Transformers without Tears: Improving the Normalization of Self-Attention

Toan Q. Nguyen & Julian Salazar
(equal contribution)

IWSLT 2019, Hong Kong
paper: https://arxiv.org/pdf/1910.05895.pdf
code: https://github.com/tnq177/transformers_without_tears

… or simply how to train Transformer

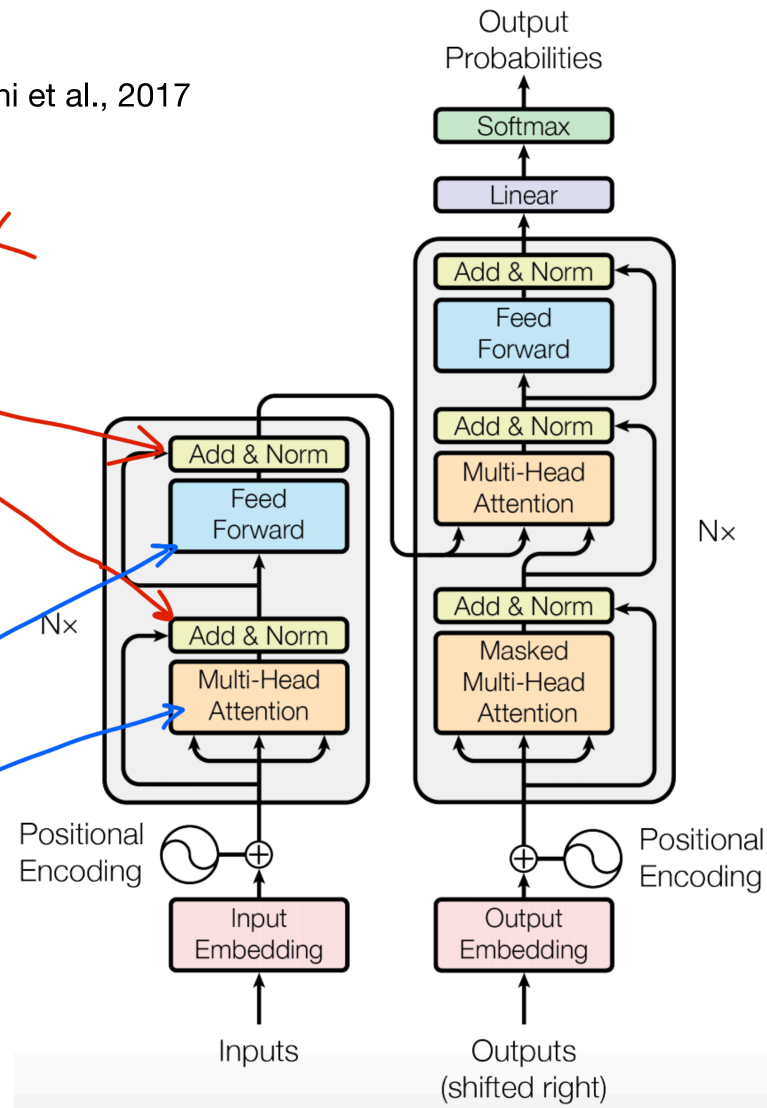
Outline

- Transformer
- **Stability**: PreNorm vs PostNorm
- **Stability**: Weight initialization
- **Low-resource NMT**: FixNorm
- **Low-resource NMT**: ScaleNorm
- Experiment results
- Analysis

Transformer

“Attention is all you need”, Vaswani et al., 2017

3 This talk



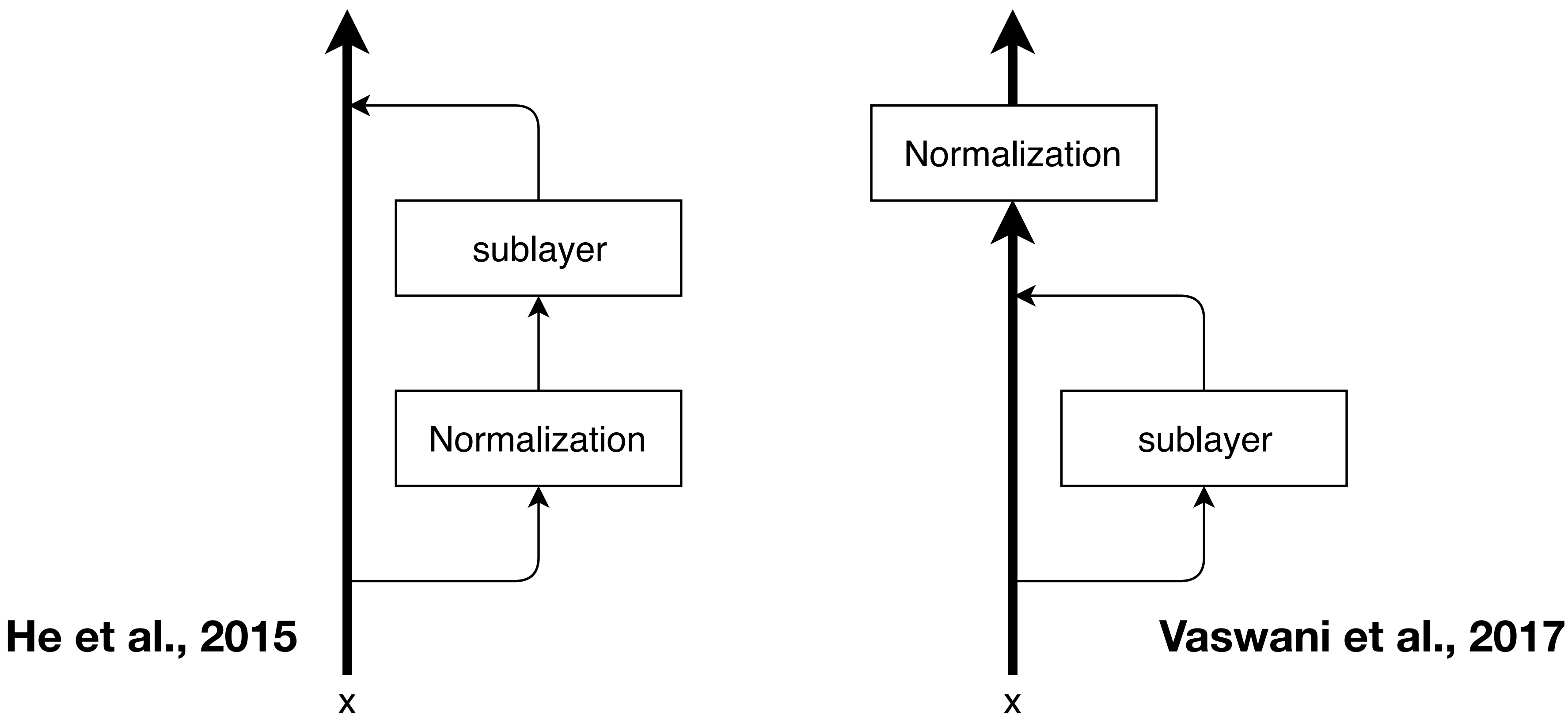
2

1

Problems?

- If you implement your own Transformer, you may find a problem with **stability**:
    - **NO** warmup, **NO** convergence
    - **WITH** warmup, sometimes still **NO** convergence
    - We'll show that the problem lies in the Residual Connections
- If you care about **low-resource NMT**:
    - Most previous works on training Transformer are about high-resource (Vaswani et al. 2017, Shazeer and Stern 2018, Popel and Bojar 2018, Chen et al. 2018…)
    - Can we train better low-resource NMT with Transformer?
    - We'll show that we can do better with better normalization

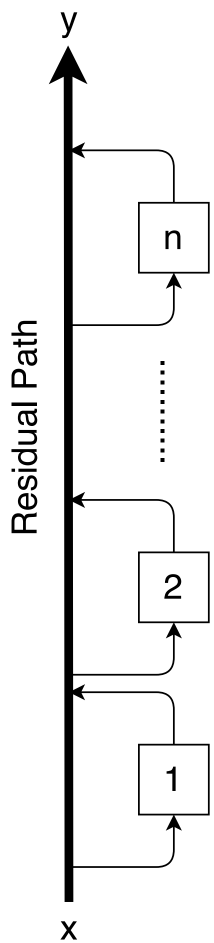
Stability: PreNorm and PostNorm



ResNet (PreNorm)

Transformer (PostNorm)

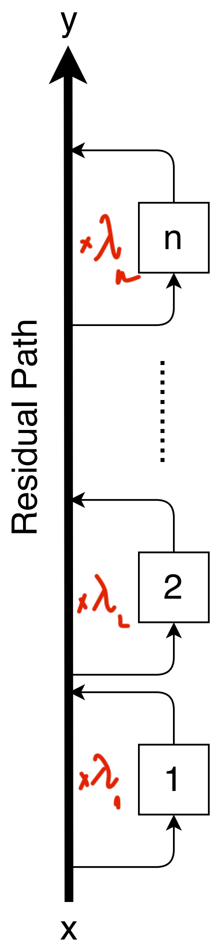
PreNorm (ResNet)



$$x_{l+1} = x_l + F_l(x_l)$$

==> Contribution of x to y: x

PreNorm (ResNet)



(He et al., 2016)
Assume we multiply output
of layer i some scalar λ_i before residual addition
i.e. $x_{l+1} = \lambda_l x_l + F_l(x_l)$

