

Table 7: GPT-3 1.3B 0-shot evaluation results. The first column is the results of the original OpenAI GPT-3 1.3B model [7]. All the other columns are in the same order as the rows in main paper Tab. 3. OpenAI results are not directly comparable to ours because the training data are different.

Case	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
Train tokens	OpenAI	baseline	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL
	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B
Avg.	47.9	42.5	43.4	43.0	42.3	43.6	43.0	43.7	43.8	41.9	42.7	43.1	42.0	42.6	42.7	42.8
(0) HellaSwag	54.7	51.9	52.3	52.4	51.8	52.7	52.2	54.1	54.3	50.9	52.0	52.9	49.9	50.6	51.6	52.1
(1) LAMBADA	63.6	62.0	61.2	61.7	60.6	61.9	61.1	62.9	62.3	59.8	61.4	62.3	59.5	59.6	61.3	61.7
(2) TriviaQA	19.7	7.0	7.91	7.63	6.66	7.65	6.07	7.9	7.55	6.15	6.46	7.54	5.9	7.2	6.37	7.42
(3) WebQs	4.63	1.38	1.62	2.07	2.56	1.38	2.02	3.15	2.17	2.46	1.67	2.31	1.03	2.26	2.66	3.2
(4) Winogrande	58.7	55.6	59.1	58.2	57.1	58.9	56.9	58.5	58.4	54.9	58.2	59.1	56.6	57.1	57.1	57.5
(5) PIQA	75.1	71.4	71.0	72.1	70.8	71.4	72.1	71.2	71.5	70.7	71.4	72.3	71.4	71.9	70.5	72.0
(6) ARC Challenge	35.5	29.4	29.6	29.3	28.8	30.1	28.9	28.7	30.1	28.5	28.2	29.7	27.2	27.0	28.7	27.6
(7) RACE-h	53.8	53.7	54.3	55.0	54.0	55.2	55.0	54.4	56.4	53.5	53.2	52.7	52.7	53.7	54.1	54.0
(8) ANLI R1	33.4	31.6	33.3	30.7	33.4	33.5	31.6	33.0	31.6	31.6	29.8	31.9	33.0	32.9	32.1	33.7
(9) ANLI R2	33.3	33.7	33.8	32.8	33.0	33.3	32.9	32.5	31.5	30.4	33.2	34.8	31.8	33.9	34.6	33.6
(10) ANLI R3	33.4	33.1	35.2	33.5	33.2	33.3	33.9	33.4	35.2	33.7	35.8	35.3	32.4	34.8	34.9	35.0
(11) OpenBookQA	46.8	32.4	31.8	32.0	31.2	34.0	34.6	34.0	34.0	31.0	33.0	33.8	30.4	32.4	33.6	32.4
(12) RACE-h	40.9	35.2	34.2	35.7	35.3	35.3	34.3	35.4	36.4	34.6	33.9	35.0	34.3	34.2	34.6	34.9
(13) BoolQ	62.4	62.4	63.1	62.5	60.2	62.7	63.6	61.9	63.6	62.0	62.8	61.0	61.2	59.6	61.5	61.9
(14) Copa	77.0	72.0	70.0	75.0	72.0	73.0	77.0	76.0	75.0	71.0	74.0	73.0	72.0	75.0	71.0	71.0
(15) RTE	56.0	54.2	58.1	54.9	52.0	56.0	54.2	55.0	54.5	55.2	54.9	54.2	59.2	55.6	55.2	54.5
(16) WSC	61.5	36.5	42.3	36.5	34.6	43.3	36.5	43.3	40.4	36.5	37.5	36.5	36.5	36.5	37.5	36.5
(17) MultiRC	13.6	1.05	2.1	1.47	3.15	0.944	0.944	0.839	2.41	0.839	0.839	0.839	0.839	1.68	1.05	1.15
(18) ReCoRD	85.2	83.3	83.7	83.5	83.2	83.8	83.3	84.7	84.3	82.8	82.4	84.0	82.5	82.6	83.6	83.6

Table 8: GPT-3 1.3B 10-shot evaluation results. The first column is the results of the original OpenAI GPT-3 1.3B model [7]. All the other columns are in the same order as the rows in main paper Tab. 3. OpenAI results are not directly comparable to ours because the training data are different. Note that OpenAI used different number of shots for each task, while we use the same 10 shots for all tasks.

