Q1: Both reviewer #4 and reviewer #5 think it is essential to compare the proposed method with Pre-LayerNorm.

A: We added additional experiments to investigate the question on how PLD compares with PreLN? We tried both Pre-LN with the hyperparameters used for training Post-LN (lr=1e-4) and the hyperparameters used for PLD training (lr=1e-3) to address the effect from the choice of hyperparameters. We trained all configurations for the same number of epochs and finetuned following the standard procedure. In both cases, PreLN is 24% slower than PLD, because PreLN still needs to perform the full forward and backward propagation in each iteration. The table below shows the finetuning results on GLUE tasks. When trained with the same hyperparameters as Post-LN, Pre-LN appears to have a much worse GLUE score (80.2) compared with Post-LN (82.1) on downstream tasks. This is because Pre-LN restricts layer outputs from depending too much on their own residual branches and inhibits the network from reaching its full potential, as recently studied in https://arxiv.org/pdf/2004.08249.pdf. When trained with the large learning rate as PLD, PreLN's result have improved to 82.6 but is 0.6 points worse than PLD (83.2), despite using 24% more compute resource. PLD achieves better accuracy than PreLN because it encourages each residual branch to produce good results independently.

Model	RTE (Acc.)	MRPC (F1/Acc.)	STS-B (PCC/SCC)	CoLA (MCC)	SST-2 (Acc.)	QNLI (Acc.)	QQP (F1/Acc.)	MNLI-m/mm (Acc.)	GLUE
BERT (Original)	66.4	**88.9**/84.8	87.1/89.2	52.1	**93.5**	**90.5**	71.2/89.2	**84.6/83.4**	80.7
BERT + PostLN	67.8	88.0/86.0	89.5/89.2	52.5	91.2	87.1	89.0/90.6	82.5/83.4	82.1
BERT + PreLN + Same lr	66.0	85.9/83.3	88.2/87.9	46.4	90.5	85.5	89.0/90.6	81.6/81.6	80.2
BERT + PreLN + Large lr	67.8	86.7/84.5	**89.6/89.1**	54.6	91.9	88.1	89.3/**90.9**	**83.6/83.7**	82.6
Shallow BERT + PreLN + Large lr	66.0	85.9/83.5	89.5/88.9	54.7	91.8	86.1	89.0/90.6	82.7/82.9	81.8
BERT + PreLN + Large lr + Random	68.2	88.2/86.2	89.3/88.8	56.8	91.5	87.2	88.6/90.3	82.9/83.3	82.7
BERT + PreLN + Large lr + TD only	68.2	88.6/**86.7**	89.4/88.9	55.9	91.3	86.8	89.1/90.7	82.7/83.1	82.7
BERT + PreLN + Large lr + PLD (TD + DD)	**69.0**	**88.9**/86.5	**89.6/89.1**	**59.4**	91.8	88.0	**89.4/90.9**	83.1/83.5	**83.2**

Q2: Reviewer #3, #4, #5 ask about a comparison to simpler and alternative schedules.

A: The current schedule is actually simple. Let us elaborate it in a more intuitive way. Along the temporal dimension (TD), we use the standard exponential decay schedule. This is because there are large embedding shifts at the beginning of the training. Intuitively, we do not want to perturb it with layer drop that can destabilize the training. Along the depth dimension (DD), we use a simple linear decay schedule from $p_0 = 1$ for the first Transformer layer, to p_L for the last Transformer layer. This is because upper layers are often getting similar estimations, especially in the later phase of the training. Our schedule induces an adaptive regularization scheme that smoothly increases the difficulty of sampling. This concept of starting-easy and gradually increasing the difficulty of the learning problem has its roots in curriculum learning (https://icml.cc/Conferences/2009/papers/119.pdf) and allows one to train better models.

We have since run more ablation experiments to evaluate the effectiveness of PLD and added results in Table 1. (1) **Shallow BERT + PreLN + Large lr** directly trains a 9-layer BERT without sampling. This configuration underperforms PreLN by 0.8 points and is 1.4 points worse than PLD likely because the model capacity has been reduced by the loss of parameters. (2) **BERT + PreLN + Large lr + Random** drops layers randomly with a fixed ratio (i.e., it has the same compute cost but without any schedule). It is 0.9 points better than shallow BERT under the same compute cost and 0.1 points better than PreLN while being 24% faster, indicating the strong regularization effect from stochastic depth. It is 0.5 points worse than PLD, which demonstrates the benefit of having schedules enabled. (3) **BERT + PreLN + Large lr + TD only (32-bit*)** disables the DD schedule in training. Its GLUE score matches "Random", suggesting that the exponential decay schedule (TD) has similar performance as the fixed constant schedule along the temporal dimension and accuracy gains of PLD is mostly from the DD schedule. However, without the temporal schedule enabled, the model diverges with NaN in the middle of half-precision (16-bit) training and has to switch to full-precision (32-bit) training, slowing down training speed. We adopted the temporal schedule since it is robust and helpful for training stability, retaining similar accuracy while reducing training cost considerably.

Q3: Reviewer #2 and #3 asks if this method be generalized to other tasks? **A:** We currently work on BERT-like model training. But we also think the techniques are likely generalizable across other workloads if they also exhibit the unrolled iterative refinement phenomenon. This would make an interesting future study. || **Q4**: Reviewer #2 asks whether PLD achieves the same std as the baseline training across multiple experiments? **A:** PLD achieves similar std as the baseline across multiple runs. Fig. 13 in the Appendix reports the median and std of finetuning results for all GLUE tasks. || **Q5:** Reviewer #4 asks to show the stability is improved on the smaller and less stable GLUE dataset. **A:** Fig. 13 and Fig.14 in the Appendix have more GLUE results. For example, Fig. 13 and Fig.14 show results on RTE, the smallest dataset in GLUE. Overall, for small datasets such as CoLA, MRPC, and RTE, PLD is more stable and outperforms the baseline with a statistically significant difference. || **Q6:** Reviewer #4 asks about the comparison with ELECTRA. **A:** we think these are two complementary techniques to speedup pre-training. ELECTRA speeds up the pre-training by replacing masked tokens with alternatives sampled from a generator framework and training a discriminator to predict the replaced token, whereas we speed up the pre-training speed through an architectural change and a schedule for layer dropping. Working together, they could reduce the end-to-end pre-training time even further.