Contribution of x to y: $x\prod_1^n \lambda_i$

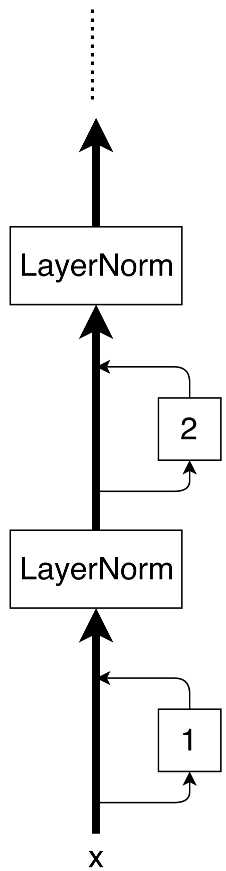
If $\lambda_i > 1,\ \prod_1^n \lambda_i \gg \Rightarrow$ **gradient explosion**

If $\lambda_i < 1,\ \prod_1^n \lambda_i \ll \Rightarrow$ **vanishing gradient**

==> λ_i should always be set to 1 (identity)

Identity mappings are very important for
healthy back-propagation

PostNorm (Transformer)



Inserting LayerNorm in between Residual Path is similar to $\lambda_i \neq 1$ which causes of Transformer's instability

Wang et al., 2019 have similar analysis to He et al., 2016 explaining how PreNorm is more stable than PostNorm

Stability: PreNorm vs PostNorm

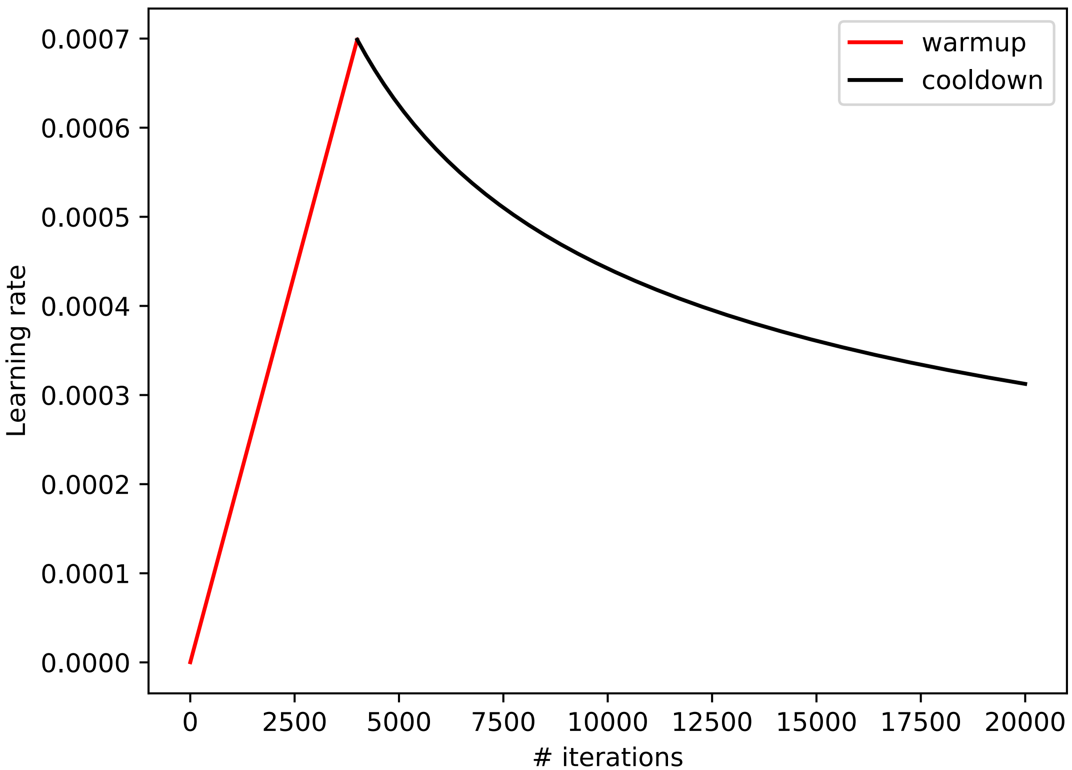
So we should always do PreNorm for Transformer

Mentioned in various works (Chen et al. 2018, Wang et al. 2019, Anonymous 2019, Parisotto et al. 2019)

Implemented in popular toolkits (tensor2tensor, fairseq, sockeye)

Also by other practitioners https://tunz.kr/post/4, https://github.com/tnq177/witwicky

Stability: Warmup



Warmup: gradually increase learning rate

Can it help LayerNorm to slowly adapt for healthier back-propagation?

Stability: Warmup

Xavier normal		# warmup steps 4k	8k	16k
Baseline	PostNorm	fail	fail	5.76
	PreNorm	28.52	28.73	28.32

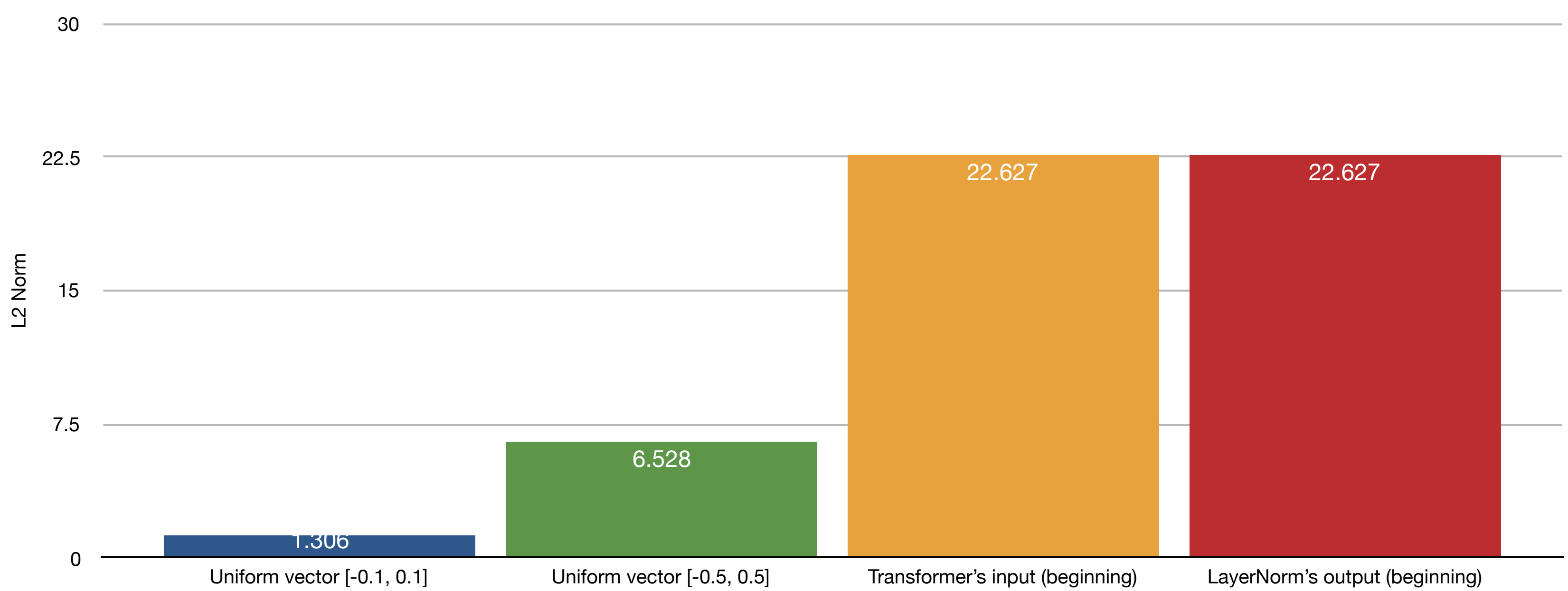
Still fails for PostNorm?

Stability: weight initialization

No convergence: starting weights are too big?

…but we use pretty standard scheme: Xavier Normal…

Transformer has big signals ==> need smaller weights



Stability: weight initialization

Attention's weights: $W_i \sim \mathcal{N}(0, \frac{2}{D+D})$

Feedforward's weights: $W_i \sim \mathcal{N}(0, \frac{2}{D+4D})$

Since feedforward's weights are small, we isolate the problem to that of attention's weights

We propose SmallInit: All weights initialized by

$$W_i \sim \mathcal{N}(0, \frac{2}{D+4D})$$

Stability: weight initialization

Xavier normal		# warmup steps 4k	8k	16k
Baseline	PostNorm	fail	fail	5.76
	PreNorm	28.52	28.73	28.32
SmallInit	PostNorm	28.17	28.20	28.62
	PreNorm	28.26	28.44	28.33