Case	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
Train tokens	OpenAI	baseline	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL	CL
	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B	300B
Avg.	49.0	44.0	44.8	44.5	44.5	44.9	44.4	44.9	45.1	44.0	44.5	44.8	42.7	43.7	43.5	44.0
(0) HellaSwag	54.9	52.4	52.7	52.6	52.0	52.7	52.8	54.7	55.1	51.2	52.2	53.4	50.5	50.9	52.2	53.0
(1) LAMBADA	57.0	57.6	56.0	57.0	55.7	57.0	57.6	59.5	59.6	55.1	56.4	58.4	54.2	55.7	57.5	58.9
(2) TriviaQA	32.1	13.5	14.0	13.9	13.2	14.7	13.0	13.5	13.7	12.6	12.9	12.4	11.5	12.0	11.5	12.3
(3) WebQs	19.6	11.8	11.9	12.0	12.9	12.6	12.5	12.5	13.8	12.1	11.5	12.0	10.0	11.6	10.2	12.1
(4) Winogrande	59.1	57.4	56.7	58.9	58.2	60.0	58.2	58.7	58.1	55.9	59.2	59.0	56.8	58.0	58.4	58.4
(5) PIQA	74.3	71.5	71.4	71.5	71.4	71.5	72.3	71.6	72.6	71.1	72.0	71.9	71.2	71.7	71.4	71.4
(6) ARC Challenge	36.7	32.8	32.2	33.4	32.7	32.8	32.5	32.8	34.6	32.3	32.7	33.4	31.7	31.2	30.5	31.7
(7) RACE-h	59.1	63.5	65.2	64.6	64.7	64.7	64.4	64.2	65.9	63.2	63.9	62.5	61.5	63.0	61.7	63.0
(8) ANLI R1	32.5	29.8	31.6	31.4	31.7	31.6	32.7	32.3	32.7	31.3	32.5	30.7	32.0	30.8	33.0	32.4
(9) ANLI R2	31.4	34.4	34.6	33.0	31.2	33.7	31.9	32.4	32.6	34.0	32.9	31.9	31.0	32.0	34.0	34.0
(10) ANLI R3	36.0	33.6	34.1	33.1	33.4	33.8	33.8	32.8	33.8	31.9	33.9	33.9	32.7	31.7	35.2	35.2
(11) OpenBookQA	50.6	32.4	34.0	34.6	34.0	35.4	35.2	33.6	32.6	33.0	33.2	33.2	33.4	33.4	33.2	29.8
(12) RACE-h	41.4	34.5	36.6	35.4	35.3	36.7	35.5	37.1	36.7	35.7	34.4	35.3	35.5	34.2	35.9	34.6
(13) BoolQ	64.1	60.8	63.5	59.4	63.1	62.1	63.1	64.2	64.0	62.8	62.1	63.8	58.8	63.4	58.2	62.0
(14) Copa	77.0	76.0	74.0	79.0	76.0	76.0	74.0	73.0	74.0	74.0	77.0	76.0	69.0	70.0	71.0	70.0
(15) RTE	50.9	48.0	55.2	50.5	53.8	52.7	49.1	53.1	52.0	56.0	54.5	55.6	48.0	56.0	48.4	51.2
(16) WSC	49.0	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5
(17) MultiRC	20.8	5.88	7.24	5.35	6.93	5.77	5.98	6.19	5.35	4.51	5.67	6.72	4.51	6.19	5.67	6.4
(18) ReCoRD	84.0	83.0	83.4	83.3	82.4	83.6	83.2	84.6	84.0	82.3	82.7	83.9	82.2	82.4	83.8	83.3

## 617 A Appendix

### 618 A.1 GPT-3 pretraining experimental setup and detailed results

619 For GPT-3 pretraining, we set some of the hyperparameters the same as the original OpenAI work [7]:  
620 seqlen 2K, batch size 512, learning rate  $2e-4$  (batch size 256 and learning rate  $3e-4$  for the GPT-3  
621 MoE 6.7B model since we use 350M as the base model). We set other hyperparameters differently:  
622 (1) OpenAI pretrains GPT-3 on 300B tokens. To evaluate data efficiency techniques, we pretrain  
623 with 9 different total training tokens: 300B, 200B (67%), 150B (50%), 96B (32%), 48B (16%), 24B  
624 (8%), 12B (4%), 6B (2%), 3B (1%). (2) When using less than 300B training tokens, we increase the  
625 peak learning rate proportionally (e.g., 2x LR when using 50% data). This is similar to the traditional  
626 learning rate scaling when using different batch sizes. However, when using extremely small amount

Table 9: GPT-3 1.3B 0-shot evaluation results when pretraining with 1%, 2%, 4%, 8%, 16%, and 32% of data.

