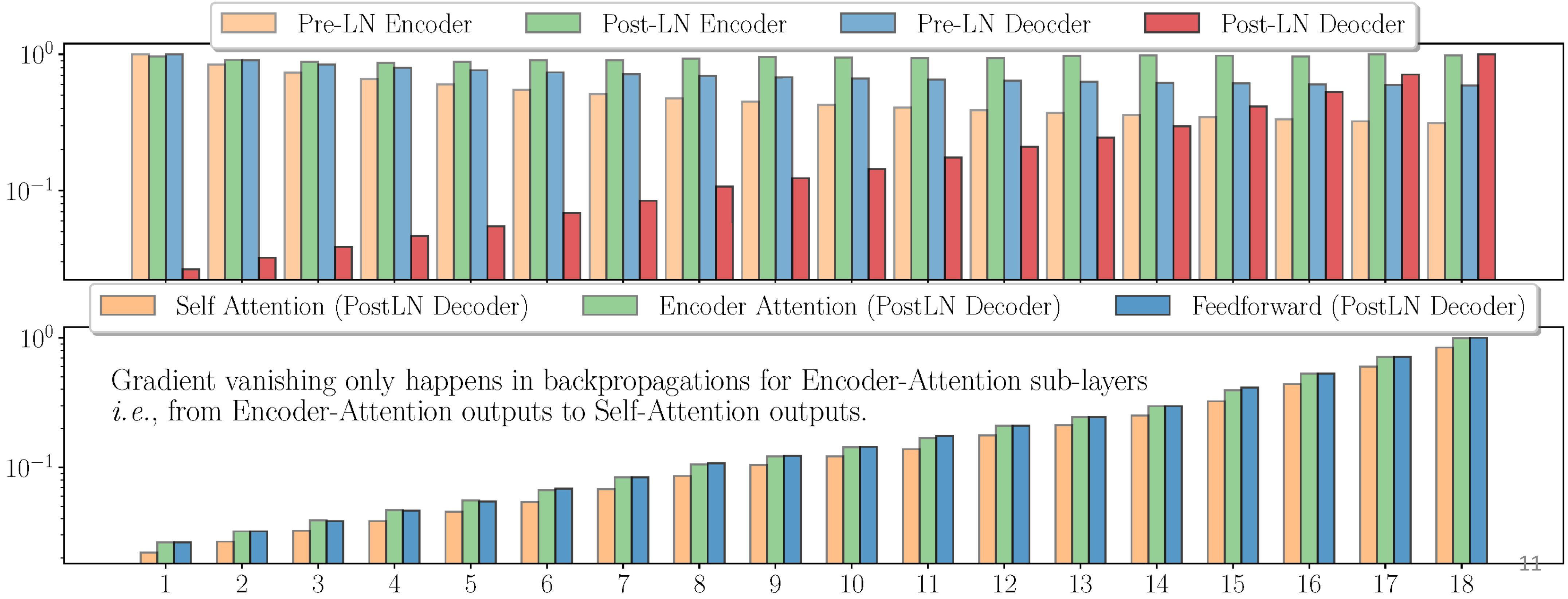
Only Post-LN decoder suffers from gradient vanishing, but neither Post-LN Encoder, Pre-LN Encoder, nor Pre-LN Decoder.



Fixing the gradient vanishing issue alone cannot stabilize training.

Only Post-LN decode Fix gradient vanishing
vanishing, but neither Post-LN Encoder, Pre-LN Encoder, nor Pre-LN Decoder.