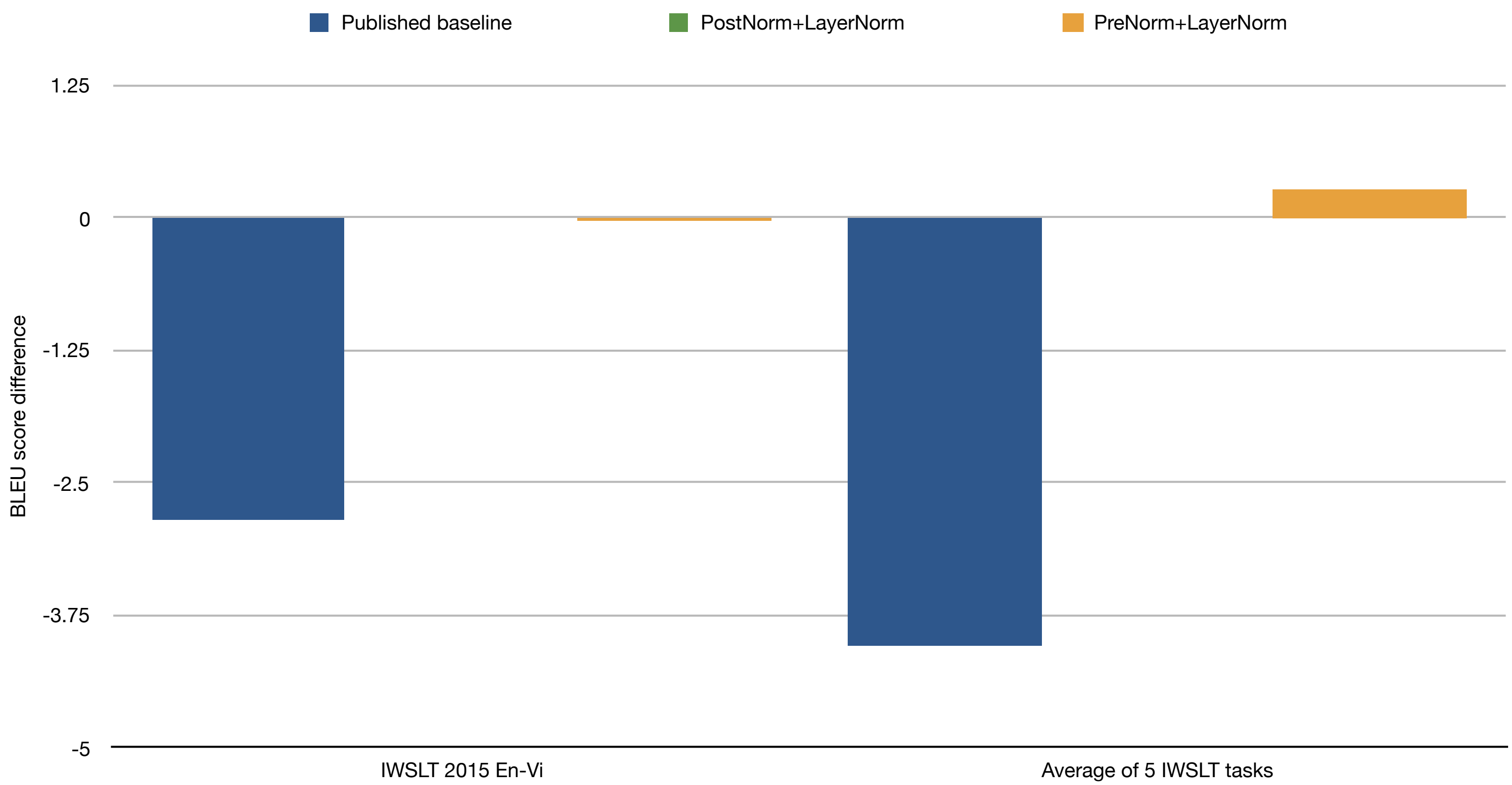
Table 2: Development BLEU on *en*→*vi* using Xavier normal initialization (baseline versus SmallInit).

Warmup+SmallInit regains stability for PostNorm

But PreNorm always works under any settings

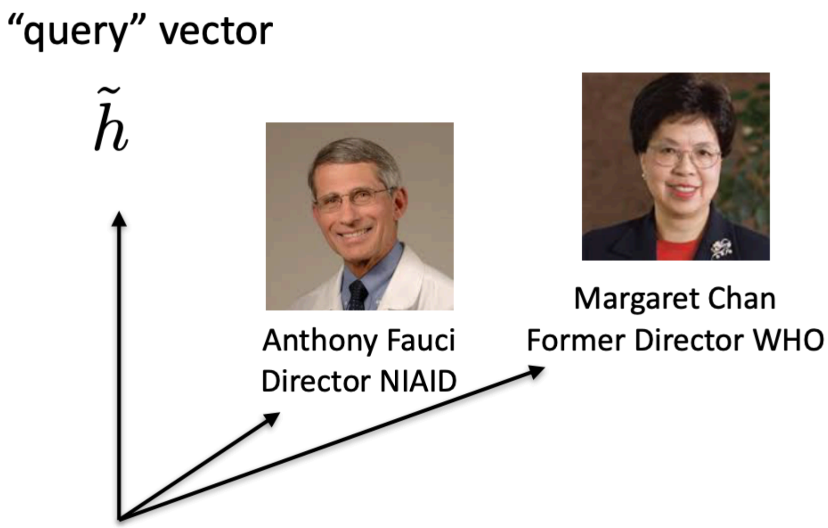
We always use PreNorm unless noted otherwise

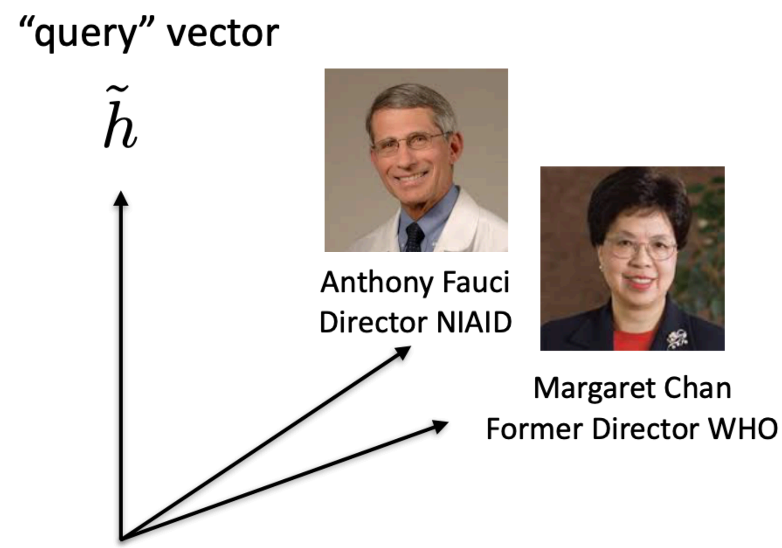
Experiments



We use IWSLT 2015 En-Vi and 4 other IWSLT datasets from Ye et al., 2018. Use BPE. More details in the paper.

Low-resource: FixNorm

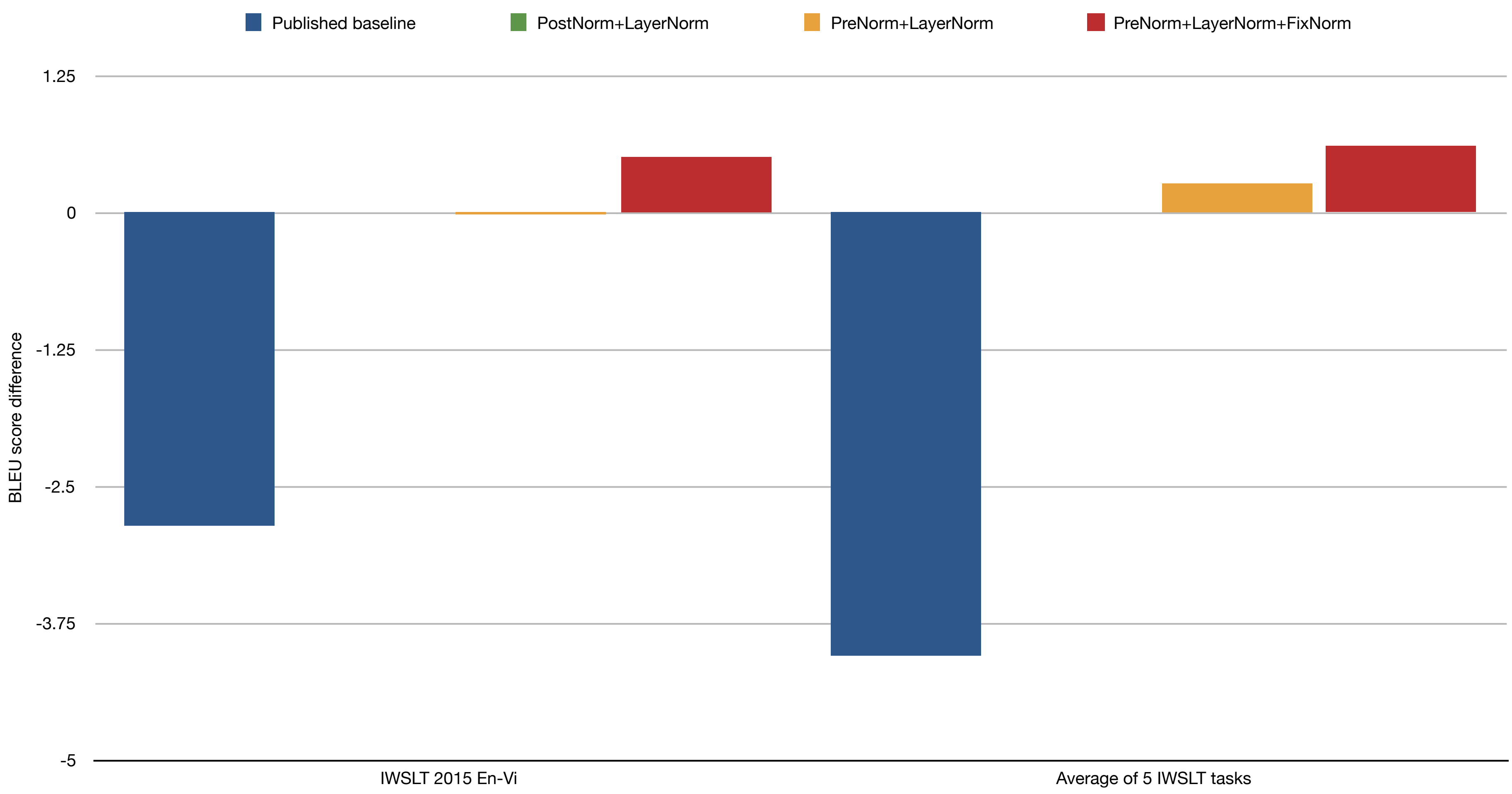




More frequent words have bigger norm than its semantically similar rare words. In this case, the model mistranslates "Fauci" to "Chan".