Case	(2)		(4)		(6)		(8)		(10)		(12)	
	CL		CL		CL		CL		CL		CL	
	(1)	seqtru	(3)	seqtru	(5)	seqtru	(7)	seqtru	(9)	seqtru	(11)	seqtru
Model size	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc
Train tokens	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B
	3B	3B	6B	6B	12B	12B	24B	24B	48B	48B	96B	96B
Avg.	34.5	35.0	36.3	36.8	37.2	38.4	38.8	40.2	39.8	41.2	41.5	42.2
(0) HellaSwag	28.7	29.3	30.8	33.2	35.4	38.1	39.0	42.7	43.5	46.9	47.8	49.9
(1) LAMBADA	28.9	32.0	38.0	41.4	43.5	49.5	50.3	53.9	54.3	58.0	57.8	60.4
(2) TriviaQA	1.18	1.4	1.58	1.56	1.79	1.89	2.28	3.91	3.5	4.82	6.29	6.16
(3) WebQs	0	0.148	0.443	0.738	1.03	0.935	0.984	0.984	1.08	2.36	2.21	2.51
(4) Winogrande	51.3	50.8	52.2	51.0	49.5	51.8	50.7	54.1	53.5	54.9	53.3	56.5
(5) PIQA	62.1	61.6	62.5	63.5	64.9	66.6	66.8	68.5	68.6	69.6	70.1	71.3
(6) ARC Challenge	22.2	22.9	24.9	23.0	24.7	24.6	24.1	26.2	26.7	26.6	28.5	28.2
(7) ARC Easy	38.8	38.4	40.5	41.0	44.1	45.2	46.4	47.7	48.6	50.7	51.2	52.7
(8) ANLI R1	33.3	33.3	32.6	33.3	31.5	31.5	31.7	32.7	33.2	33.7	33.4	33.0
(9) ANLI R2	33.2	34.6	35.8	32.7	31.7	32.8	32.6	33.6	33.1	34.0	34.1	34.4
(10) ANLI R3	32.8	33.9	35.4	32.9	34.4	34.9	35.4	34.5	32.2	35.1	33.7	33.5
(11) OpenBookQA	25.6	24.4	26.2	27.2	28.2	28.0	28.8	29.6	30.4	31.6	32.2	31.6
(12) RACE-h	27.1	28.5	28.9	29.4	30.0	31.2	32.2	32.5	31.8	33.5	34.5	35.2
(13) BoolQ	58.4	56.4	53.3	56.8	56.0	57.3	59.2	62.0	58.7	60.3	61.9	60.1
(14) Copa	61.0	64.0	66.0	71.0	68.0	69.0	70.0	72.0	69.0	69.0	70.0	71.0
(15) RTE	52.7	52.3	53.4	53.1	53.4	54.2	54.2	53.4	52.3	53.1	53.4	55.6
(16) WSC	36.5	36.5	39.4	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5
(17) MultiRC	0.839	0.839	1.15	0.839	1.47	0.839	0.839	0.839	0.839	1.36	0.839	1.47
(18) ReCoRD	60.6	63.4	66.6	70.3	71.5	75.6	75.8	78.8	78.7	81.3	81.4	82.3

Table 10: GPT-3 1.3B 10-shot evaluation results when pretraining with 1%, 2%, 4%, 8%, 16%, and 32% of data.

Case	(2)		(4)		(6)		(8)		(10)		(12)	
	CL		CL		CL		CL		CL		CL	
	(1)	seqtru	(3)	seqtru	(5)	seqtru	(7)	seqtru	(9)	seqtru	(11)	seqtru
Model size	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc	baseline	+voc
Train tokens	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B	1.3B
	3B	3B	6B	6B	12B	12B	24B	24B	48B	48B	96B	96B
Avg.	33.9	35.0	35.6	36.6	37.3	38.8	38.8	40.7	40.7	42.3	43.0	43.2
(0) HellaSwag	28.9	29.5	31.3	33.2	35.2	38.2	39.3	43.1	43.6	47.0	47.9	50.3
(1) LAMBADA	24.5	27.5	32.2	36.2	37.6	44.9	44.0	50.7	47.0	53.2	51.8	57.0
(2) TriviaQA	0.804	1.36	1.75	3.05	3.21	4.93	5.27	6.96	7.51	9.45	10.6	11.0
(3) WebQs	1.08	1.72	2.17	2.9	3.44	5.22	4.87	6.94	7.73	8.66	10.4	11.4
(4) Winogrande	51.6	51.0	52.2	50.2	51.8	54.0	51.7	55.2	57.0	55.1	57.0	56.1
(5) PIQA	60.9	62.0	62.1	63.9	65.3	66.5	66.0	67.9	68.8	69.7	69.8	71.1
(6) ARC Challenge	21.9	23.2	24.0	24.3	24.8	24.9	26.5	27.7	28.0	29.8	31.5	32.1
(7) ARC Easy	38.7	41.9	44.9	47.1	50.0	52.4	54.1	55.6	56.4	59.8	60.6	62.5
(8) ANLI R1	31.7	33.5	33.4	32.8	34.1	32.6	35.2	33.0	31.6	33.7	33.0	31.2
(9) ANLI R2	33.1	35.0	30.3	34.7	35.6	34.4	34.2	31.0	33.6	34.4	32.4	32.5
(10) ANLI R3	33.9	34.8	35.1	33.2	33.5	34.2	33.4	33.2	34.5	33.2	34.2	32.8
(11) OpenBookQA	25.0	26.0	27.2	28.4	28.8	26.0	27.2	28.6	29.2	31.2	32.6	33.0
(12) RACE-h	26.9	27.8	29.1	28.9	29.1	30.5	32.3	31.9	32.0	34.3	34.4	35.0
(13) BoolQ	49.1	50.0	45.6	49.1	45.4	56.2	48.0	56.3	55.6	60.2	62.1	58.3
(14) Copa	62.0	66.0	70.0	66.0	69.0	67.0	71.0	70.0	66.0	70.0	72.0	72.0
(15) RTE	53.1	49.5	47.3	50.2	48.4	48.7	48.0	56.3	55.6	50.9	54.2	49.1
(16) WSC	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5	36.5
(17) MultiRC	5.25	4.72	4.41	4.3	5.35	4.3	5.14	3.88	5.04	5.56	5.67	6.93
(18) ReCoRD	59.8	63.0	66.0	69.6	70.8	75.0	74.6	78.7	77.8	80.9	80.7	82.1