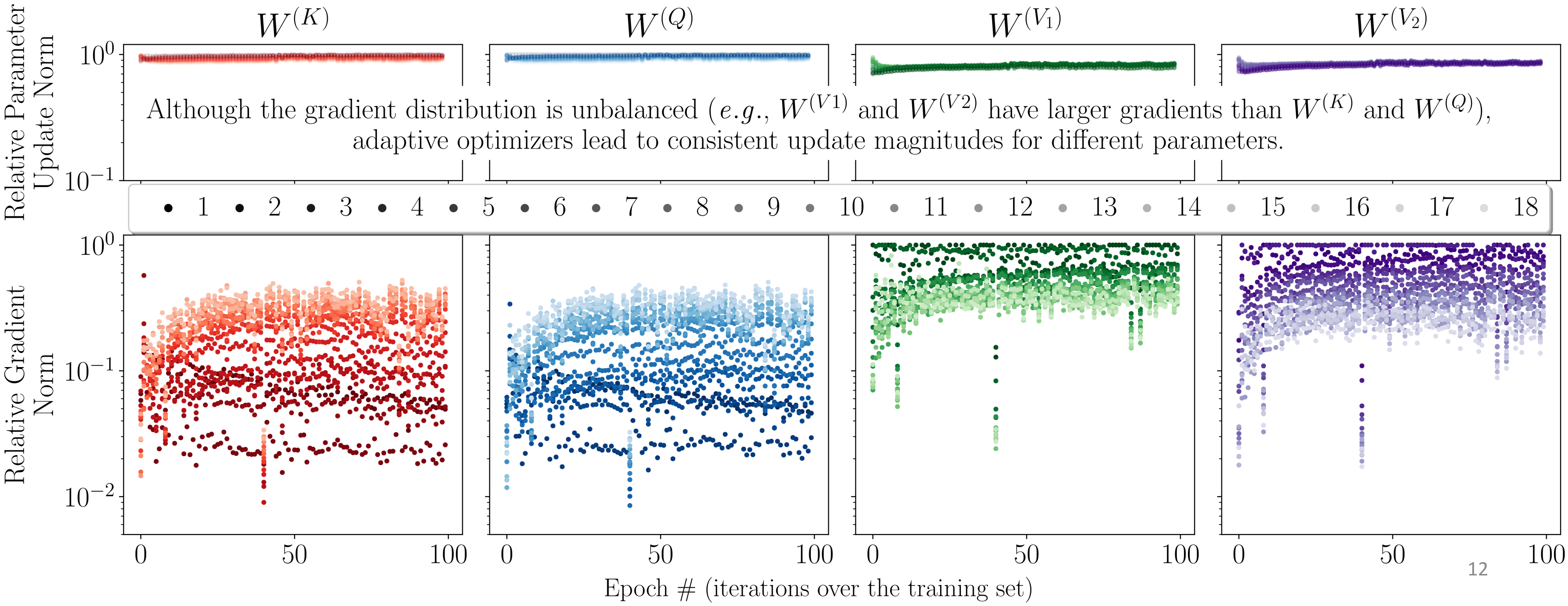
Encoder	Decoder	Gradient	Training
Post-LN	Post-LN	Vanishing	Diverged
Post-LN	Pre-LN	~~Vanishing~~	Diverged
Pre-LN	Pre-LN	~~Vanishing~~	Converged



Unbalanced gradients are largely handled by adaptive optimizers.

Relative Gradient Norm $\frac{|\nabla \boldsymbol{w}_i^t|}{\max_j |\nabla \boldsymbol{w}_j^t|}$ Relative Parameter Update Norm $\frac{|\boldsymbol{w}_i^{t+1} - \boldsymbol{w}_i^t|}{\max_j |\boldsymbol{w}_j^{t+1} - \boldsymbol{w}_j^t|}$

As unbalanced gradients are largely handled by adaptive optimizers, it necessitates the use of adaptive optimizers.



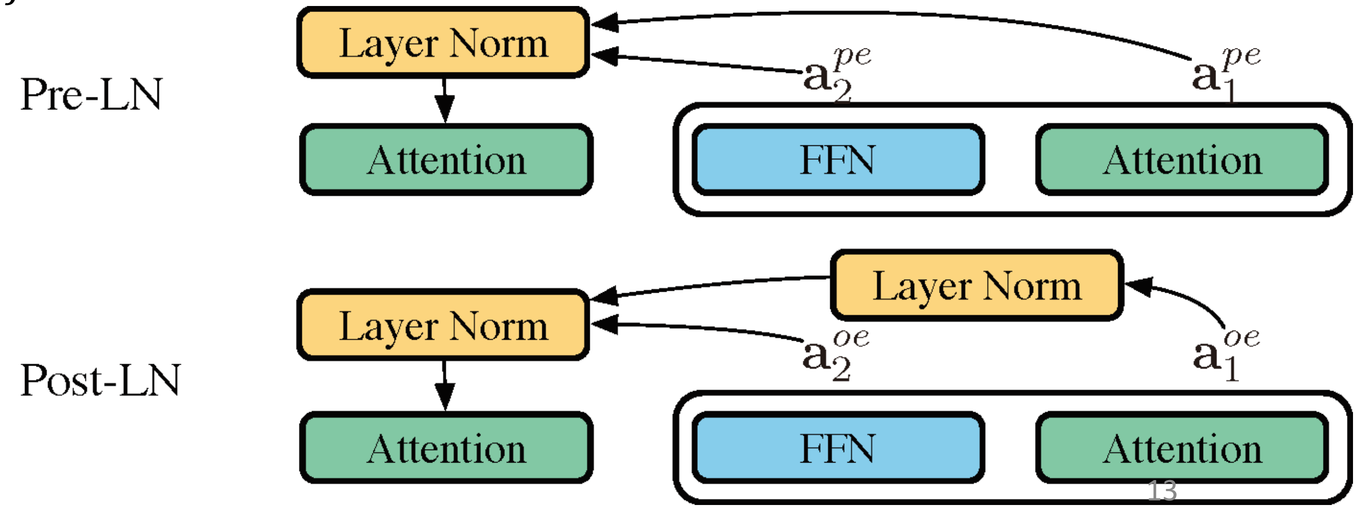
Amplification Effect

Post-LN and Pre-LN aggregates residual branch outputs differently.

