NEURAL INFORMATION PROCESSING SYSTEMS

Critical Initialization of Wide and Deep Neural Networks using Partial Jacobians

Darshil Doshi, Tianyu He, Andrey Gromov

Paper

GitHub

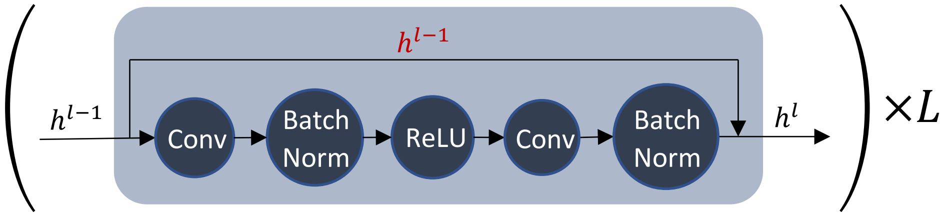
Overview

Deep neural networks need to be initialized at “criticality” to avoid exploding/vanishing gradients and ensure non-exponential scaling with depth.

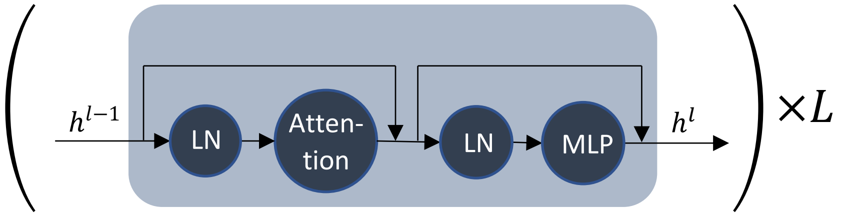
Key Contributions:

1. Novel diagnostic for critical initialization, **Averaged Partial Jacobian Norm (APJN)**, that is..
    - applicable to *general* feedforward architectures (Transformers, CNNs, MLPs etc.)
    - numerically cheap to estimate
    - analytically sound; equivalent to known theoretical measures

2. Identification and analysis of **everywhere-critical architectures**:
    - Architectures can be designed to be *critical regardless of their initialization*, by using specific combinations of normalization layers and residual connections

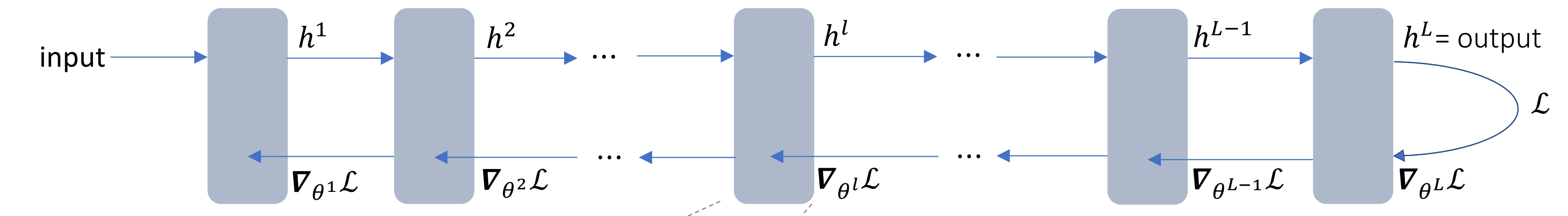


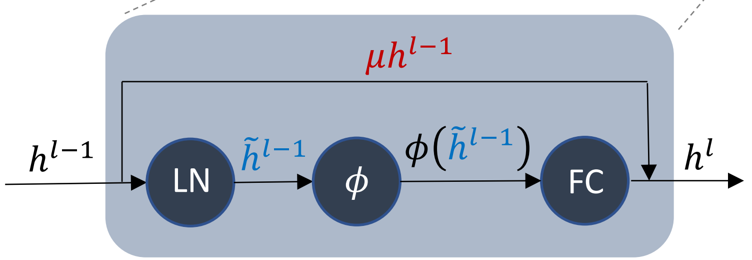
ResNet(V2)



Vision Transformer

Signal Propagation





$$\boldsymbol{h}^{l+1} = \boldsymbol{W}^{l+1}\phi(\tilde{\boldsymbol{h}}^{l}) + \boldsymbol{b}^{l+1} + \mu\boldsymbol{h}^{l}$$

$$\theta^{l+1} \coloneqq \{ W_{ij}^{l+1}\sim\mathcal{N}(0,\sigma_w^2/N_l);\quad b_i^{l+1}\sim\mathcal{N}(0,\sigma_b^2) \}$$

