LayerNorm是Transformer的最优解吗？

原创 rumor酱 夕小瑶的卖萌屋 2020-04-07 22:28:54 手机阅读

来自专辑
卖萌屋@自然语言处理

一只小狐狸带你解锁 **炼丹术&NLP** 秘籍

前言

众所周知，无论在CV还是NLP中，深度模型都离不开归一化技术（Normalization）。在CV中，深度网络中一般会嵌入批归一化（BatchNorm，BN）单元，比如ResNet；而NLP中，则往往向深度网络中插入层归一化（LayerNorm，LN）单元，比如Transformer。

为什么在归一化问题上会有分歧呢？一个最直接的理由就是，**BN用在NLP任务里实在太差了（相比LN）**，此外，**BN还难以直接用在RNN中**，而RNN是前一个NLP时代的最流行模型。

虽然有大量的实验观测，表明NLP任务里普遍BN比LN差太多，但是迄今为止，依然没有一个非常严谨的理论来证明LN相比BN在NLP任务里的优越性。甚至，连BN自身为什么work的问题都一直存在争议。

早期对BN有效性的解释是其有助于缓解神经网络“**内部协方差漂移**”（Internal Covariance Shift，ICS）问题。即，后面的层的学习是基于前面层的分布来的，只有前面一层的分布是确定的，后面的层才容易学习到有效的模式，然而，由于前面的层的分布会随着batch的变化而有所变动，导致了后面的层看来“前面一直在动，我无法安心学习呀”。

而BatchNorm这类归一化技术，**目的就是让每一层的分布稳定下来**，让后面的层可以在前面层的基础上安心学习知识。顾名思义，BatchNorm就是通过对batch size这个维度归一化来让分布稳定下来。LayerNorm则是通过对Hidden size这个维度归一化来让某层的分布稳定。

然而，后来也有一些研究diss了这个解释，说这个解释是错误或不充分的（incorrect/incomplete）[1]，近期也有一些研究[2][3]表明BN之所以有助于训练深度神经网络，是因为它可以让loss曲面变得更加平滑。Anyway，这依然是一个未完全解开的老谜。

除了BN之外，LN也有同样的“为什么work”的终极问题。研究[4]表明，LN在反向时有助于梯度的归一化。也有研究[5][6]表示LN的主要作用是在训练初期缓解梯度消失和爆炸的问题，提升稳定性。