of data (e.g., 1% data), we find that using too larger learning rate (e.g., 100x) could lead to divergence. In such case we keep halving learning rate until the training succeed. (3) We do not use OpenAI's batch size warmup method because our GPT-3 125M model pretraining experiments show that it does not help on model quality under the same total training tokens. And the smaller batch sizes prevent us to pretrain on large number of GPUs at the beginning, which leads to longer training wall-clock time; (4) Since we don't use the batch size warmup, our training has more tokens at early steps. Thus we increase the linear learning rate warmup duration from OpenAI's 375M tokens to 3B tokens (except when using 3B tokens in total, where we use first 1.5B tokens for warmup); (5) OpenAI uses a single cycle cosine learning rate decay over 260B tokens, and the min learning rate is 10% of peak learning rate. However, based on our experiments and related works [57, 20], we changed the decay duration to always equal to total training token and the min learning rate to always equal to 1e-6, which provide better model quality. When calculating the total consumed training token, we take CL

Table 11: GPT-3 MoE 6.7B 0-shot evaluation results.

Case	(2) CL seqtru	
	(1) baseline	+voc +rLTD
Model size	6.7B	6.7B
Train tokens	300B	300B
Avg.	42.8	43.5
(0) HellaSwag	53.0	53.3
(1) LAMBADA	60.1	59.6
(2) TriviaQA	11.0	9.31
(3) WebQs	2.95	2.31
(4) Winogrande	56.0	56.8
(5) PIQA	72.0	71.8
(6) ARC Challenge	28.9	28.9
(7) ARC Easy	54.5	54.2
(8) ANLI R1	33.6	30.8
(9) ANLI R2	32.8	34.1
(10) ANLI R3	33.6	35.5
(11) OpenBookQA	33.6	32.4
(12) RACE-h	33.8	35.0
(13) BoolQ	61.5	57.5
(14) Copa	71.0	74.0
(15) RTE	54.5	55.2
(16) WSC	36.5	51.0
(17) MultiRC	1.89	1.78
(18) ReCoRD	82.4	82.6

and random-LTD (which change number of tokens on certain steps) into consideration. For CL and random-LTD hyperparameters, we use the low-cost tuning strategy described in Sec. 3.

To evaluate the quality of pretrained GPT-3 models, we perform 0-shot and 10-shot evaluations on 19 tasks used by original OpenAI work: HellaSwag [59], LAMBADA [35], TriviaQA [22], WebQs [4], Winogrande [44], PIQA [5], ARC Challenge/Easy [11], ANLI R1/R2/R3 [34], OpenBookQA [32], RACE-h [27], BoolQ [10], Copa [1], RTE [12], WSC [28], MultiRC [56], and ReCoRD [60]. Since there is no additional training involved in 0/10-shot evaluations, it’s impossible to try multiple seeds thus each task only has one accuracy result. We then take the average accuracy over the 19 tasks.

Tab. 7 and 8 present the 0-shot and 10-shot accuracy results for each task achieved by the pretrained GPT-3 1.3B models. Tab. 9 and 10 present the 0-shot and 10-shot accuracy results for the same GPT-3 1.3B model but pretrained with even less data as discussed in main paper Fig. 2, Sec. 1, and Sec. 4.1. Tab. 11 presents the 0-shot accuracy results for each task achieved by the pretrained GPT-3 MoE 6.7B models, as discussed in main paper Sec. 4.1.