$N_l \to \infty$

$$h_i^l\sim\mathcal{N}(0,\mathcal{K}^l)$$

Critical Initializtion

To analyse the behaviour of gradients, we define APJN:

$$\mathcal{J}^{l_0,l} \coloneqq \mathbb{E}_\theta\left[\left\|\boldsymbol{\nabla}_{\boldsymbol{h}^{l_0}}\boldsymbol{h}^l\right\|_F^2 / N_l\right]$$

Gradients scale depends on scaling of APJN $\mathcal{J}^{l,L}$

$$\boldsymbol{\nabla}_{\theta^l}\mathcal{L} = \left(\boldsymbol{\nabla}_{h^L}\mathcal{L}\right)\left(\boldsymbol{\nabla}_{h^l}h^L\right)\left(\boldsymbol{\nabla}_{\theta^l}h^l\right)$$

$$\left\|\boldsymbol{\nabla}_{\theta^l}\mathcal{L}\right\|^2 \approx O\left(\left\|\boldsymbol{\nabla}_{\boldsymbol{h}^L}\mathcal{L}\right\|^2 \cdot \mathcal{J}^{l,L} \cdot \mathcal{K}^l\right)$$

In the limit $N_l \to \infty$, APJN can be written as a product of layer-to-layer APJNs:

$$\mathcal{J}^{l_0,l} = \mathcal{J}^{l_0,l_0+1}\,\mathcal{J}^{l_0+1,l_0+2}\cdots\mathcal{J}^{l-1,l}$$

$\mathcal{J}^{l-1,l}$ only depends on $\sigma_w^2, \sigma_b^2, \phi$ and μ.

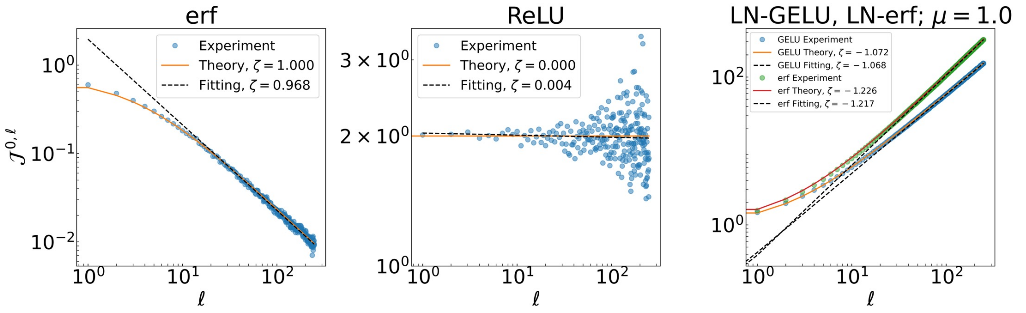
To avoid exploding/vanishing gradients, we want $\mathcal{J}^{l_0,l}$ to behave **non-exponentially** with l. This can be achieved by:

