Analyzing the Training Dynamics of Image Restoration Transformers: A Revisit to Layer Normalization

MinKyu Lee, Sangeek Hyun, Woojin Jun, Hyunjun Kim, Jiwoo Chung, Jae-Pil Heo
Sungkyunkwan University

github.com/2minkyulee/i-LN

KeyWords: Image Restoration (IR), IR Transformer, LayerNorm (LN)

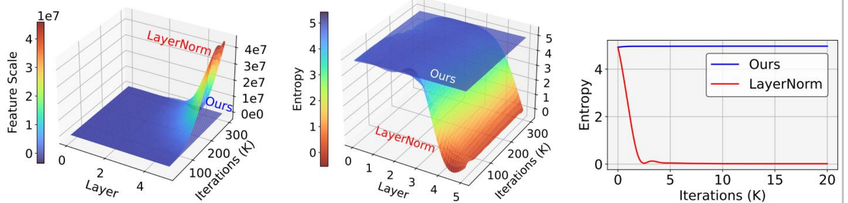
TLDR; LayerNorm disrupts spatial information, thus is not suitable for IR tasks.

Image Restoration aims to obtain clean RGB images, given degraded images as input.
- Includes tasks such as Super-Resolution (SR), Denoising (DN), Deraining (DR), and JPEG Compression Artifact Removal (CAR).

Here, we focus on conventional **IR Transformers** that uses **LayerNorm (LN)**
- This includes representative backbone models such as SwinIR, HAT, DRCT
- We focus on the 1) per-token and 2) input independent normalization scheme of LN.

Observation and Key Motivation

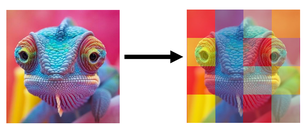
Observation of Extreme Feature Statistics under Vanilla LN



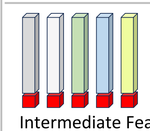
(a) Feature Magnitudes (b) Channel-wise Entropy (c) Entropy close-up

- Under vanilla LN, feature magnitude diverges up to ***million-scale***, and channel-wise Entropy **collapses** (i.e., there exists extremely peaky channels)

Conflicts btw LN and IR



1. LN works in a **Per-token manner:**
   → Disrupts spatial correlation between tokens/pixels.
2. LN works in an **Input-blind manner:**
   → Features are mapped into a unified space, thereby losing low-level statistics (e.g., contrast)

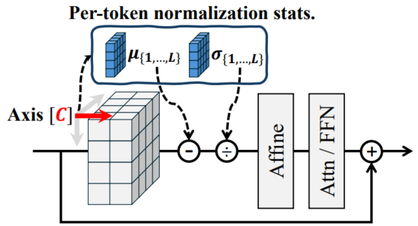


A small portion of feature, with extremely large values

The Network Wants to Ignore LN

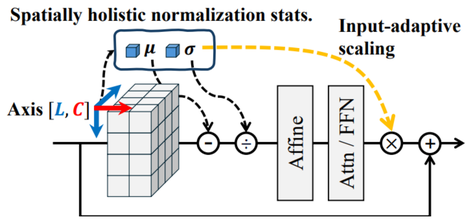
→ The network learns to generate components that dominate the overall statistics, thereby bypassing the LN operation.

Method: Image Restoration Tailored LayerNorm (*i*-LN)



Vanilla LN (that has limitations)

Per-token norm disrupts spatial relationship
Input-blindness removes input statistics



Our Solution (i-LN)

Do it in a spatially holistic manner
Restore the removed statistics

Pseudo-Code (LN vs i-LN)

```
def ffn_LN(x):

    skip = x.clone()
    mean, std = get_stats( x, dim=C )
    x = (x-mean)/std
    x = x * gamma + beta # Affine
    x = ffn(x)

return x+skip
```

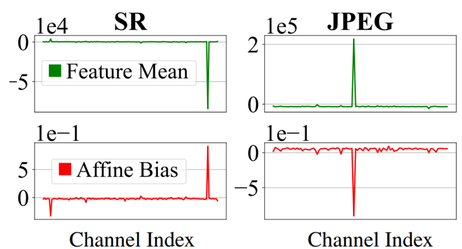
Vanilla LN

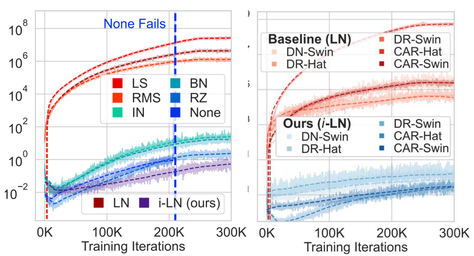
```
def ffn_iLN(x):

    skip = x.clone()
    mean, std = get_stats( x, dim=L,C )
    x = (x-mean)/std
    x = x * gamma + beta # Affine
    x = ffn(x) * std

return x+skip
```