## A.2 BERT-large pretraining experimental setup and detailed results

For BERT-large pretraining, we set some of the hyperparameters the same as the Megatron-LM work [46] since it achieves better model quality than original BERT: seqlen 512, batch size 1024, learning rate  $1e-4$  with linear warmup up at first 10000 steps and then linearly decay to  $1e-5$ . We set other hyperparameters differently: (1) Megatron-LM pretrains over 2M steps (1049B tokens). To evaluate data efficiency techniques, we pretrain with 3 different total training tokens: 1049B, 703B (67%), and 524B (50%). (2) When using less than 1049B training tokens, we increase the peak learning rate proportionally. (3) Megatron-LM decays the learning rate over 2M steps. Since our techniques could change the number of tokens at some steps, we change the decay to token-based and set the decay duration always the same as total training tokens. For CL and random-LTD hyperparameters, we use the low-cost tuning strategy described in Sec. 3.

To evaluate the quality of pretrained BERT-large models, we finetune the models for 8 tasks from the GLUE benchmark [52]: MNLI, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, RTE. We follow the finetuning hyperparameters from the original BERT work [14]: 3 epochs, batch size 32. For learning rate we test  $5e-5$ ,  $4e-5$ ,  $3e-5$ ,  $2e-5$  on the baseline and find that  $3e-5$  provides the best average GLUE score, thus we select  $LR=3e-5$  for the comparison between baseline and proposed work. We perform finetuning on 5 seeds (1234 to 1238) and take the median/std on each task, then we take the average of the median scores as the average GLUE score, and take the average of std scores as the overall std.

Tab. 12 presents the finetuning results for each task achieved by the pretrained BERT-large models.

Table 12: BERT-large finetuning results. The first row is the results of the original BERT-large model [14]. All the other rows are in the same order as the rows in main paper Tab. 4. Original BERT results are not directly comparable to ours because the training data and total training token are different.

Case	Train tokens	Average	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
(0)original	43B	82.1	86.7	85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1
(1)baseline	1049B	87.29±0.53	88.54±0.16	89.25±0.13	92.1±0.07	94.12±0.15	94.33±0.48	64.36±1.59	90.43±0.21	89.32±0.81	83.2±1.16
(2)CL_seqtru	1049B	87.31±0.57	89.03±0.14	89.35±0.24	92.21±0.03	94.12±0.11	94.68±0.1	62.08±2.06	90.72±0.27	89.58±0.52	83.98±1.64
(3)CL_seqreo	1049B	87.48±0.61	88.81±0.16	89.27±0.19	92.2±0.12	93.99±0.28	94.79±0.42	62.86±1.85	90.51±0.25	89.32±0.85	85.55±1.34
(4)CL_voc	1049B	87.36±0.64	88.64±0.23	89.24±0.16	92.32±0.05	94.03±0.09	95.14±0.31	63.34±1.82	90.07±0.18	89.84±1.06	83.59±1.83
(5)CL_seqtru_voc	1049B	87.6±0.34	88.9±0.1	89.29±0.17	92.26±0.05	94.26±0.19	95.25±0.4	64.6±0.6	90.38±0.25	90.62±0.22	82.81±1.05
(6)CL_seqreo_voc	1049B	87.06±0.52	88.73±0.13	88.91±0.26	92.32±0.07	93.92±0.08	94.91±0.25	61.05±1.15	90.36±0.23	89.32±1.13	83.98±1.34
(7)random-LTD	1049B	88.17±0.48	88.74±0.25	89.18±0.21	92.27±0.1	94.32±0.21	95.02±0.38	67.3±1.5	90.65±0.15	90.1±0.63	85.94±0.89
(8)CL_seqtru_voc+random-LTD	1049B	87.69±0.32	88.79±0.13	89.26±0.04	92.34±0.08	94.21±0.23	95.14±0.36	65.46±0.68	90.44±0.19	89.58±0.56	83.98±0.59
(9)baseline	703B	87.19±0.49	88.75±0.18	89.11±0.19	92.13±0.08	93.99±0.16	95.14±0.46	62.07±1.44	90.08±0.31	89.84±0.68	83.59±0.87
(10)CL_seqtru_voc	703B	87.29±0.62	88.96±0.07	89.15±0.25	92.21±0.09	94.23±0.08	95.25±0.33	62.19±1.75	89.92±0.21	90.1±0.55	83.59±2.25
(11)random-LTD	703B	87.99±0.38	88.86±0.1	88.79±0.12	92.01±0.12	94.25±0.17	94.68±0.32	67.1±0.9	90.55±0.19	89.32±0.39	86.33±1.12
(12)baseline	524B	86.61±0.5	88.53±0.14	88.77±0.17	92.04±0.11	93.93±0.19	95.02±0.25	61.05±1.22	89.88±0.25	88.28±1.08	82.03±1.13
(13)CL_seqtru_voc	524B	86.9±0.33	88.66±0.14	89.25±0.21	92.08±0.05	93.99±0.26	95.02±0.17	63.34±0.52	89.96±0.25	88.54±0.22	81.25±1.14
(14)random-LTD	524B	87.32±0.48	88.81±0.15	88.9±0.13	91.96±0.04	94.28±0.14	94.91±0.43	64.41±1.32	90.39±0.25	89.06±0.18	83.2±1.67
(15)CL_seqtru_voc+random-LTD	524B	87.44±0.46	88.9±0.19	88.9±0.13	92.19±0.09	94.17±0.12	94.68±0.35	65.97±1.09	90.31±0.22	89.06±0.79	82.81±1.13