$$\left.\mathcal{J}^{l-1,l}\right|_{l\to\infty} = 1$$

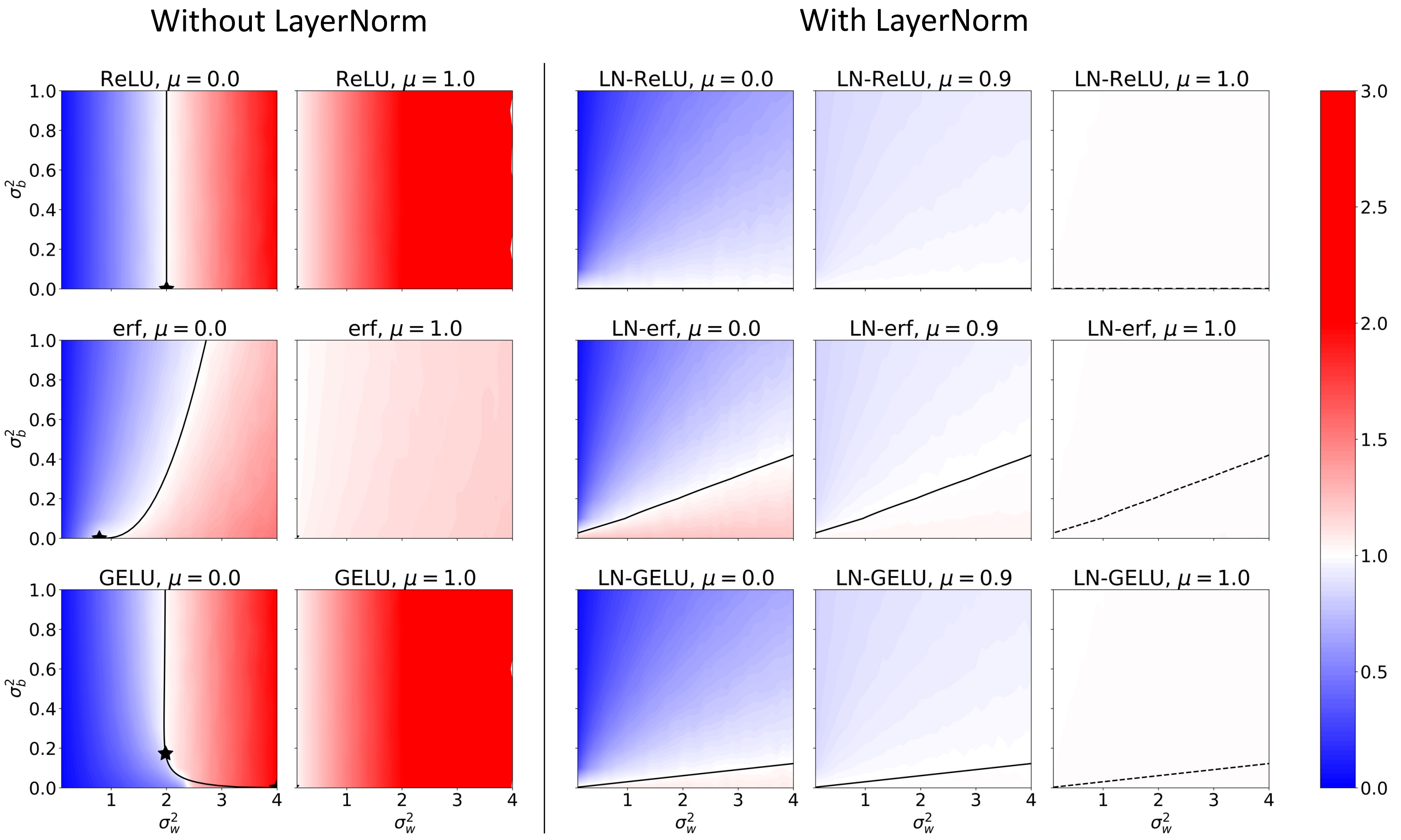
This gives us the **critical line** in the $\sigma_w - \sigma_b$ plane.

Without LayerNorm, demanding non-exponential behaviour of $\mathcal{K}^l$ gives us the **critical point** in the $\sigma_w - \sigma_b$ plane.

At the critical point, $\mathcal{J}^{l_0,l}$ scales algebraically with l : $\boxed{\mathcal{J}^{l_0,l} \sim l^{-\zeta}}$