Proposed i-LN

Analysis

- Affine Params in vanilla LN
  → the network wants to cancel out the peaky channels
- Feature Mag. Under Various Norm scheme/ IR Task
  → Extreme features are observed in various IR tasks.





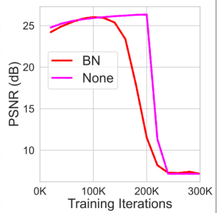
Experimental Results

- Experiment results on Super-Resolution (Comparing vanilla LN vs our i-LN)
  → HAT_1 indicates a lightweighted setting.

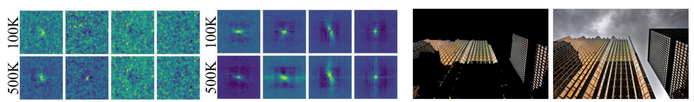
Backbone	Scale	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HAT_1 + LN	×2	38.14	.9610	33.78	.9196	32.19	.9000	32.16	.9297	38.84	.9778
HAT_1 + *i*-LN	×2	**38.37**	**.9619**	**34.08**	**.9218**	**32.42**	**.9028**	**33.32**	**.9385**	**39.69**	**.9794**
DRCT_1 + LN	×2	38.19	.9613	33.28	.9197	32.28	.9010	32.60	.9323	39.23	.9785
DRCT_1 + *i*-LN	×2	**38.23**	**.9614**	**33.86**	**.9206**	**32.31**	**.9014**	**32.79**	**.9344**	**39.40**	**.9788**
HAT_1 + LN	×4	32.51	.8992	28.79	.7876	27.68	.7411	26.55	.8015	31.01	.9150
HAT_1 + *i*-LN	×4	**32.72**	**.9019**	**29.01**	**.7915**	**27.84**	**.7456**	**27.17**	**.8167**	**31.82**	**.9228**
DRCT_1 + LN	×4	32.50	.8989	28.85	.7871	27.73	.7414	26.63	.8021	31.24	.9169
DRCT_1 + *i*-LN	×4	**32.57**	**.8997**	**28.91**	**.7887**	**27.76**	**.7426**	**26.79**	**.8063**	**31.41**	**.9188**

- Comparison Against other Normalization Schemes (†: eval-mode.)

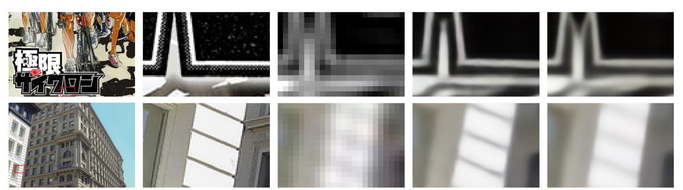
Idx	Method	SH	Set14		BSD100		Urban100		Manga109	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
1	LayerNorm	✗	28.79	.7876	27.68	.7411	26.55	.8015	31.01	.9150
2	LayerScale	✗	28.89	.7887	27.76	.7426	26.75	.8058	31.37	.9178
3	RMSNorm	✗	28.88	.7879	27.74	.7417	26.67	.8037	31.24	.9165
4	ReZero	✓	28.81	.7861	27.70	.7406	26.41	.7964	31.05	.9147
5	None	✓	-	-	-	-	-	-	-	-
6	InstanceNorm	✓	28.98	.7907	27.80	.7445	27.02	.8136	31.46	.9199
7	BatchNorm†	✓	28.95	.7901	27.80	.7442	26.70	.8123	31.39	.9186
8	*i*-LN (Ours)	✓	**29.01**	**.7915**	**27.84**	**.7456**	**27.17**	**.8167**	**31.82**	**.9228**



- Relative Position Embedding Visualization
  → Ours show improved spatial patterns
- Post-Training Quantization
  → Baseline fails under fp16



- Visual Comparison for Real-world x4 Super-Resolution with the HAT_1 Model



(a) HQ (b) HQ crop (c) Input (d) *i*-LN (Ours) (e) LN