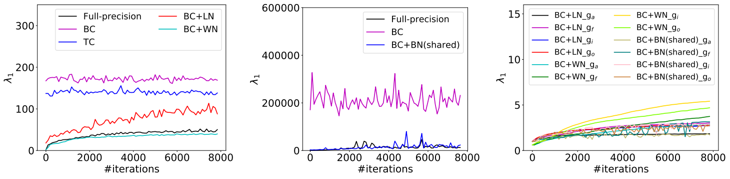
Reviewer 1 **“using normalization to improve the accuracy loss caused by quantization is not so impressive"**: Our main contribution is on the theoretical analysis of quantized LSTM training and normalization. On the empirical results, Tables 2-4 show that accuracy loss due to quantization can be significant (especially for BinaryConnect and TerConnect), and normalization helps to recover performance, sometimes to the level comparable with the full-precision baseline.

"λ_1 values ... before and after quantization without normalization": Figure(a) shows λ_1 values for full-precision LSTM, binarized (using BinaryConnect (BC)), and ternarized (using TerConnect (TC)) LSTMs without normalization, on character language modeling with *Penn Treebank*. As can be seen, after BC and TC, λ_1 is much larger.

"g and λ_1 values ... before and after quantization with normalization": Figure(a) shows λ_1 values for the binarized LSTM with weight/layer normalization. Figure(b) shows λ_1 values with batch normalization.[1] As can be seen, normalization makes λ_1 in binarized LSTM much smaller. Finally, Figure(c) shows the corresponding g values. Because of lack of space, results for the ternarized LSTM are not shown.



(a) λ_1 in Propositions 2.1, 3.1-3.2. (b) λ_1 in Propositions 3.3-3.4. (c) g.

precision	quantization	normalization	MNIST	pMNIST	size
full	-	-	98.9	90.2	159
1-bit	BWN	-	98.6	89.7	8
		weight	98.7	**91.3**	11
		layer	**98.8**	90.8	14
		batch (shared)	20.6	40.1	21
		batch (separate)	98.6	91.1	4914
2-bit	TWN	-	98.6	90.4	13
		weight	98.6	92.1	16
		layer	**98.8**	91.7	18
		batch (shared))	26.5	38.3	25
		batch (separate)	98.7	**93.1**	4919

“sequential MNIST task, the batch normalization (shared) method totally failed": In this task, each time step corresponds to an input pixel. The use of shared batch normalization statistics implicitly assumes different pixels have similar characteristics. However, this may not be reasonable (e.g., pixels around the edge are typically darker).

“ quantization with scaling factors": The table above adds BWN/TWN results on sequential MNIST task (Table 4). The weight/layer normalized quantized LSTMs have comparable results as full-precision baselines, but much smaller.

“conclusions still hold in GRU?": This is an open issue, as analysis for GRU is different. We leave this as future work.

“How about in multilayer LSTMs?": Below we add results on 2-layer LSTM for *Penn Treebank* task (first number: test PPL, second: size(KB), “no" means no normalization). Again, BinaryConnect/TerConnect fail when directly used, but achieve results comparable as the full-precision baseline when normalized, while being much smaller in size.

Full-precision: no(81.67, 26427) weight(81.82, 26467) layer(81.02, 26508) batch-shared(80.60,26589) batch-separate(81.42,29352)
BinaryConnect: no(134.47, 846) weight(81.86, 886) layer(82.10, 927) batch-shared(80.55, 1008) batch-separate(81.48, 3771)
TerConnect: no(639.61,1671) weight(80.54, 1711) layer(80.06, 1752) batch-shared(79.25, 1833) batch-separate(79.89, 4596)

Reviewer 2 **“simple application of batch normalization works better or similar to SBN?"**: Yes.

“In [1] the authors claim that the size of their network is 5 KBytes, while in the Table4, the SBN size is 5526KBytes?": As mentioned in line 185-188, [1] does not count the additional storage for the full-precision mean and standard deviation statistics (in batch normalization) at the T time steps. For the sequential MNIST task, $T = 784$, and these statistics are in fact even much larger than the model itself. The unnormalized binary model in our Table 4 has size 8 KBytes. This is slightly larger than 5 KBytes because we do not quantize $\mathbf{W}_{x*}$ (Appendix A.3). For detailed model size analysis, please see Remark 3.2.

"propositions on upper bounds are quite straight-forward": LSTM, due to introduction of c_t, is more difficult to analyze than vanilla RNN. Adding normalization makes the analysis even more non-trivial. For example, in batch normalization, samples in a minibatch become related and the gradient flow is intertwined in different dimensions (please see steps 1-3 in Appendix D.8). Also, we are the first to derive these upper bounds to analyze why quantized LSTM is difficult to train, and how normalization helps.

“experimental results are simple combinations of previous works": In the experiments, we thoroughly run and study various 1-bit and 2-bit quantization methods with different normalizations (weight/layer/batch) on different tasks.

Reviewer 3 **“λ_2 typically not zero"**: We expect a nonzero λ_2 will make gradient explosion happen more easily (lines 90-91).

“only made on the upper bound of the gradient magnitude": We agree that a lower bound will also be useful. However, even for vanilla RNN, only an upper bound can be derived in (Pascanu et al., 2012).

“effect of all the normalization scaling parameters ‘g’": Please refer to our third reply to Reviewer 1.

"larger network consisting of multiple LSTM layers" : Please refer to our last reply to Reviewer 1.

[1] The bounds in Propositions 3.3 and 3.4 are based on the squared weight norm. Hence, Figure(b) plots the coefficient before $\sum_{k=1}^{N} \| \frac{\partial \xi_m}{\partial \mathbf{h}_t^k} \|^2$. With an abuse of notation, we still call this value λ_1.