所以说，BN和LN本身的作用机理都没有完全搞清楚，自然也很难去证明为什么BN在NLP数据上就不work，LN就更work。

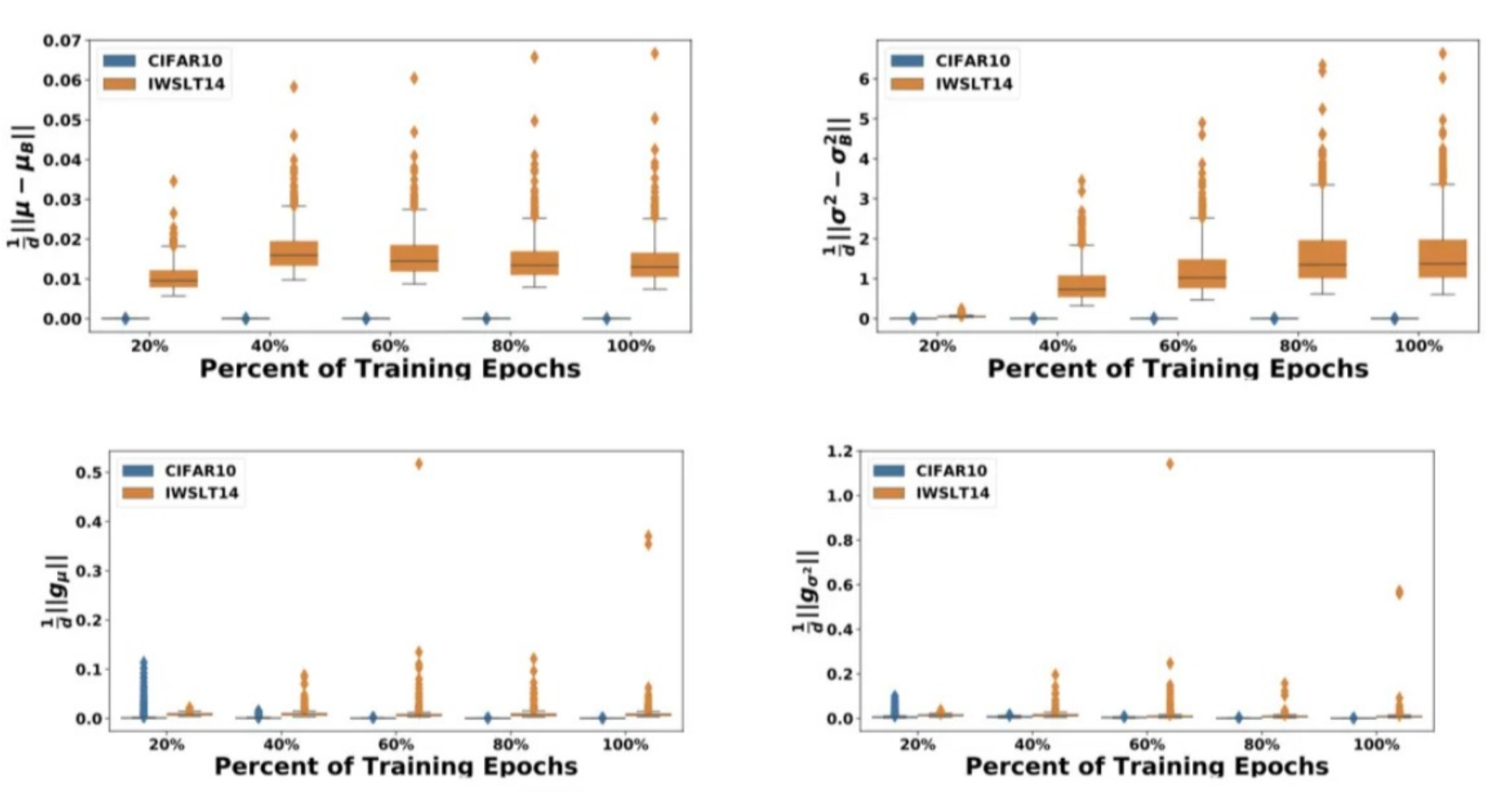
不过，近期小夕无意间刷到了一篇UC Berkeley的《Rethinking Batch Normalization in Transformers》[7]，发现了一个比较有趣的实验结论，并基于这个观测，作者提出了一种针对NLP data（确切说是Transformer）改进的新的归一化方法，叫幂归一化（PowerNorm）。

后台回复【0407】获取论文PDF噢～

强上BN后的Transformer

作者这里做了一个实验，为BN在NLP data（Transformer）上不work提供了一个更加微观的观测证据。

首先，作者将Transformer中的LN都替换成了BN，然后在CV和NLP两个任务上观测BN中的两个统计量（即均值μ和方差σ^2）及其他们的梯度g_μ和g_{σ^2}在训练过程中的稳定程度。



上图中，蓝色是ResNet20在Cifar-10做图像分类的结果，橙色是Transformer+BN在IWSLT14做翻译的结果。X轴是训练时间，Y轴是基于batch的统计值和它对应的移动平均值的欧式距离。

可以看到，ResNet20在Cifar-10任务上统计量的震荡很小，而使用BN的Transformer不仅震荡剧烈，还有很极端的异常值，这会导致μ和σ的统计不准确，**造成train/test不一致，预测效果下降**。