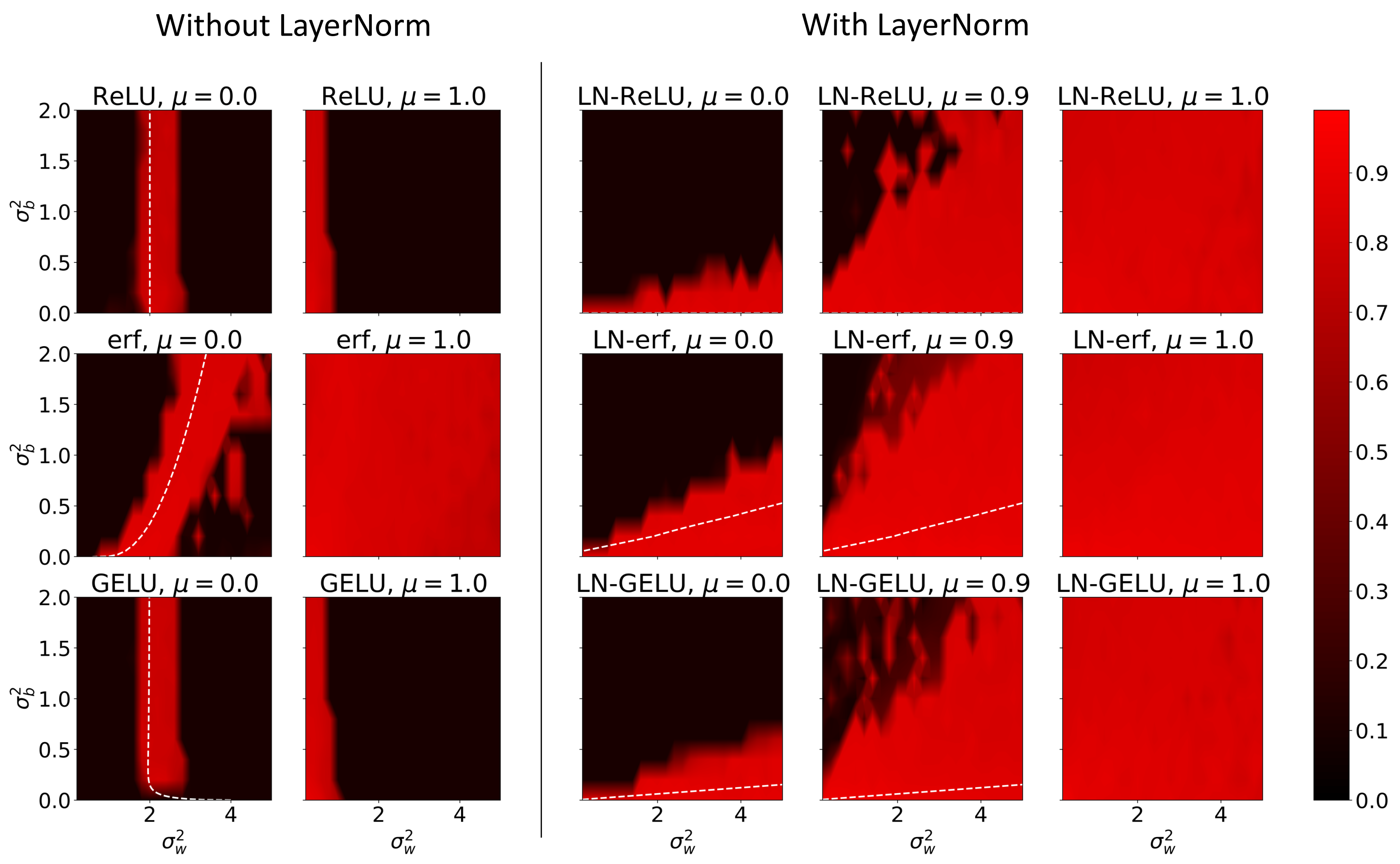
APJN Phase Diagrams



$\mathcal{J}^{l-1,l}$ phase diagrams of MLP ReLU, erf and GELU activations

- For real, finite width networks, we use numerical estimates for APJN; utilizing backward pass.
- Networks with **pre**-LayerNorm and $\mu = 1$ are **everywhere critical**!
  In this case, $\mathcal{J}^{l_0,l} \sim l^{-\zeta}$ where ζ depends on σ_w, σ_b.
- Bounded activations, with $\mu = 1$ *without* LayerNorm are **semi-critical**.
  In this case, $\mathcal{J}^{l_0,l} \sim e^{\sqrt{l/\lambda}}$, where λ depends on σ_w, σ_b.

Training Results



Training accuracy of $L = 50$ MLP on FashionMNIST dataset.