Solution: Fix word embedding norm to 1: $e \leftarrow \dfrac{e}{||e||}$ (Nguyen and Chiang 2018)

Experiments



Low-resource: Layer Normalization

LayerNorm (Ba et al., 2016) stems from BatchNorm (Ioffe and Szegedy, 2015)

Ioffe and Szegedy, 2015: BatchNorm helps by solving the internal covariate shift

Santurkar et al., 2018: BatchNorm actually helps by smoothing the loss landscape

Santurkar et al., 2018: **other ℓ_p normalization methods work too**

Zhang and Sennrich, 2019: propose RMSNorm which normalizes by root mean square. It's **faster** than LayerNorm, achieves **comparable** result.

Low-resource: ScaleNorm

LayerNorm: $\bar{x}_i = \dfrac{x_i - \mu}{\sigma} a_i + b_i$

RMSNorm: $\bar{x}_i = \dfrac{x_i}{\text{RMS(x)}} a_i$, RMS(x) $= \sqrt{\dfrac{1}{n} \sum_{i=1}^{n} x_i^2}$

We propose ScaleNorm: $\bar{x} = g \dfrac{x}{\|x\|}$

Low-resource: ScaleNorm

ScaleNorm is similar to FixNorm but at the input-level instead of output word embedding

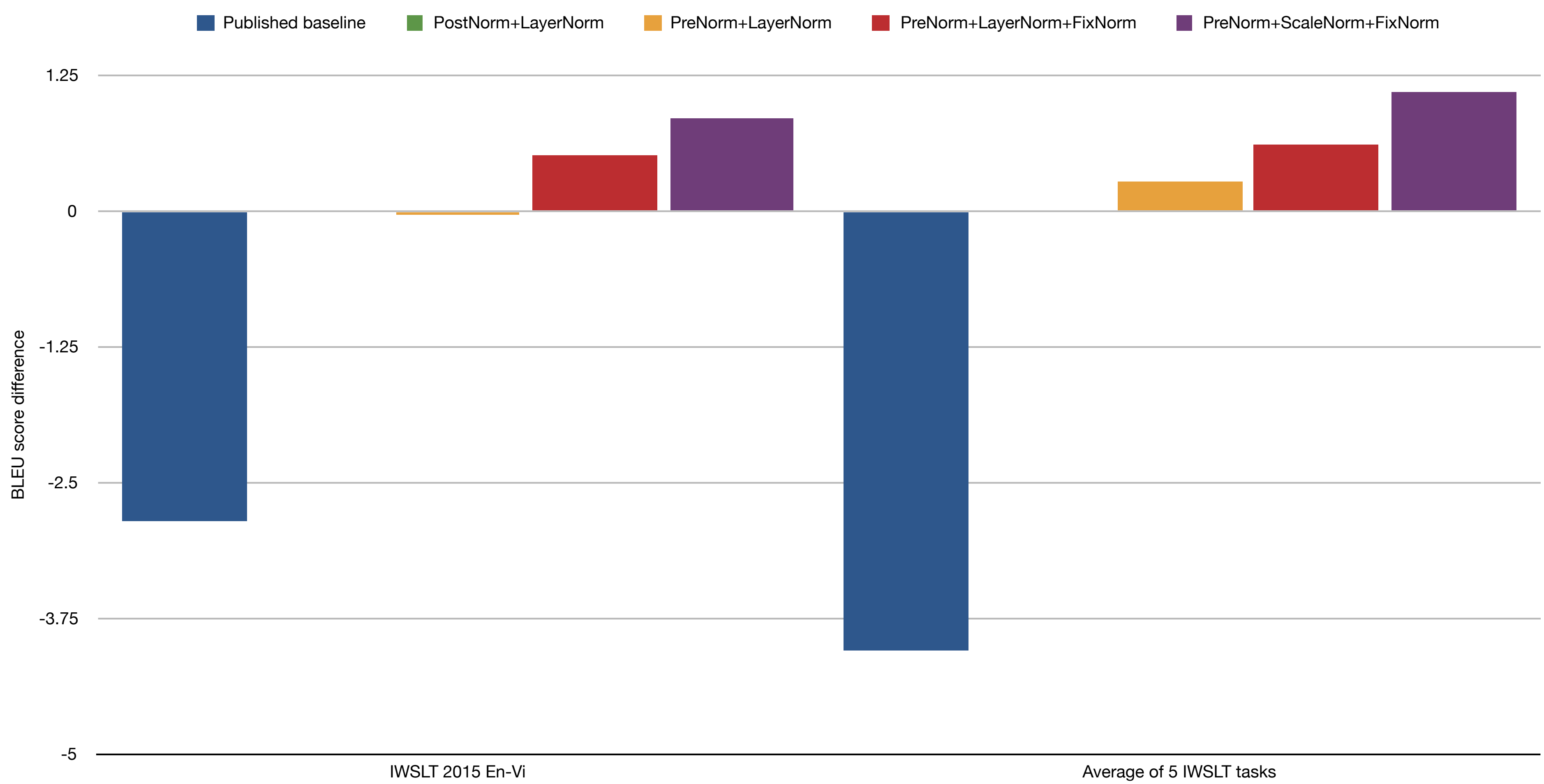
ScaleNorm has **no centering, no mean-shifting after scaling, 1 scale parameter per layer**

Speed: ScaleNorm > RMSNorm > LayerNorm

ScaleNorm+FixNorm at final output layer means maximizing cosine distance

Nguyen and Chiang 2018 uses a special case of ScaleNorm+FixNorm which shows improving translation for low-resource NMT

Experiments



Experiments: Learning rate

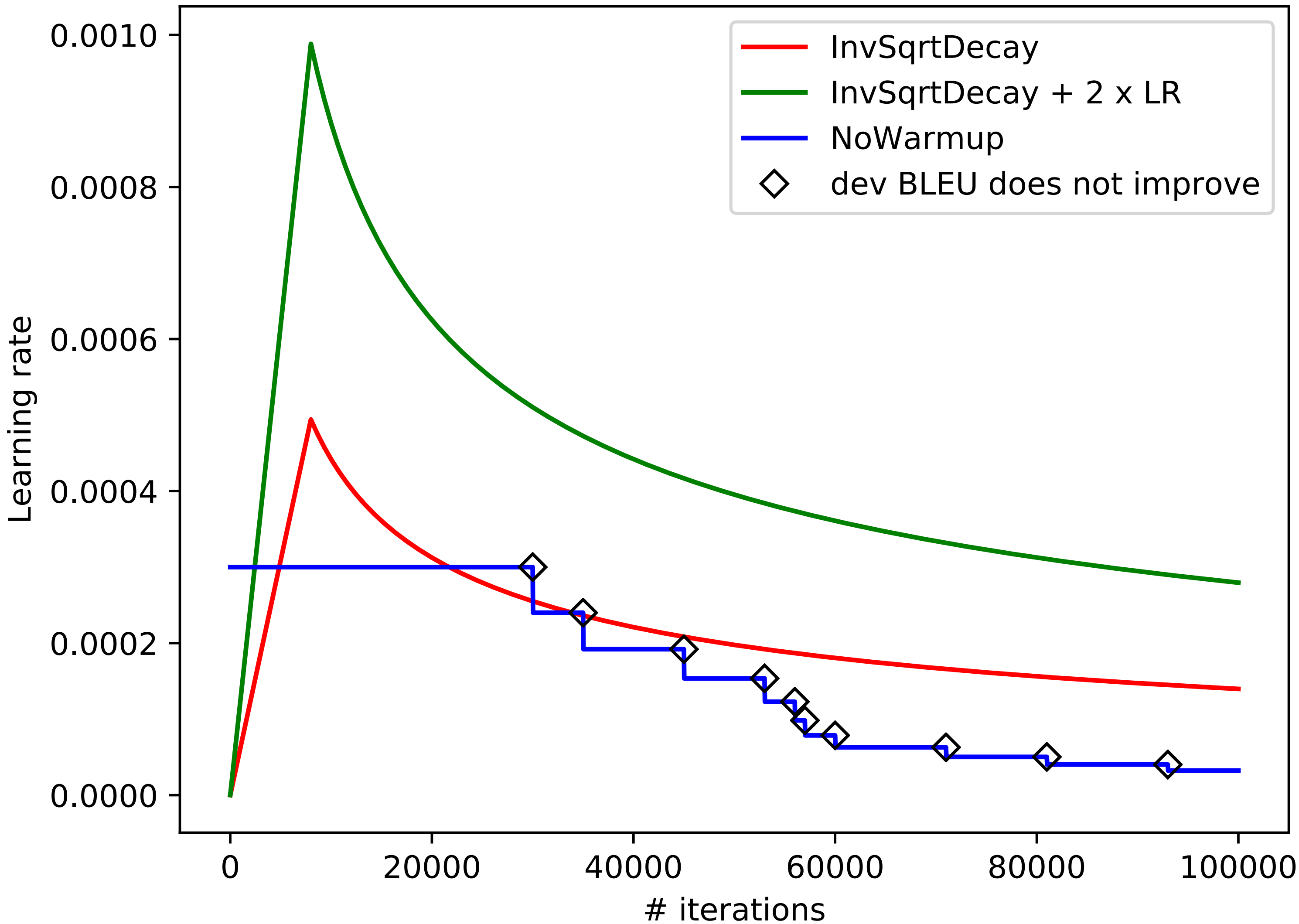
Do we really need warmup?

Does the good old “decay when dev BLEU doesn’t improve” still work?