基于这个有趣的观测结果，作者这里针对性的提出了两点改进，并将改进后的BN称之为幂归一化（PowerNorm，PN）。

PowerNorm

1. PN-V

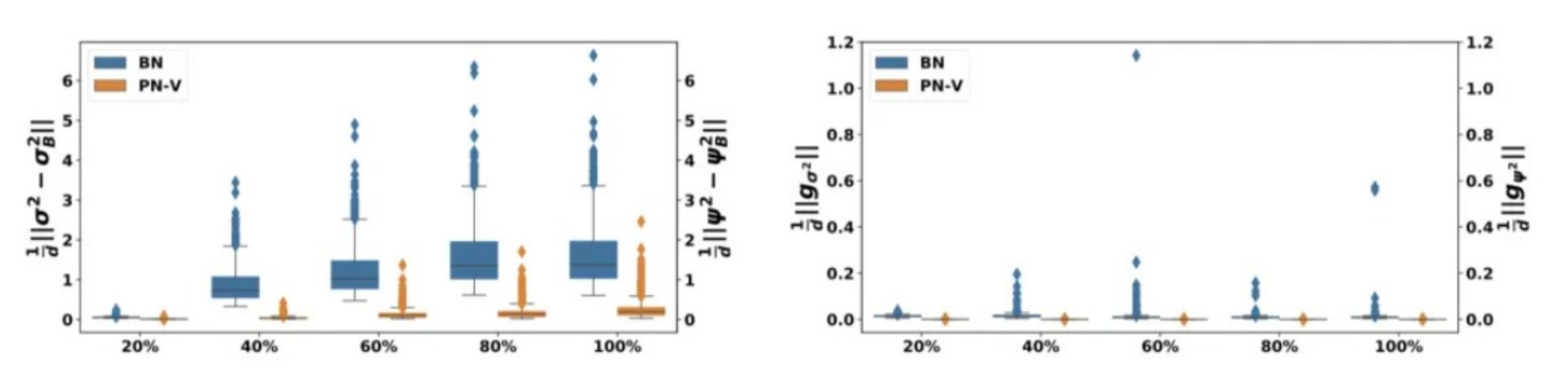
BN强制将数据转换成均值为0方差为1的正态分布，但在数据本身均值方差剧烈震荡的情况下，强制移动均值会起到不好的效果。因此作者提出了新的scale方式，只强制数据有**unit quadratic mean**：

$$\psi_B^2=\frac{1}{B}\sum x_i^2$$
$$\hat{X}=\frac{X}{\psi_B}$$
$$Y=\gamma\cdot\hat{X}+\beta$$

这样针对batch的前向只需一个统计量，反向也简化成g_{ψ^2}一个梯度：

$$\frac{\partial\mathcal{L}}{\partial x_i}=\frac{1}{\psi_B\gamma}\frac{\partial\mathcal{L}}{\partial y_i}-\frac{1}{\psi_B\gamma B}\sum_{j\in B}(\frac{\partial\mathcal{L}}{\partial y_j}\hat{x}_i\hat{x}_j)$$

对比新的ψ（橙色）和之前的σ（蓝色），发现震荡明显减小：



2. Running Statistics in Training

从PN-V的改进可以看到，虽然震荡减少了很多，但还是有很多异常值。因此作者改用移动平均的方式计算ψ：

$$\hat{X}_t=\frac{X_t}{\psi_{t-1}}$$
$$Y_t=\gamma\cdot\hat{X}_t+\beta$$
$$\psi_t^2=\alpha\psi_{t-1}^2+(1-\alpha)\psi_B^2$$

但使用移动平均的话，在求梯度时无法对之前所有ψ求导，因此作者用当前batch的统计量去近似，感兴趣的同学可以看下论文中的推导。

与LN的比较

虽然如前所述，难以说清楚在NLP data上LN比BN优越在哪里，但是是容易说清楚PN对BN的优越性的（毕竟PN的诞生就是基于BN在NLP data上的实验观测）。

那么问题来了：PN和LN哪个更有效？自然也没法直接在理论层面上进行比较，所以作者跑了一把实验，分别尝试了机器翻译和语言模型任务：

Model	IWSLT14 small	WMT14 big
Transformer [37]	34.4	28.4
DS-Init [48]	34.4	29.1
Fixup-Init [49]	34.5	29.3
Scaling NMT [27]	/	29.3
Dynamic Conv [40]	35.2	29.7
Transformer + LayerDrop [6]	/	29.6
Pre-Norm Transformer$_{\text{LN}}$	35.5	29.5
Pre-Norm Transformer$_{\text{BN}}$	34.4	28.1
Pre-Norm Transformer$_{\text{PN-V}}$	35.5	28.5
Pre-Norm Transformer$_{\text{PN}}$	**35.9**	**30.1**