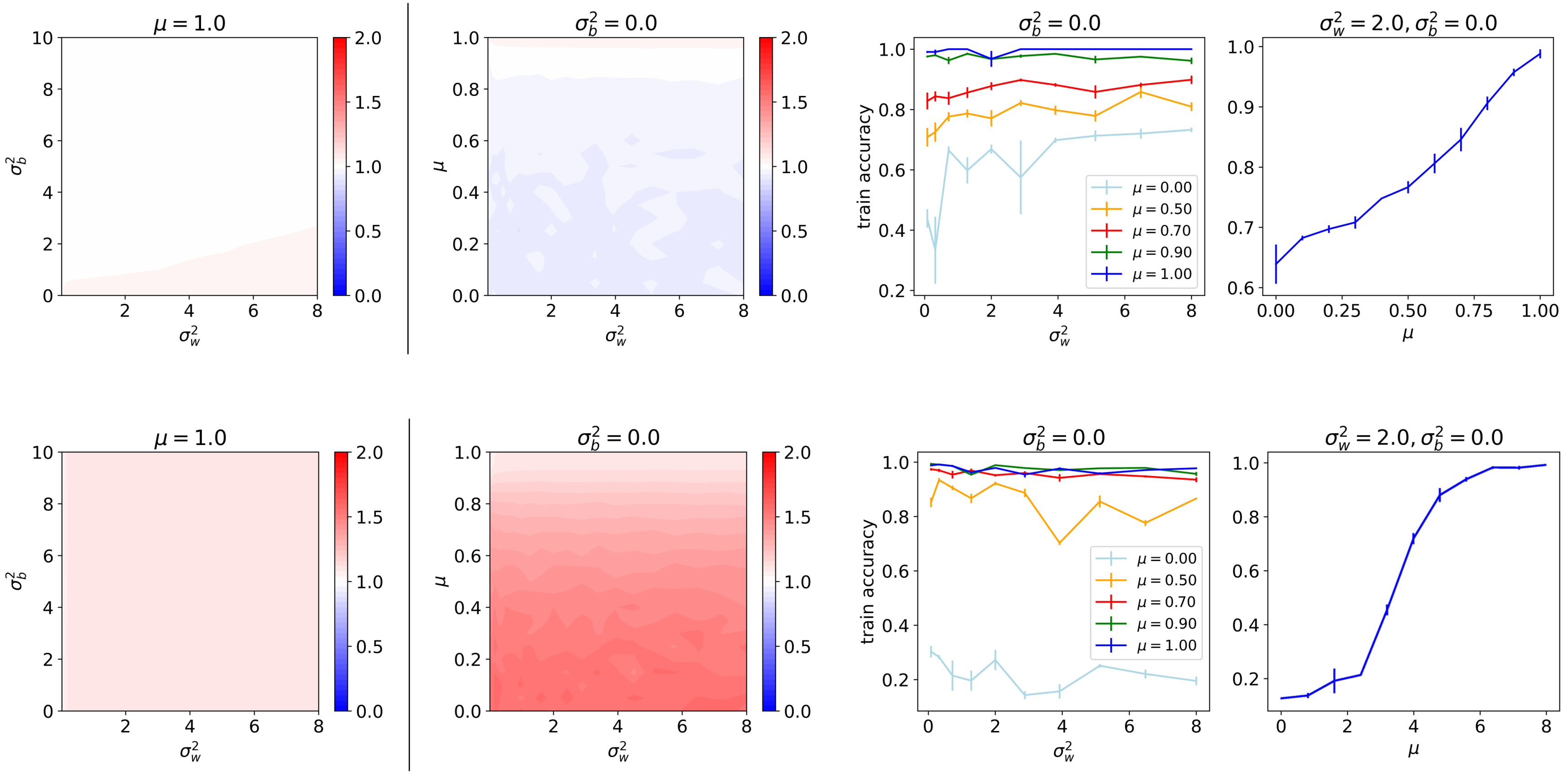
- Training results are in excellent agreement with APJN phase diagrams.
- Networks with Pre-LayerNorm and $\mu = 1$ are, in fact, **everywhere trainable**!
- Network with erf, $\mu = 1$ and no LayerNorm has enhanced trainability.

ResNet110 V2

$\mathcal{J}^{l-1,l}$ phase diagrams for $(\sigma_w - \sigma_b)$ and $(\sigma_w - \mu)$; training accuracies on CIFAR10.