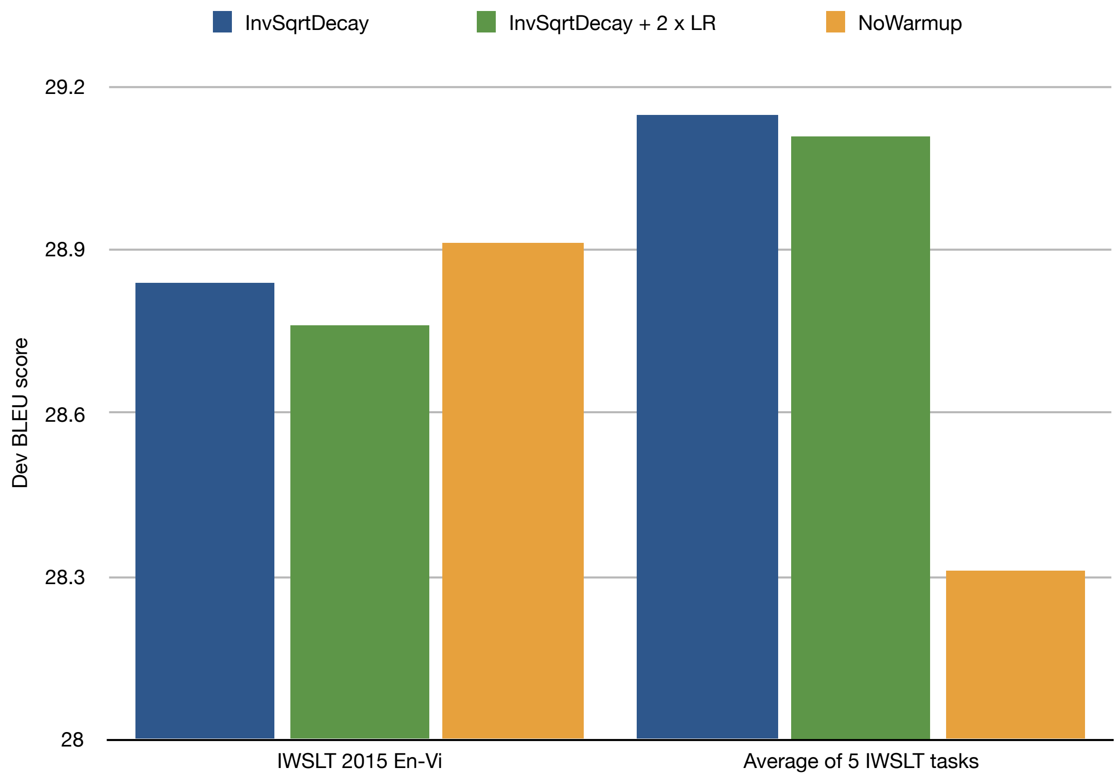
Do we really need to continuously decay learning rate?

Can we train low-resource, small batch size (4096 tokens/batch) with very high learning rate?

Experiments: Learning rate



Experiments: Learning rate



We can often get away without warmup

Warmup is still useful

Can use large learning rate even with small batch size

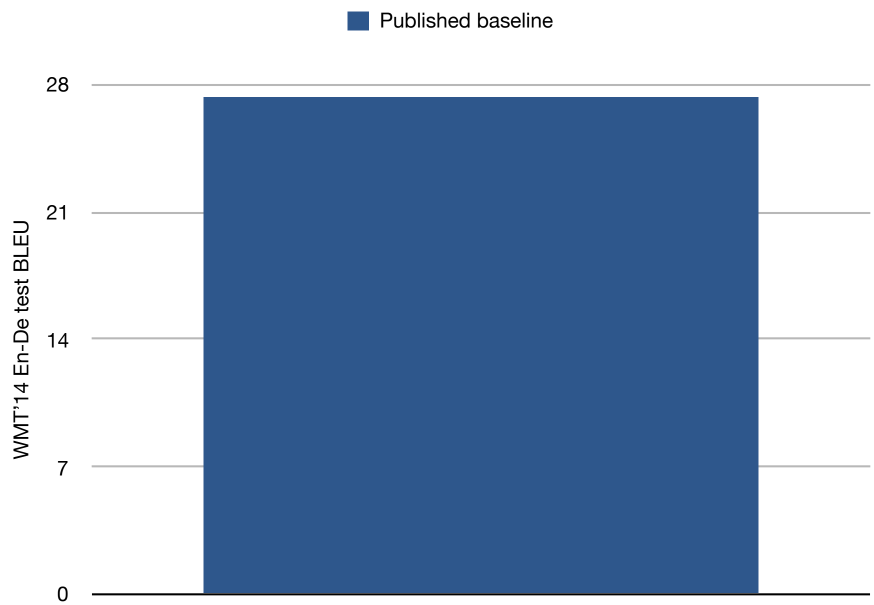
Experiments: PreNorm vs PostNorm again

	4 layers	5 layers	6 layers
PostNorm	18.31	fails	fails
PreNorm	28.33	28.13	28.32

Table 5: Development BLEU on *en*→*vi* using NoWarmup, as number of encoder/decoder layers increases.

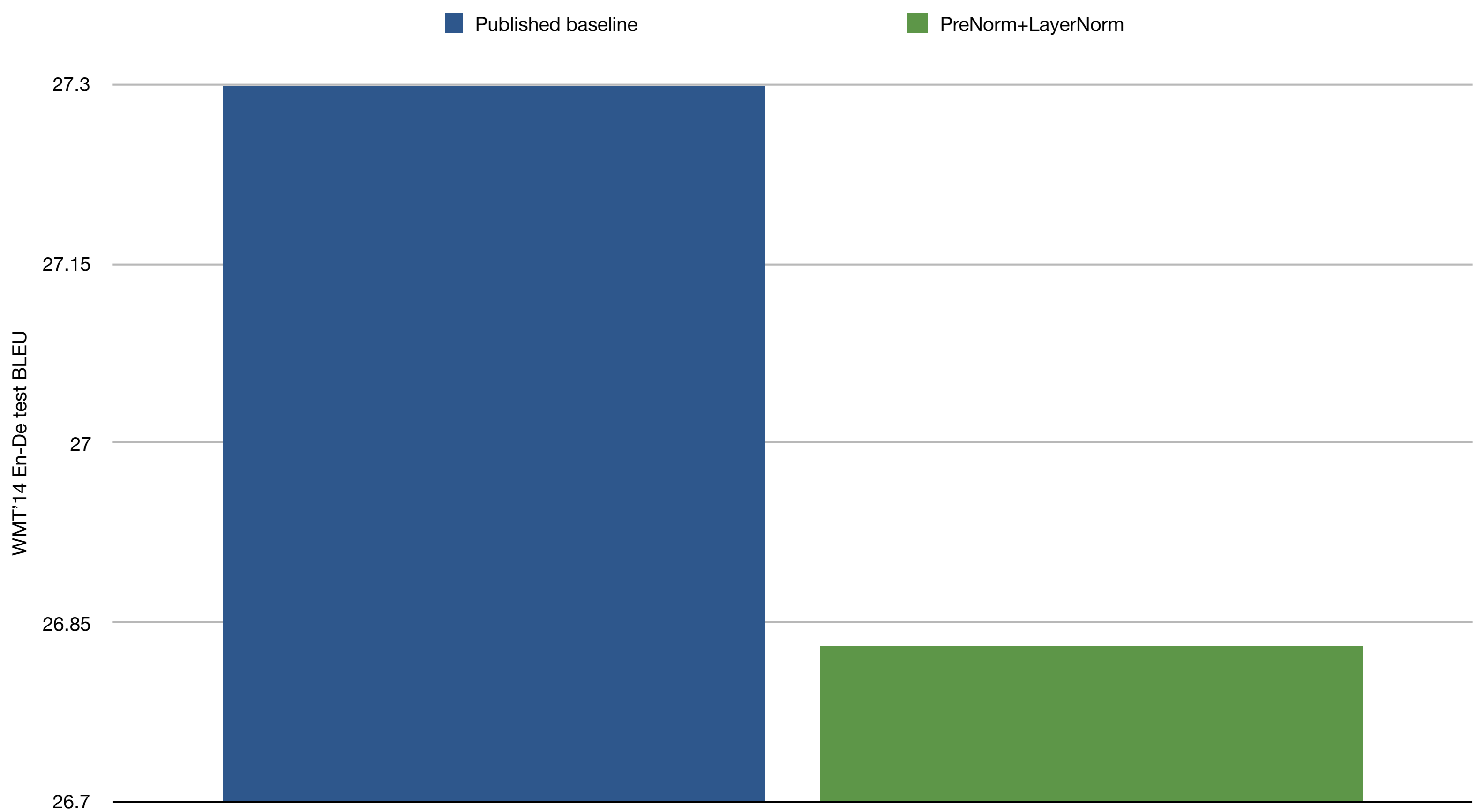
Can SmallInit help PostNorm without warmup? NO!