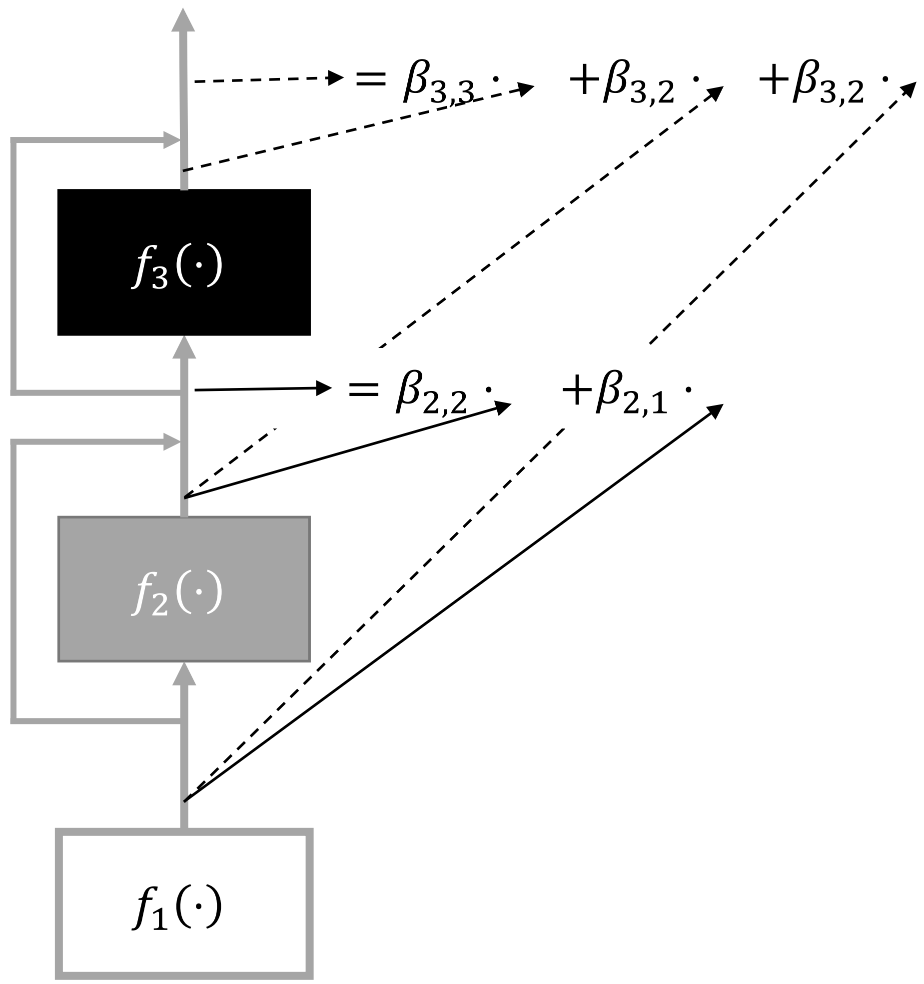
For a residual layer $x + f(x)$, we refer $f(x)$ as residual outputs and $x + f(x)$ as layer outputs

$\beta_{i,j}$ integrates all LNs and captures layer dependency.

$$\text{Input of } (\text{i+1})^{\text{th}} \text{ module/output of i}^{\text{th}} \text{ layer} = \sum_j \beta_{i,j} \cdot \text{ Normalized output of } \text{j}^{\text{th}} \text{ module}$$