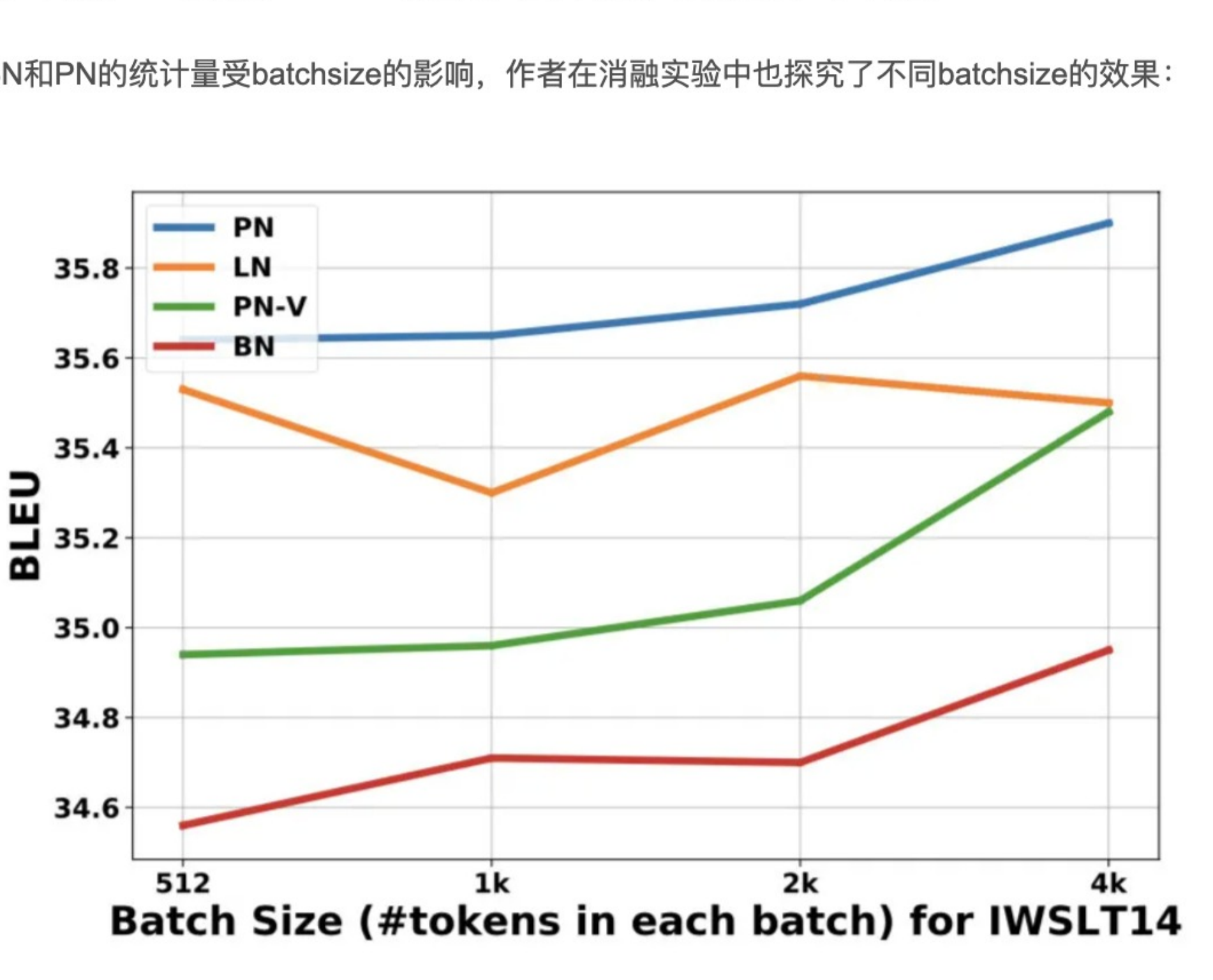
Table I: *MT performance (BLEU) on IWSLT14 De-En and WMT14 En-De testsets. Using* PN-V *instead of BN significantly improves the performance, but LN still outperforms. However,* PN *achieves much higher BLEU scores, as compared to LN.*