High resource: a different story

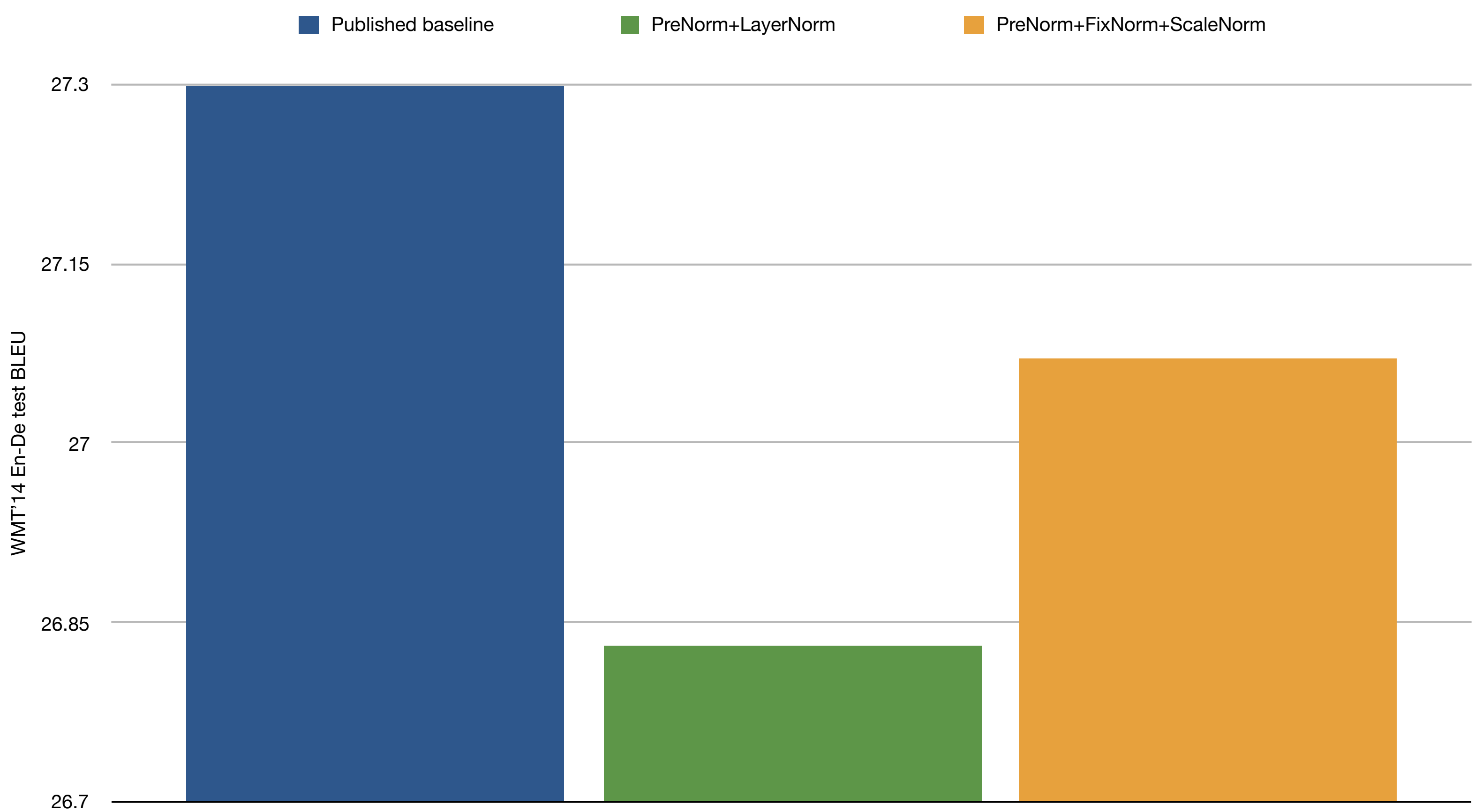


We use the standard high-resource baseline WMT 2014 En-DE

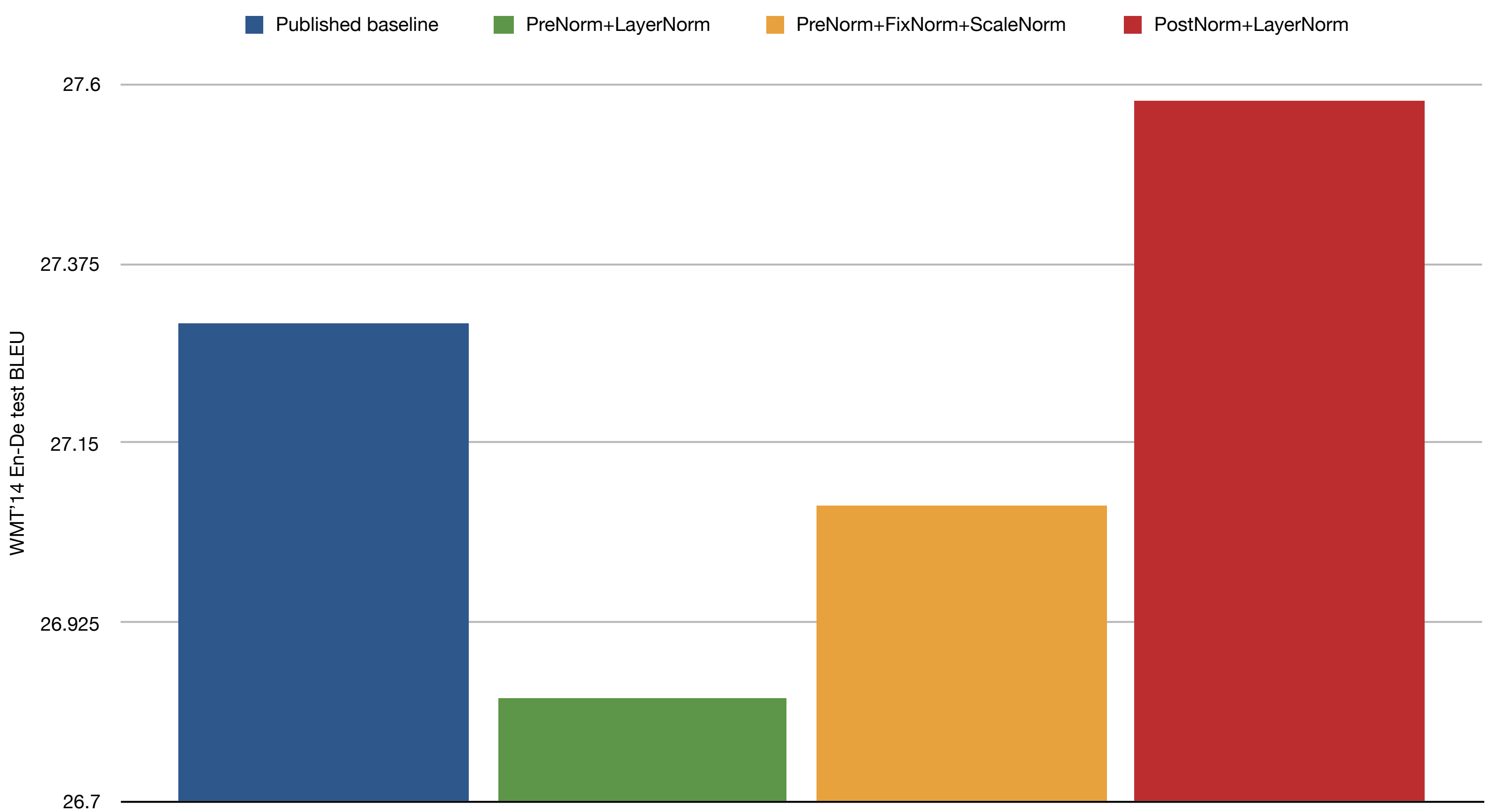
High resource: a different story



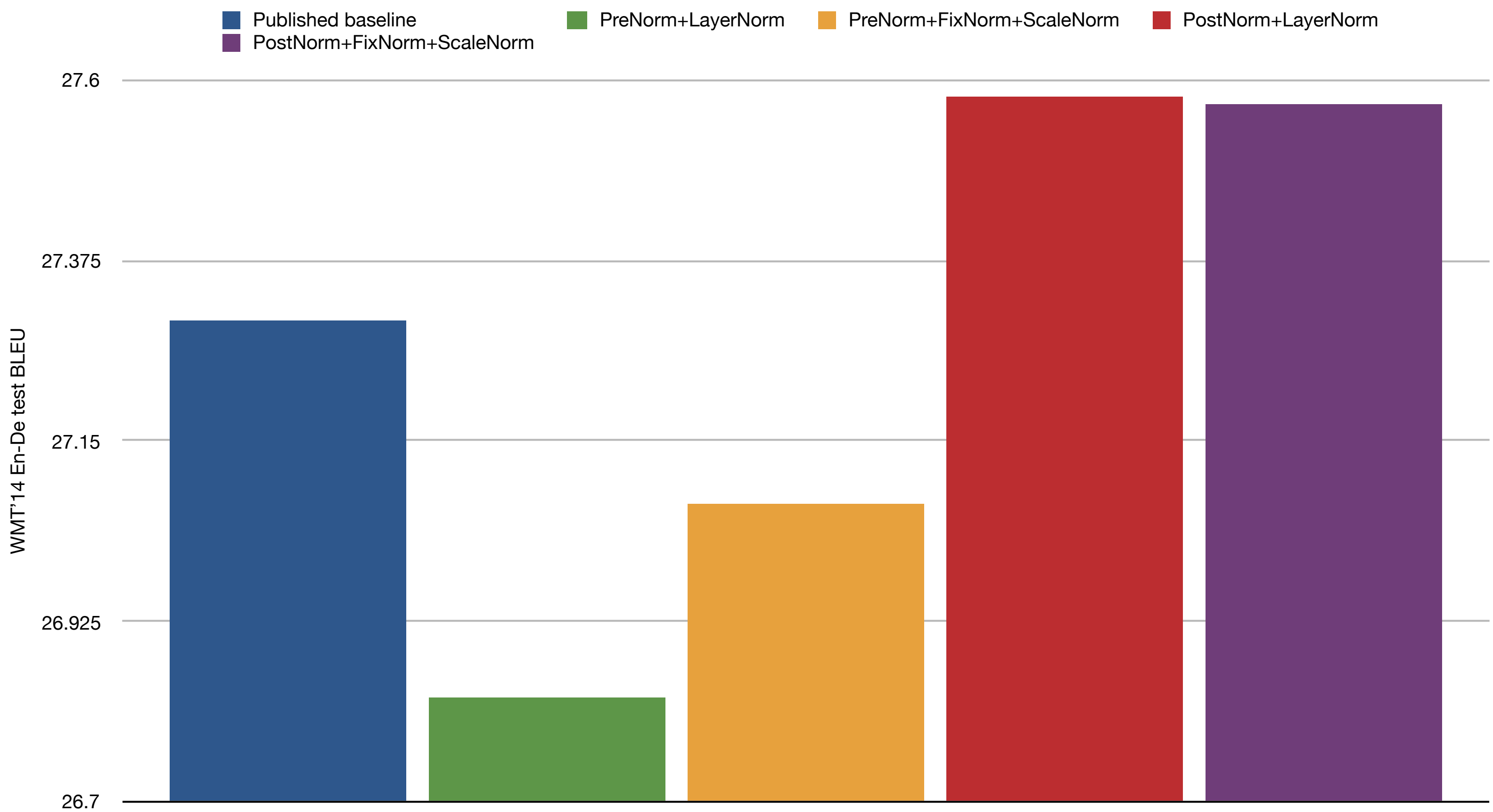
High resource: a different story



High resource: a different story



High resource: a different story



High resource:
a different story

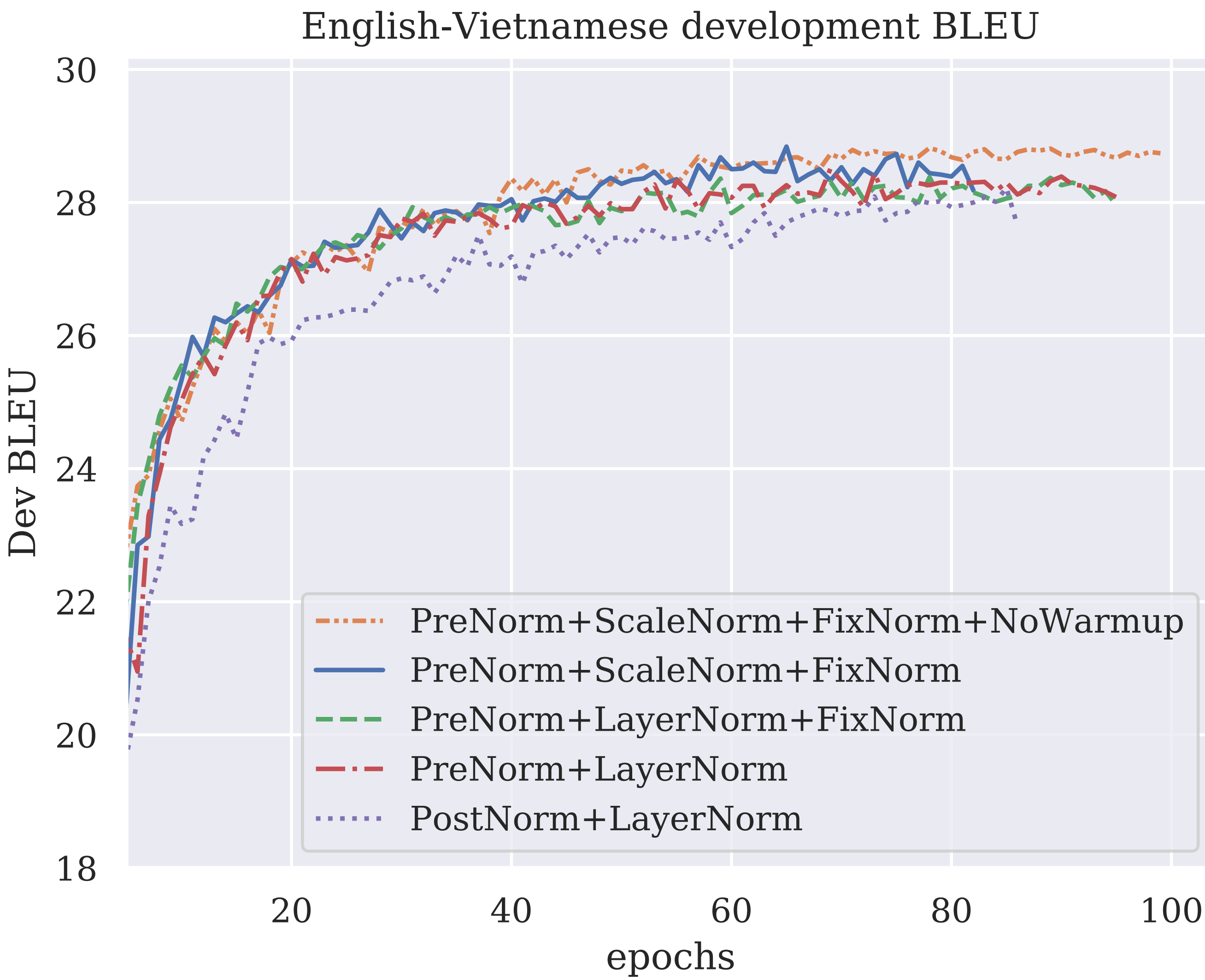