### A.3 GPT-2 finetuning experimental setup

Due to the lack of published training recipe, we first perform a hyperparameter search for the baseline case (256 combinations of batch size, LR schedule, number of epochs). Then using the combination that provides best baseline validation perplexity, we apply CL and random-LTD (each with 16 different combinations of their two hyperparameters) to verify if they could further improve the model quality.

For GPT-2<sub>350M</sub> finetuning on PTB [30], we use an already-pretrained GPT-2<sub>350M</sub> model checkpoint and an example script <sup>2</sup> from Huggingface. Given the much smaller training cost (about 38min on a single V100 for 5 epochs), we focus on improving the model quality under the same amount of data. Due to the lack of published training recipe, we first perform a hyperparameter search for the baseline case: we tried 256 combinations of batch size (4, 8, 16, 32), learning rate (2e-5, 3e-5, 5e-5, 10e-5), learning rate warmup (0% and 10% linear warmup steps), learning rate decay (no decay, linear decay), and number of epochs (2, 3, 5, 10). For this sweep we only use one seed (1234) due to the number of combinations. Results show that the best combination among the 256 cases is: batch size 4, learning rate 10e-5, 0% learning rate warmup, linear learning rate decay, and 5 epochs. Results also show that for this task using more epochs (5 or 10) leads to better validation perplexity than less epochs (2 or 3).

Then using this combination that provides best baseline validation perplexity, we apply CL and random-LTD (each with 16 different combinations of their two hyperparameters) to verify if they could further improve the model quality. For CL we test 5 metrics (seqtru, seqres, voc, seqtru\_voc, seqres\_voc), each with 16 different combinations of its two hyperparameters: start difficulty  $d_s$  (8, 32, 128, 512 for seqtru/seqres, and 1%, 10%, 30%, 50% for voc) and total CL steps  $T_c$  (10%, 30%, 50%, 70% of the baseline’s total steps). Results show that the seqres metric provides the best model quality, and its best hyperparameter combination is  $d_s = 32, T_c = 70\%$  of baseline steps. For random-LTD we test 16 different combinations of its two hyperparameters: start seqlen  $r_s$  (8, 32, 128, 512) and total steps  $T_r$  (10%, 30%, 50%, 70% of the baseline’s total steps). Results show that the best hyperparameter combination is  $r_s = 128, T_r = 30\%$  of baseline steps. For CL+random-LTD composed case, we re-tuned the combination of  $T_c$  and  $T_r$  (CL will first adjust seqlen before random-LTD. To have a meaningful composition, it essentially requires  $T_c < T_r$ ) and the best combination is  $d_s = 32, r_s = 128, T_c = 10\%, T_r = 30\%$  of baseline steps. At last, for the best case of baseline, CL, random-LTD, and CL+random-LTD, we run another 4 seeds (1235 to 1238) and then calculate the median/std of the validation perplexity.

<sup>2</sup>[https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run\\_clm\\_no\\_trainer.py](https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_clm_no_trainer.py)

Table 13: Comparing random-LTD (w/o MSLG) and TokenBypass under various constant dropping schedule. Baseline achieves a perplexity of  $16.11 \pm 0.04$ .

