Dear Reviewers:

Thanks for all your insightful comments and constructive suggestions. We will correct all typos in our final version. We first respond to a common concern:

About generality of RMSNorm for different downstream tasks, model architectures, and initializations: We mainly experiment on language-related tasks, because this is where the use of LayerNorm is most widespread. However, note that our experiments show the effectiveness of RMSNorm on heterogeneous architectures and initializations, covering different RNN variants and self-attentional models, and various activations (such as sigmoid, tanh, linear and softmax), with initializations ranging from uniform, normal, orthogonal with different initialization ranges or variances. Details can be found in previous work on which we base our comparisons, but we will include more detail to be self-contained.

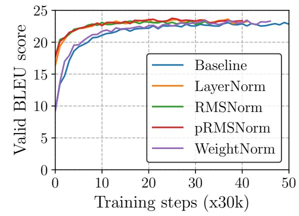

Figure 1: BLEU curve over training steps on newstest2013 devset.

Model	Test Error	Time
Baseline	8.96%	51s
BatchNorm	8.25%	66s
WeightNorm	8.28%	53s
LayerNorm	10.49%	72s
RMSNorm	8.83%	61s (**15%**)

Table 1: Test error and time (sec) per training epoch on CIFAR-10 classification task. The speedup of RMSNorm over LayerNorm is shown in bracket.

In addition, we also experiment on the CIFAR-10 classification task. We train a modified version of the ConvPool-CNN-C architecture, and follow the same experimental protocol as in the WeightNorm paper [20] using their public source code. LayerNorm is applied to the width and height dimensions of image representation. We perform gain scaling and bias shifting on the channel dimension. Our results (Table 1) show that RMSNorm outperforms Baseline and LayerNorm in test error, and achieves 15% speed-up over LayerNorm, though it underperforms the BatchNorm and WeightNorm.

Comparison with weight normalization: We performed experiments with RNNSearch, using the WeightNorm implementation provided by the base toolkit (Theano-version Nematus). Results in Figure 1 show that WeightNorm converges slower and requires more training steps. In addition, the overall translation quality of WeightNorm on testsets ($21.7/23.5$ on Test14/Test17, respectively) underperforms those of LayerNorm and (p)RMSNorm. We also attempted integrating WeightNorm into pytorch-based RNNSearch using the official API (*nn.utils.weight_norm*), but this led to out-of-memory problems.

= *To R3:* We will include the recent discussion on internal covariate shift in our final version. The scalar notation in (2) follows LayerNorm paper [3], and we will change (1) to make the whole paper consistent. By "1%" in Fig 3, it actually means 10%. In Table 7, "OE[30]" denotes the original results reported by [30]. [3] reproduce their work ("OE[3]"), and add LayerNorm ("OE+LayerNorm[3]") to demonstrate LayerNorm's effectiveness. All these numbers are from existing work, and other numbers are from our own experiments. We will make this clear in our final version.

= *To R4:* Please see the above common response.

= *To R5: On l_p norm:* We didn't experiment with all choices of p for l_p norm, but we experimented with l_2 norm for RNNSearch. Results in Fig. 2 and Table 2 show that L2Norm does not work well in terms of both convergence and final translation quality.

On optimizer hyperparameters: For NMT model, we adopt Adam optimizer. The RNNSearch model is trained with an initial learning rate of 10^{-4}, which is half-decayed if no improvement is observed on devset. The learning rate for Transformer is adapted according to Eq. (3) in paper [29] with a warmup step of 4000. We adopt the base setting. We will include these details in the final version.

On mean-centering and weight initialization: See common response for the range of weight initializations tested; R5 suggests that mean-centering in LayerNorm (which RMSNorm abandons) may make models more robust towards arbitrary weight/bias initializations. We perform an experiment on RNNSearch MT model with tensorflow-Nematus, and change the center of weight initialization to 0.2. Results in Figure 2 show that LayerNorm becomes very unstable with abnormal initialization, but RMSNorm is more robust (both underperform the original initialization). Our empirical evidence so far suggests that RMSNorm is similarly robust as LayerNorm, or more.

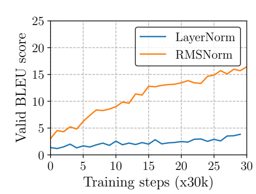

Table 2: BLEU curve of LayerNorm and RMSNorm on devset when initialization center is around 0.2.

c. Error bars for reported accuracies and timing numbers We perform only a single full training run for each of the ≈ 30 models due to resource limitations. Note that we *do not* claim RMSNorm is better than LayerNorm in quality, but *comparable*. For the timing numbers, we report the standard deviation of three runs on three different models (for Baseline/LayerNorm/RMSNorm, respectively): 3.4/32.5/11.8 (RNNSearch with tensorflow-Nematus), 6.3/5.7/5.2 (Attentive Reader model) and 0.23/1.31/0.035 (Transformer model; extremely low variance due to use of different computing platform). We will show more details in the final version.