High-resource often uses large batch size which has more stable gradients. This could help solving the instability problem.

ScaleNorm + FixNorm achieves comparable result

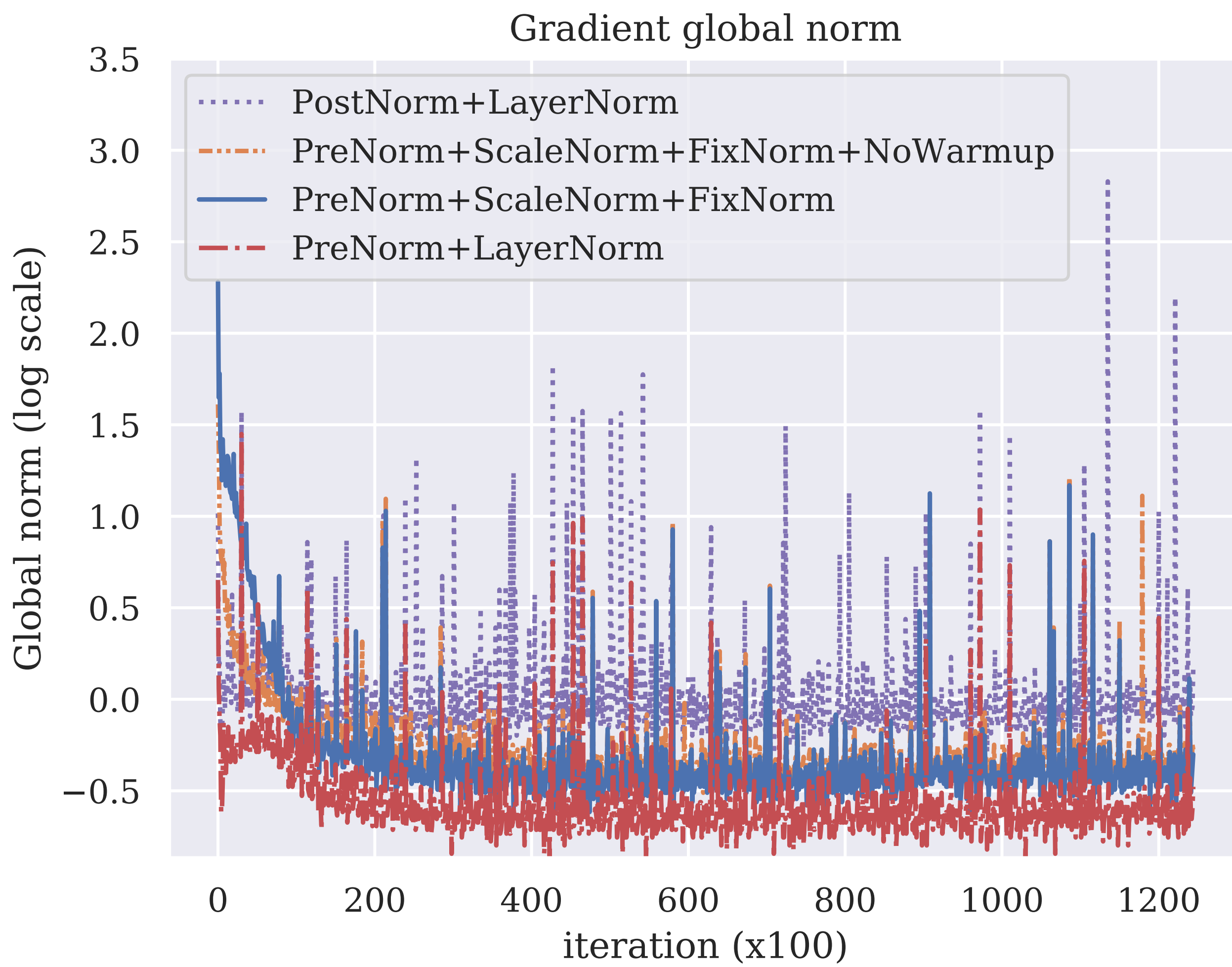
ScaleNorm is faster than LayerNorm

We recommend to always replace LayerNorm with ScaleNorm+FixNorm

Analysis


English-Vietnamese development BLEU
Dev BLEU
epochs
30
28
26
24
22
20
18
20
40
60
80
100
PreNorm+ScaleNorm+FixNorm+NoWarmup
PreNorm+ScaleNorm+FixNorm
PreNorm+LayerNorm+FixNorm
PreNorm+LayerNorm
PostNorm+LayerNorm