LayerNorm

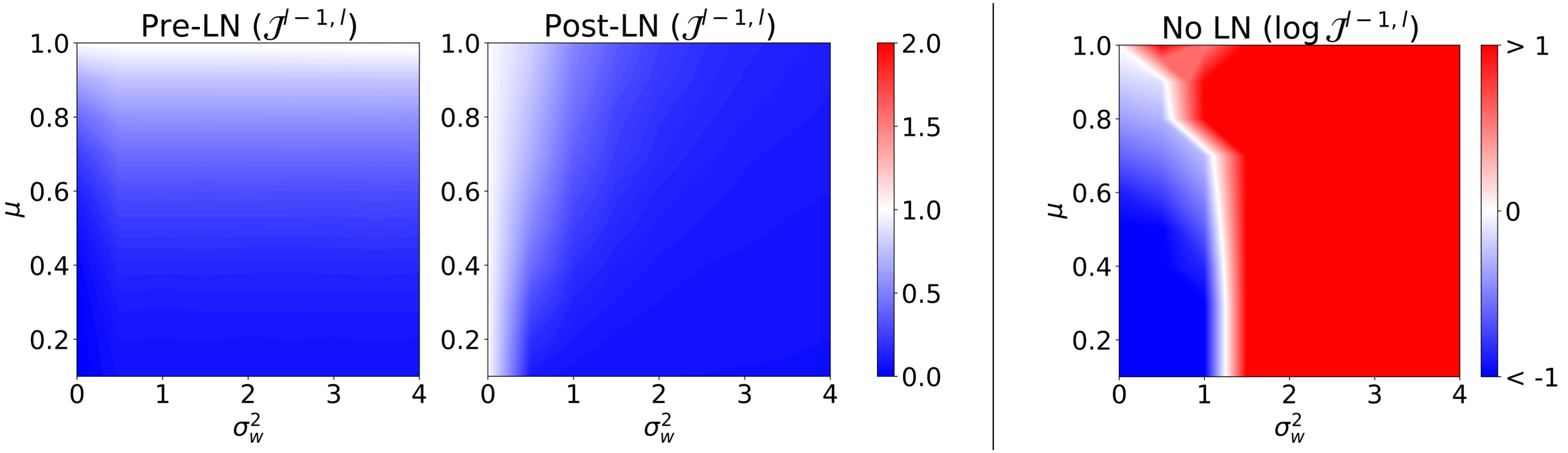
BatchNorm



- In both cases, the architecture is everywhere-critical with $\mu = 1$.
- $\mu < 1$ cases are drastically different for LayerNorm and BatchNorm.

Vision Transformer

$\mathcal{J}^{l-1,l}$ phase diagrams for $(\sigma_w - \mu)$, with pre-LN, post-LN and no LN.

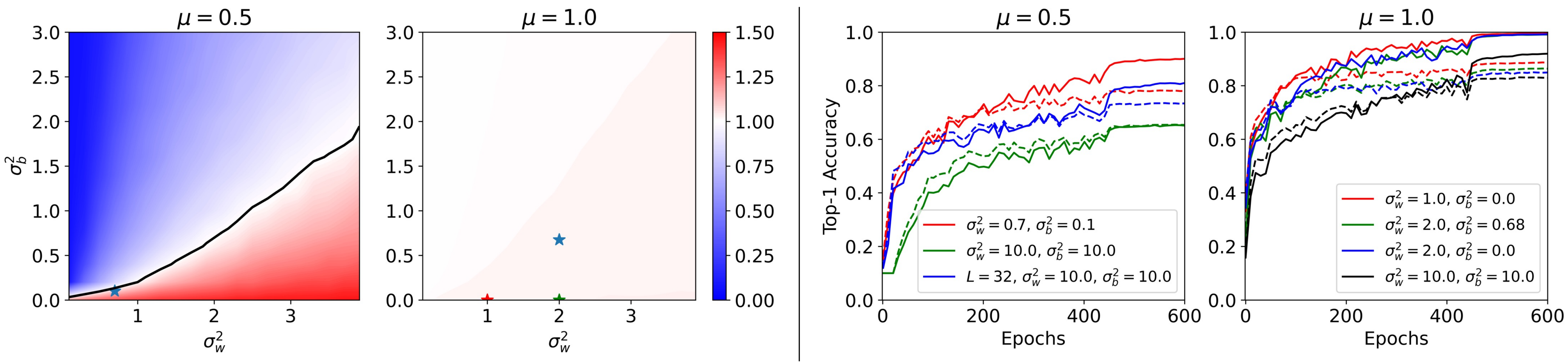


- In the pre-LN case, $\mu = 1.0$ is *everywhere-critical*.
- Post-LN and no LN cases do not feature everywhere-criticality.
- The advantage of Pre-LN Transformer is empirically known in literature.

Xiong et al. "On Layer Normalization in the Transformer Architectures". (2020)

MLP-Mixer

$\mathcal{J}^{l-1,l}$ phase diagrams for $\mu = 1.0$ and $\mu = 0.5$; training accuracies on CIFAR10.



- $\mu = 1.0$ case is *everywhere-critical*; while $\mu = 0.5$ is not.
- As a result, $\mu = 1.0$ trained well for all initializations; whereas $\mu = 0.5$ deteriorates far from the critical line.

Thank you!

Questions + comments?

Paper

GitHub

Email Darshil