Token saving ratio	1.88%	12.75%	23.72%	34.59%	45.45%	56.43%
random-LTD (w/o MSLG)	$16.15 \pm 0.01$	$16.83 \pm 0.06$	$17.95 \pm 0.08$	$20.02 \pm 0.05$	$23.35 \pm 0.16$	$30.65 \pm 0.78$
TokenBypass	$16.4 \pm 0.04$	$17.3 \pm 0.06$	$18.59 \pm 0.19$	$23.09 \pm 0.23$	$28.56 \pm 0.24$	$35.91 \pm 0.26$

#### A.4 ViT finetuning experimental setup

We apply random-LTD to the vision transformer (ViT) [15] on finetuning tasks to demonstrate the broader applications of our method across different domains. We use the pretrained models published in [54] and test on two small image recognition benchmarks—CIFAR10 and CIFAR100 [26], and one large-scale dataset—ImageNet [13]. For ImageNet (CIFAR10/100), we use the 12-layer (24-layer) pretrained ViT with an input resolution  $224 \times 224$  in which each patch of size  $16 \times 16$  such that the sequence length becomes  $196 + 1$  (the extra token is for position). ImageNet (CIFAR10/100) is trained on 8-GPU (1-GPU) and the batch size is 32 (128) images per GPU. The training budget for all three datasets is 14 epochs and a small constant learning rate is used based on grid search. Particularly, the best learning rate for ImageNet (CIFAR) is  $5e-5$  ( $1e-4$ ). For ImageNet (CIFAR), when applying random-LTD the sequence length is started with 66 (32) and linearly reaches to the 197 full sequence length at 80% of the total baseline training iterations, equivalent to a 1.3x (1.4x) data saving.

#### A.5 Comparing random-LTD with the TokenBypass work

In main paper Sec. 4.2 we demonstrate that random-LTD achieves 2x data saving while maintaining model quality for BERT pretraining, greatly surpassing the 1.3x data saving achieved by the state-of-the-art TokenBypass work [21]. In this section we provide additional discussion and evaluation to compare random-LTD with TokenBypass.

We include the illustration of the comparison between baseline, TokenBypass, and random-LTD in Fig. 7. First, the takeaway from TokenBypass can be summarized into (1) drop unimportant tokens starting from an intermediate layer of the model, (2) the dropping schedules is a fixed constant function (drop half of the tokens), and (3) the dropping criterion based on the “accumulated masked language modeling loss” (which is referred to as “token loss” since it needs each token’s loss)

However, TokenBypass have several limitations (1) only tested on BERT pretraining (we find that it’s less effective in GPT pretraining and finetuning), (2) the bypass layer starting only from an intermediate layer (e.g., 6L for BERT-base), and (3) the dropping criterion based on “token loss” may not be accessible for some tasks, like classification problems.

Acknowledging that we are inspired by their excellent work and trying to solve their limitations, we believe random-LTD consists of three differences: (1) drop tokens starting from the 2nd layer of the model, (2) propose a linear increasing dropping schedule to close the training and inference discrepancy, and (3) the new random dropping criterion (which has lower overhead and can be easily applied to tasks without “token loss”, such as vision transformer). Next, we provide more direct comparisons between random-LTD and TokenBypass on GPT-2 finetuning and GPT-3 pretraining tasks. Note that because this study was performed in parallel with other experiments, the hyperparameter choices are different from the experiments in main paper.

**GPT-2 finetuning on PTB with various constant dropping schedule.** To better demonstrate the benefit of random selection per layer, we provide a study with various constant dropping schedule. Particularly, from the second layer to the last second layer, we use one of the sequence lengths from 921, 819, 716, 614, 512, 409, of which the corresponding token saving ratio are shown in Tab. 13. We finetune GPT-2<sub>350M</sub> (24 layers) on the PTB dataset with constant learning rate  $5e-5$  and Adam optimizer for 15 epochs (batch-size 8). The results are the best validations (average of three runs and one standard deviation) of random-LTD (without Monotonic Sequence Length Growth, MSLG) and TokenBypass.

As shown in Tab. 13, for all cases random-LTD has better performance than TokenBypass, even without one of the key contributions, Monotonic Sequence Length Growth (MSLG). This further verifies the conjecture we made in the main paper: “However, several works [50, 31, 51] have shown that MHA focuses on different tokens at different layer depths and the attention map aligns with the dependency relation most strongly in the middle of transformer architectures. Therefore, TokenBypass