$\beta_{i,j}$ integrates LNs and captures layer dependency



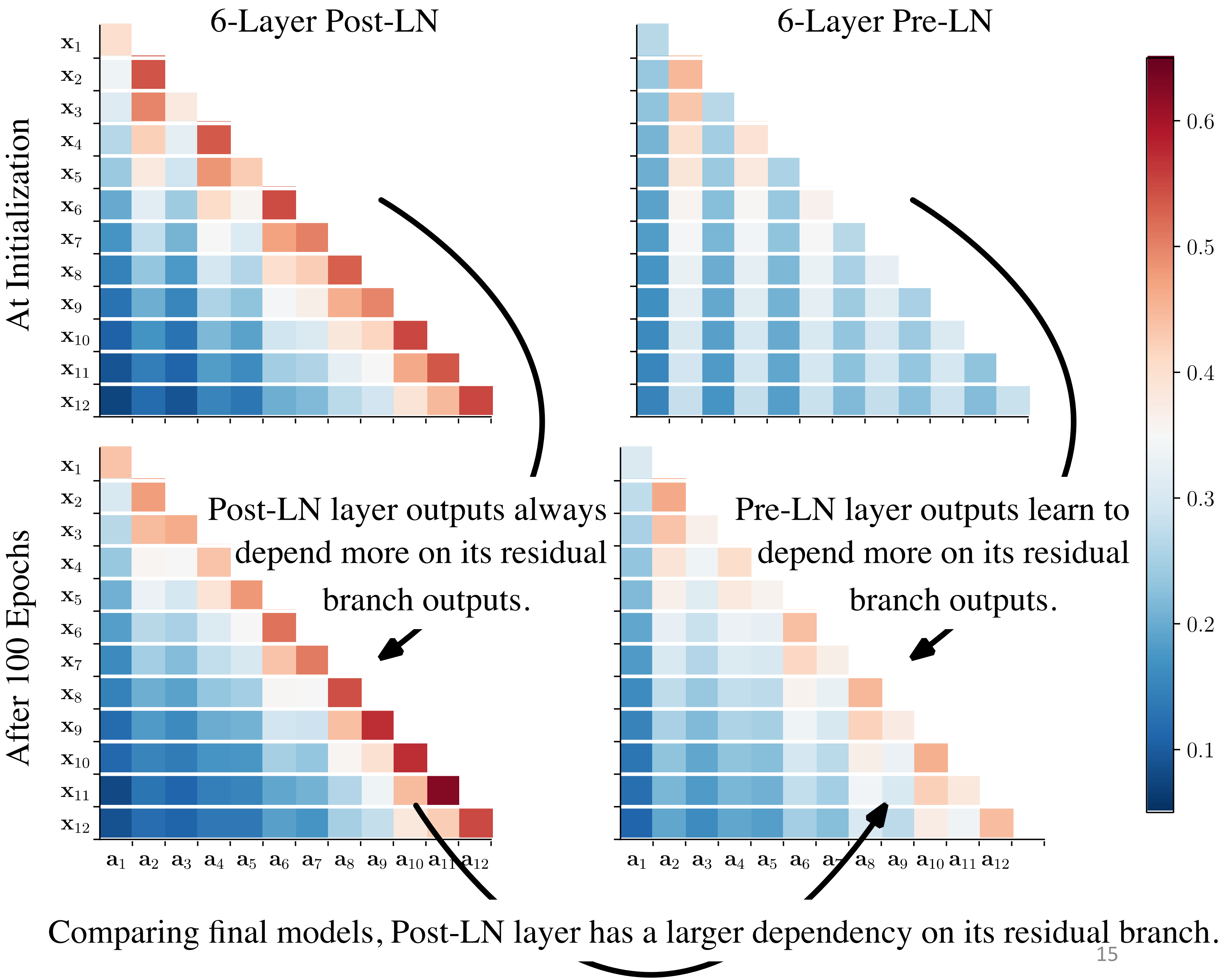
Refer $\beta_{i,i}$ as the dependency on its own residual branch.

standard deviation of j[th] output

For example, $\beta_{i,j} = \dfrac{\boxed{\mathrm{Std}[\mathbf{a_j}]}}{\boxed{\mathrm{Std}[\sum_{k\le i} \mathbf{a_k}]}}$ for Pre-LN

standard deviation of the sum of the first i outputs.

Dependency on Residual Branches



Large Dependency Destabilizes Training

Under some conditions, we have: $\mathrm{Var}[\boxed{\mathcal{F}(\mathbf{x_0}, W) - \mathcal{F}(\mathbf{x_0}, W + \delta)}] \approx \sum_{i=1}^{N} \boxed{\beta_{i,i}^2} C$

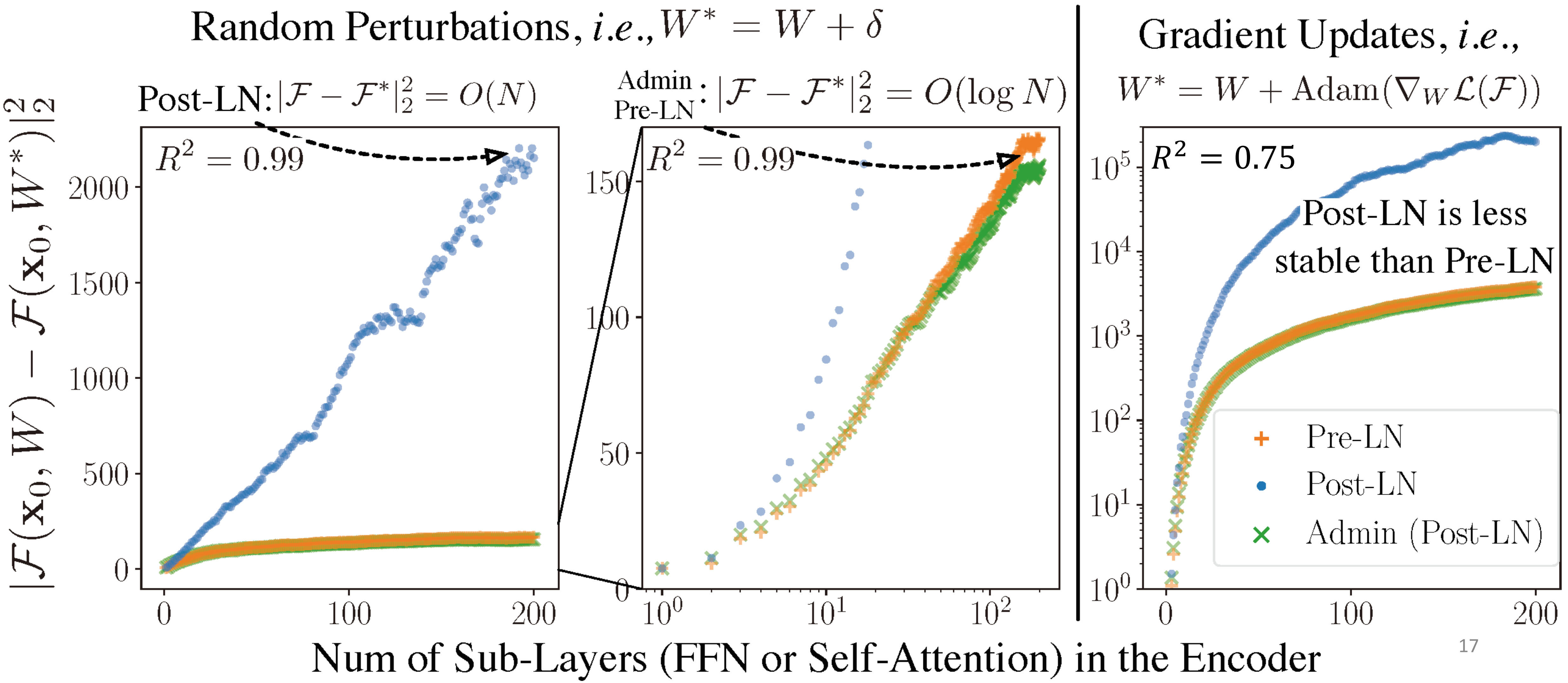
Model output change.