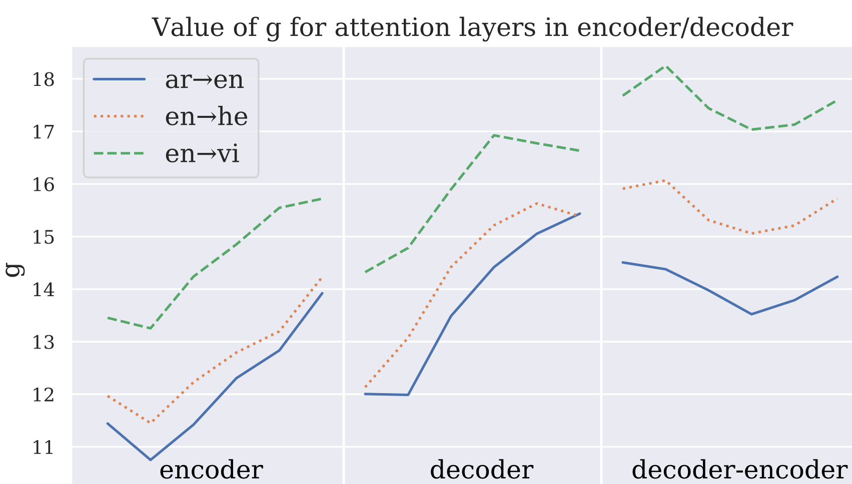
Analysis

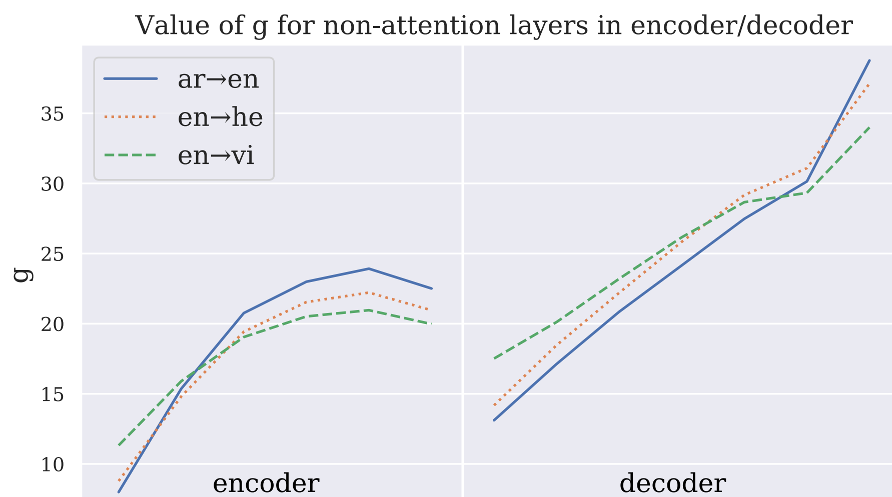


Gradient global norm
PostNorm+LayerNorm
PreNorm+ScaleNorm+FixNorm+NoWarmup
PreNorm+ScaleNorm+FixNorm
PreNorm+LayerNorm
Global norm (log scale)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000
1200
iteration (x100)

Analysis: learned values of g(s)

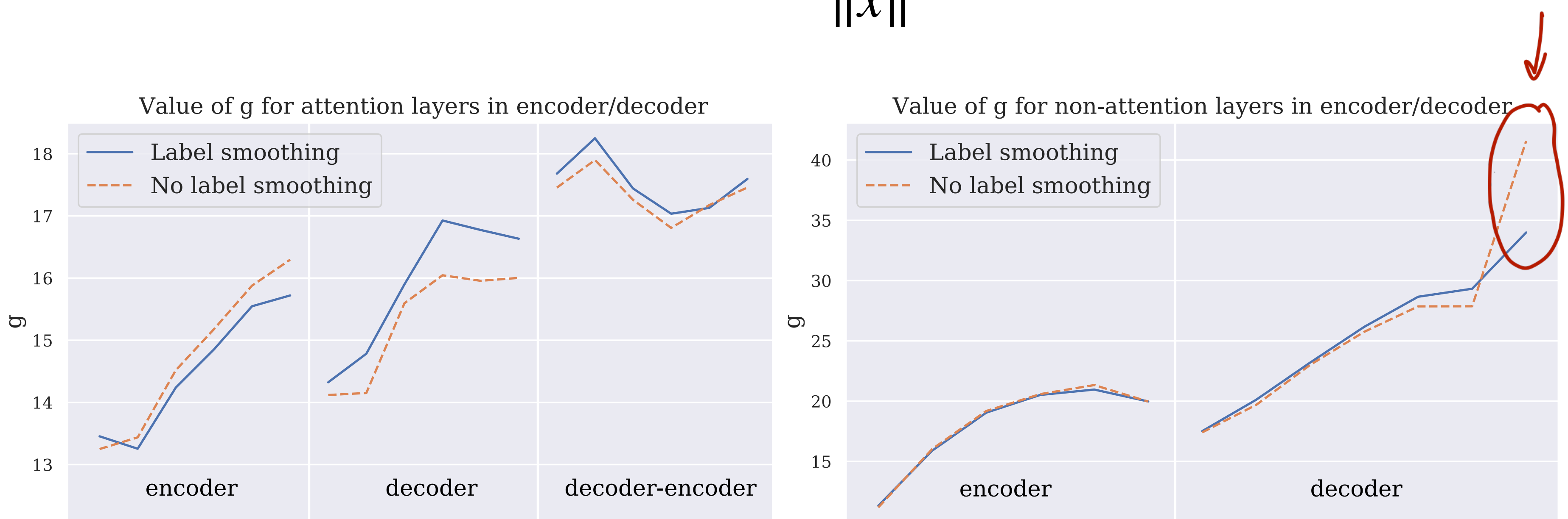
$$\bar{x} = g \frac{x}{\|x\|}$$





Analysis:
label smoothing vs no label smoothing

$$\bar{x} = g\frac{x}{\|x\|}$$



Conclusions

We propose 3 changes to Transformer: PreNorm + FixNorm + ScaleNorm

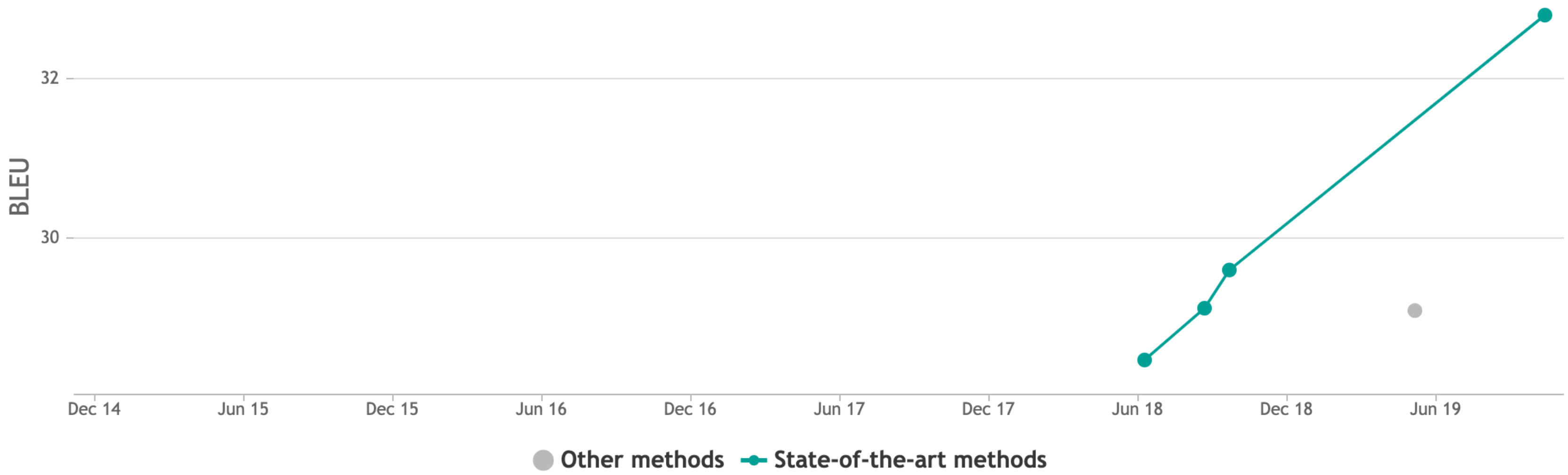
Significantly improve low-resource NMT

Comparable on high-resource NMT (FixNorm+ScaleNorm)

Faster

SOTA on IWSLT 2015 En-Vi

Machine Translation on IWSLT2015 English-Vietnamese



Edit

RANK	METHOD	BLEU	PAPER TITLE	YEAR	PAPER	CODE
1	Transformer+BPE+FixNorm+ScaleNorm	32.8	Transformers without Tears: Improving the Normalization of Self-Attention	2019		

https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english-1

Questions?

Thanks for listening ☺️

paper: https://arxiv.org/pdf/1910.05895.pdf

code: https://github.com/tnq177/transformers_without_tears

References

1. “Attention is all you need”, Vaswani et al., 2017
2. “Adafactor: Adaptive Learning Rates with Sublinear Memory Cost”, Shazeer and Stern, 2018
3. “Training tips for the Transformer model”, Popel and Bojar, 2018
4. “The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation”, Chen et al., 2018
5. “Identity Mappings in Deep Residual Networks”, He et al., 2016
6. “Learning Deep Transformer Models for Machine Translation”, Wang et al., 2019
7. “On Layer Normalization in the Transformer architecture”, Anonymous, 2019
8. “Stabilizing Transformers for Reinforcement Learning”, Parisotto et al., 2019
9. “The Sockeye neural machine translation toolkit”, Hieber et al., 2018
10. “Tensor2Tensor for Neural Machine Translation”, Vaswani et al., 2018
11. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”, Ott et al., 2019
12. “Improving Lexical Choice in Neural Machine Translation”, Nguyen and Chiang, 2018
13. “Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift”, Ioffe and Szegedy, 2015
14. “How does batch normalization help optimization?”, Santurkar et al., 2018
15. “Layer Normalization”, Ba et al., 2016

References

- 16. “Root Mean Square Layer Normalization”, Zhang and Sennrich, 2019
- 17. “Effective approaches to attention-based neural machine translation”, Luong et al., 2015
- 18. “Deep Sparse Rectifier Neural Networks”, Glorot et al., 2011
- 19. “Deep Residual Learning for Image Recognition”, He et al., 2015
- 20. “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”, Veit et al., 2016
- 21. “When and Why are pre-trained word embeddings useful for Neural Machine Translation”, Ye et al., 2018