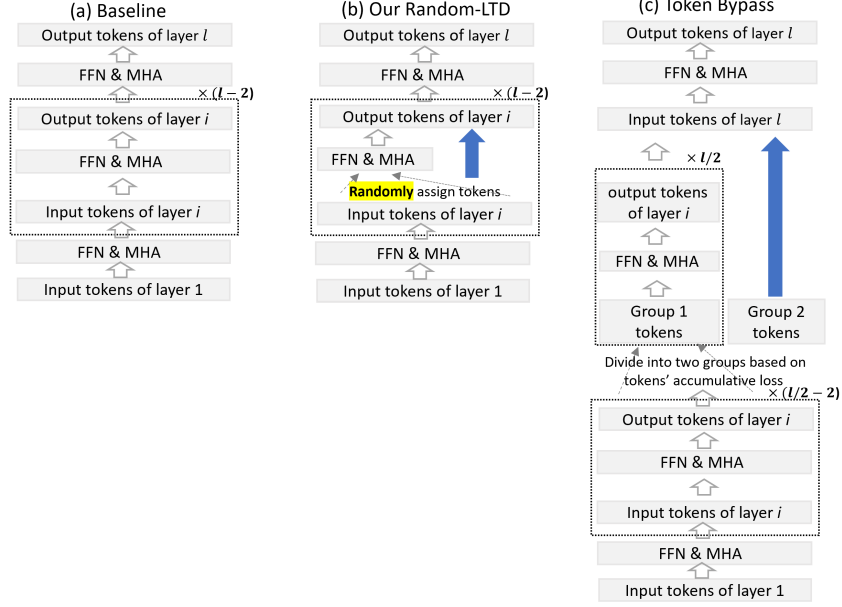


Figure 7: Illustration of the transformer model for the baseline training (left), TokenBypass training (right) and random-LTD training (middle). Compared to TokenBypass, random-LTD requires no criterion on the dropped tokens and trains well for all middle layers. The box with dash line is a repeated block. For both (a) and (b), the block is repeated by  $l - 2$  times, while for (c), the block is repeated by  $l/2$ . In the box, “Output tokens of layer  $i$ ” is the same as “Input tokens of layer  $i + 1$ ”.

Table 14: Comparing random-LTD and TokenBypass (both with our proposed MSLG applied) under various token saving ratios. Baseline achieves a perplexity of  $16.11 \pm 0.04$ .

Token saving ratio	8%	16%	24%	32%	40%	47%	52%	55%
random-LTD	15.91 $\pm$ 0	15.86 $\pm$ 0.06	15.86 $\pm$ 0.01	15.85 $\pm$ 0.02	16.05 $\pm$ 0.06	17.02 $\pm$ 0.05	18.41 $\pm$ 0.04	20.01 $\pm$ 0.06
TokenBypass (w/ MSLG)	16.1 $\pm$ 0.02	16.09 $\pm$ 0.05	16.21 $\pm$ 0.03	16.54 $\pm$ 0.01	17.06 $\pm$ 0.04	18.64 $\pm$ 0.04	23.12 $\pm$ 0.22	25.77 $\pm$ 0.57

used in [21], i.e., fully skipping middle layers, may hinder the learnability/generalization of the architecture during pretraining/inference.”

**GPT-2 finetuning on PTB with our proposed MSLG.** We are also curious if MSLG can help boost the performance of TokenBypass. Therefore, we also perform the comparison between random-LTD (with MSLG) and TokenBypass (with MSLG) on GPT-2 finetuning. We start at sequence length from 128 and linearly increase to full sequence 1024, with a different total steps to achieve different token saving ratios shown in Tab. 14. The rest of the hyperparameters are the same as the previous experiment.

Note that under MSLG it is hard to control the overall token saving ratio to be the same as constant dropping schedule case. But comparing Tab. 14’s 24%/47%/55% with Tab. 13’s 23.72%/45.45%/56.43%, we can clearly see the benefit of MSLG. Meanwhile, comparing the results of random-LTD and TokenBypass (with MSLG), it is clear that random-LTD still has better performance than TokenBypass for all cases. This shows that the other components of random-LTD, particularly the layerwise dropping mechanism, has its unique advantage over accumulated token loss for auto-regressive generative models.

**GPT-3 pretraining.** To directly compare the two techniques on pretraining tasks, we pretrain a GPT-3 350M model with 30B tokens. Due to limited time and resource, this is a smaller model and 10% of data compared to our other GPT-3 pretraining experiments. And due to the same reason we only compare the validation loss at the end of pretraining, but our experience shows that this metric has strong correlation with downstream task zero/few-shot evaluation performance. Based on the last GPT-2 finetuning experiment, here we again apply MSLG to TokenBypass. Results in Tab. 15 shows that under the same token saving ratio, random-LTD provides significantly better model quality than TokenBypass.

Table 15: Comparing random-LTD and TokenBypass (both with our proposed MSLG applied) on GPT-3 pretraining.

	Validation loss
baseline	8.22
random-LTD (37.76% token saving)	8.26
TokenBypass (w/ MSLG, 37.76% token saving)	9.62

774 **Other downstream tasks.** TokenBypass cannot be easily extended to various downstream tasks. The  
775 reason is that the TokenBypass criterion is based on the “token loss”, but downstream tasks, e.g.,  
776 classification and regression (GLUE benchmark), do not have “token loss”. Therefore, we did not  
777 find an easy way to apply TokenBypass on those tasks.