Dependency on its own residual branch (the weight for i^{th} residual outputs in i^{th} layer outputs).

Corollary 1. For Pre-LN, $\mathrm{Var}[\mathcal{F}(\mathbf{x_0}, W) - \mathcal{F}(\mathbf{x_0}, W + \delta)] = O(\log \mathrm{N})$ where N is layer #.

Corollary 2. For Post-LN, $\mathrm{Var}[\mathcal{F}(\mathbf{x_0}, W) - \mathcal{F}(\mathbf{x_0}, W + \delta)] = O(\mathrm{N})$ where N is layer #.

Large dependency destabilizes training



Large dependency destabilizes training

Why *warmup* helps to alleviate the instability of Post-LN?

Under some conditions, we have: $\mathrm{Var}[\mathcal{F}(\mathbf{x_0}, W) - \mathcal{F}(\mathbf{x_0}, W + \delta)] \approx \sum_{i=1}^{N} \beta_{i,i}^2 \boxed{C}$

Related to gradient norm

However, the difference between $O(\log \mathrm{N})$ and $O(\mathrm{N})$ is significant for deep networks (large N). In our experiments, <u>simply increasing the warmup steps</u> <u>fails</u> to stabilize the training of <u>deep Transformers</u> successfully.

Model Initialization

We add $\boldsymbol{\omega_i}$ to restrict the layer dependency in the early stage of Post-LN.

$$\mathbf{x}_i = f_{\mathrm{LN}}(\mathbf{b_i}), \text{where } \mathbf{b_i} = \mathbf{x_{i-1}} \cdot \boldsymbol{\omega_i} + f_i(\mathbf{x}_{i-1})$$