Model	PTB	WikiText-103
	Test PPL	Test PPL
Tied-LSTM [13]	48.7	48.7
AWD-LSTM-MoS [44]	56.0	29.2
Adaptive Input [2]	57.0	20.5
Transformer-XL$_{\text{base}}$ [4]	54.5	24.0
Transformer-XL$_{\text{large}}$ [4]	–	18.3
Tensor-Transformer$_{\text{1core}}$ [21]	57.9	20.9
Tensor-Transformer$_{\text{2core}}$ [21]	49.8	18.9
Tensor-Transformer$_{\text{1core}}$ + LN	53.2*	20.9*
Tensor-Transformer$_{\text{1core}}$ + BN	60.7	27.2
Tensor-Transformer$_{\text{1core}}$ + PN-V	55.3	21.3
Tensor-Transformer$_{\text{1core}}$ + PN	**47.6**	**17.9**

Table II: *Results with state-of-the-art methods on PTB and WikiText-103. '-' indicates no reported results in that setting, '*' indicates the results are from our own implementation.* PN *achieves 5.6/3 points lower testing PPL on PTB and WikiTest-103, respectively, compared to LN.*

对于上述结果，小夕也去paperwithcode网站查了一下，目前IWSLT14的SOTA是36.3，论文中的35.9可以排在第二的位置；WMT14 En-De的SOTA是35，论文中的30.1可以排在第五的位置；WikiText-103的SOTA是10.8，论文的结果排在第八名第位置。由于作者没有做其他优化，看起来总体结果还是不错的～当然，PN在其他NLP data和任务上是否有效，还有待进一步验证。

由于BN和PN的统计量受batchsize的影响，作者在消融实验中也探究了不同batchsize的效果：



可以看到PN在整体上还是优于LN的。

总结

由于深度学习的不可解释性，归一化方法在网络中真正的作用和优劣一直是个谜。本文针对BN提供了一个新的研究角度，通过对统计量及梯度的稳定性观测，找到了BN为什么在NLP问题上不work的其中一个原因，即**数据分布的震荡和异常值导致train/test不一致**。基于该观测证据，作者对BN进行了对应的改进，提出了更适合NLP data的幂归一化PowerNorm，得到了优于原生BN的效果，且在部分任务上超过了LN的表现。

另外，看到这里后，相信会有很多小伙伴会关心BERT+PN的效果。燃鹅众所周知，要复现BERT的预训练过程是非常不可描述的，所以作者这里没有给出相应实验，也是合乎情理的。

所以目前结构创新都不会用BERT去验证，机器翻译和LM任务确实是常规benchmark。是否真正有用，可能需要慢慢被大家用起来才知道。

后台回复【0407】获取论文PDF噢～

参考文献

[1] *Ali Rahimi. Nuerips 2017 test-of-time award presentation, December 2017: https://www.zachpfeffer.com/single-post/2018/12/04/Transcript-of-Ali-Rahimi-NIPS-2017-Test-of-Time-Award-Presentation-Speech*
[2] *How does batch normalization help optimization?: https://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf*
[3] *PyHessian: Neural networks through the lens of the Hessian.: https://arxiv.org/pdf/1912.07145.pdf*
[4] *Understanding and Improving Layer Normalization: https://arxiv.org/abs/1911.07013*
[5] *Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention: https://arxiv.org/abs/1908.11365*
[6] *Fixup Initialization: Residual Learning Without Normalization: https://arxiv.org/abs/1901.09321*
[7] *Rethinking Batch Normalization in Transformers: https://arxiv.org/abs/2003.07845*
[8] 详解深度学习中的*Normalization，BN/LN/WN: https://zhuanlan.zhihu.com/p/33173246*