Admin --- **Ad**aptive **m**odel **in**itialization

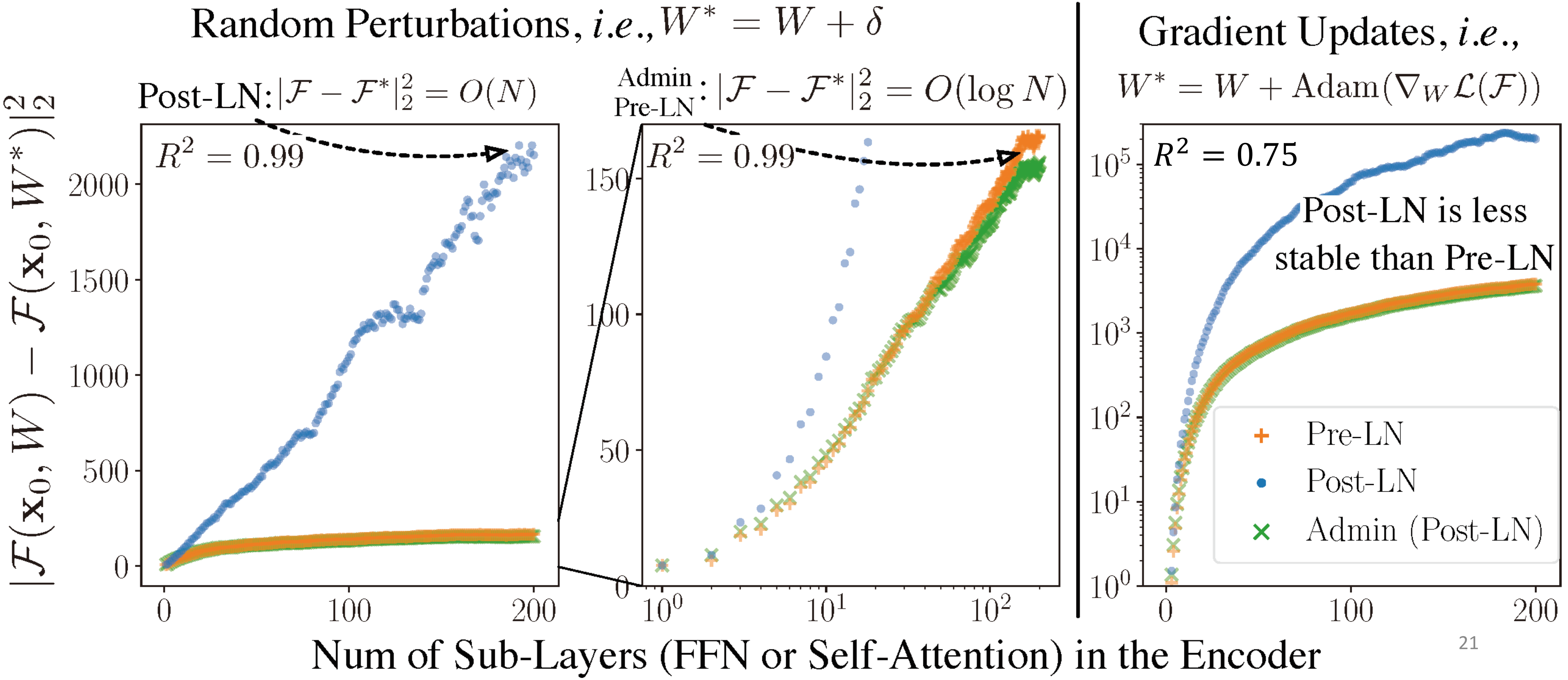
We add $\boldsymbol{\omega_i}$ to restrict the layer dependency in the early stage of Post-LN.

$$\mathbf{x}_i = f_{\text{LN}}(\mathbf{b_i}), \text{where } \mathbf{b_i} = \mathbf{x_{i-1}} \cdot \boldsymbol{\omega_i} + f_i(\mathbf{x}_{i-1})$$

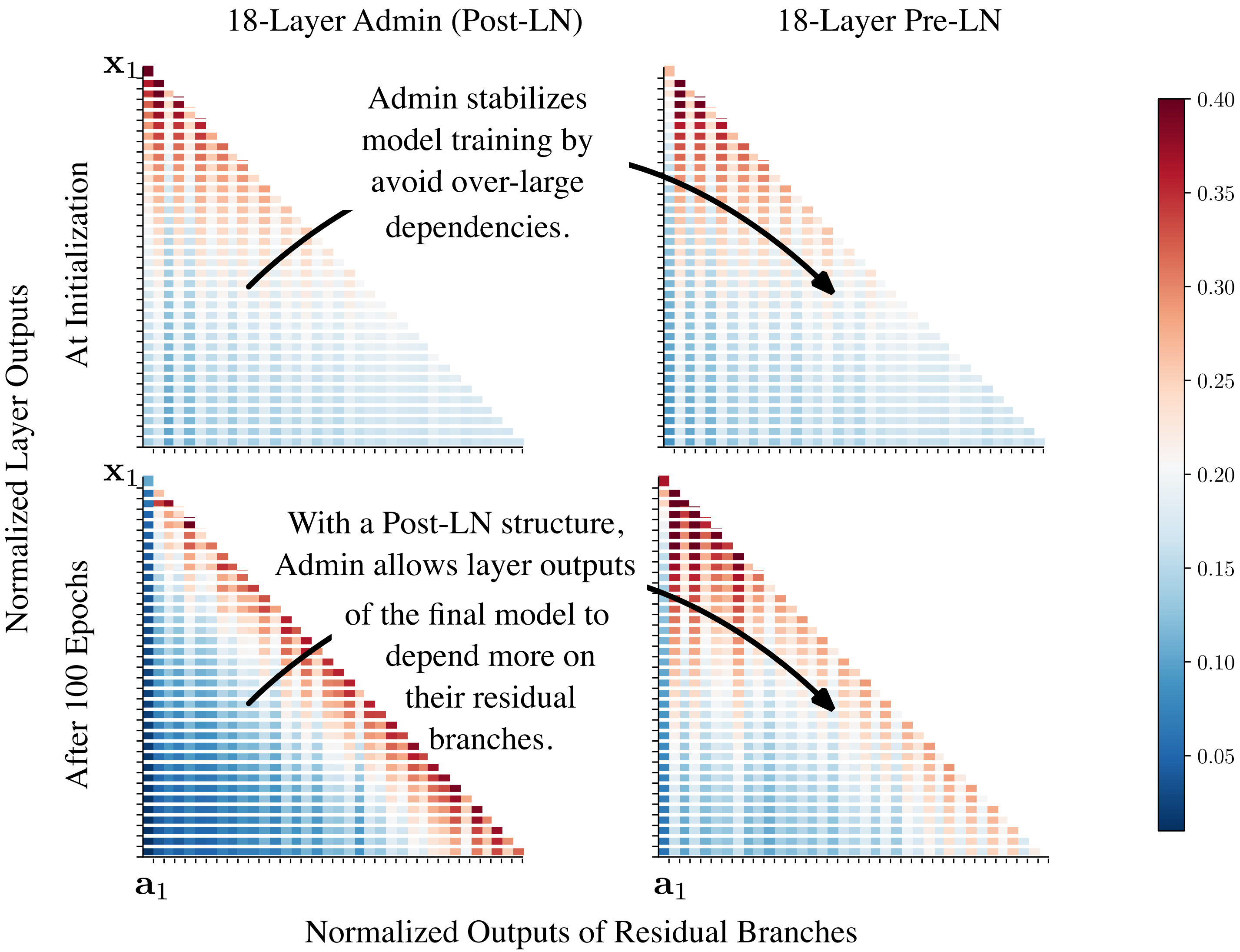
Also, we divide the initialization to two stages:

- Initialize $\boldsymbol{\omega_i}$ as 1, and empirically estimate the variance of $\mathrm{Var}[\mathbf{x}_i]$;
- Based on estimated variance, initialize $\boldsymbol{\omega_i}$ to ensure $\mathrm{Var}[\mathcal{F}(\mathbf{x_0}, W) - \mathcal{F}(\mathbf{x_0}, W + \delta)] = O(\log \mathrm{N})$ at initialization.

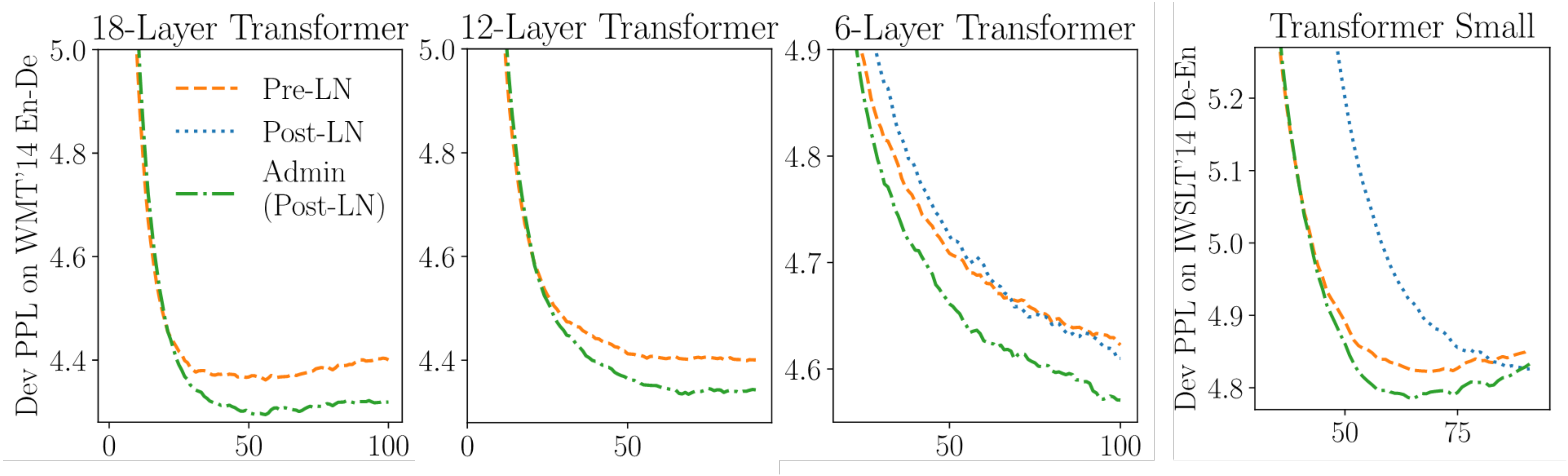
Large dependency destabilizes training



Admin --- **Ad**aptive **m**odel **in**itialization



Experiments



Experiments

Dataset	IWSLT'14 De-En	WMT'14 En-Fr		WMT'14 En-De		
Enc #-Dec #	6L-6L (small)	6L-6L	60L-12L	6L-6L	12L-12L	18L-18L
Post-LN	35.64±0.23	41.29	failed	27.80	failed	failed
Pre-LN	35.50±0.04	40.74	43.10	27.27	28.26	28.38
Admin	**35.67±0.15**	**41.47**	**43.80**	**27.90**	**28.58**	**29.03**

Without introducing any additional hyper-parameters, it achieves the new state-of-the-art on WMT'14 En-Fr (w.o. additional supervision including back translation).

Experiments

Dataset	IWSLT'14 De-En	WMT'14 En-Fr		WMT'14 En-De		
Enc #-Dec #	6L-6L (small)	6L-6L	60L-12L	6L-6L	12L-12L	18L-18L
Post-LN	35.64±0.23	41.29	failed	27.80	failed	failed
Pre-LN	35.50±0.04	40.74	43.10	27.27	28.26	28.38
Admin	**35.67±0.15**	**41.47**	**43.80**	**27.90**	**28.58**	**29.03**

We systematically evaluate deep Admin networks and summarizes results in the following report:

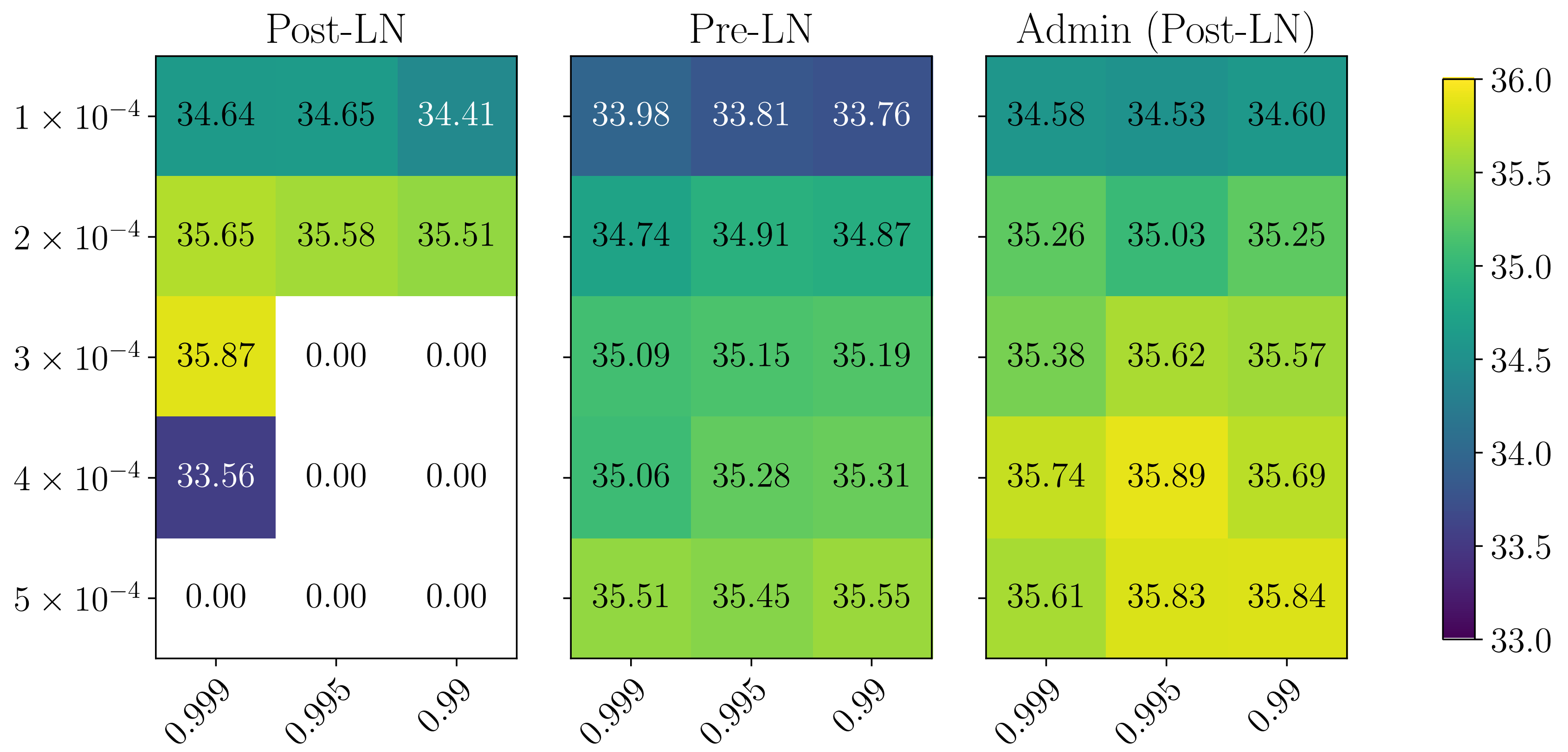
Liu, X., Duh, K., Liu, L., & Gao, J. (2020). Very deep transformers for neural machine translation. arXiv preprint arXiv:2008.07772.

Highlights: **30.1** BLEU on WMT'14 En-De, **46.4** BLEU on WMT'14 En-Fr (**w. back-translation**)

Experiments

BLEU on IWSLT'14

Learning rates



β_2 for second momentum.

Take Away

- Unbalanced gradient is not the root cause of the unstable Transformer training. It is largely addressed by adaptive optimizers.
- Large dependencies on residual branches amplifies the fluctuation and destabilizes training.
- Controlling such dependencies at initialization, Admin is more stable, converges faster, and leads to better performance.