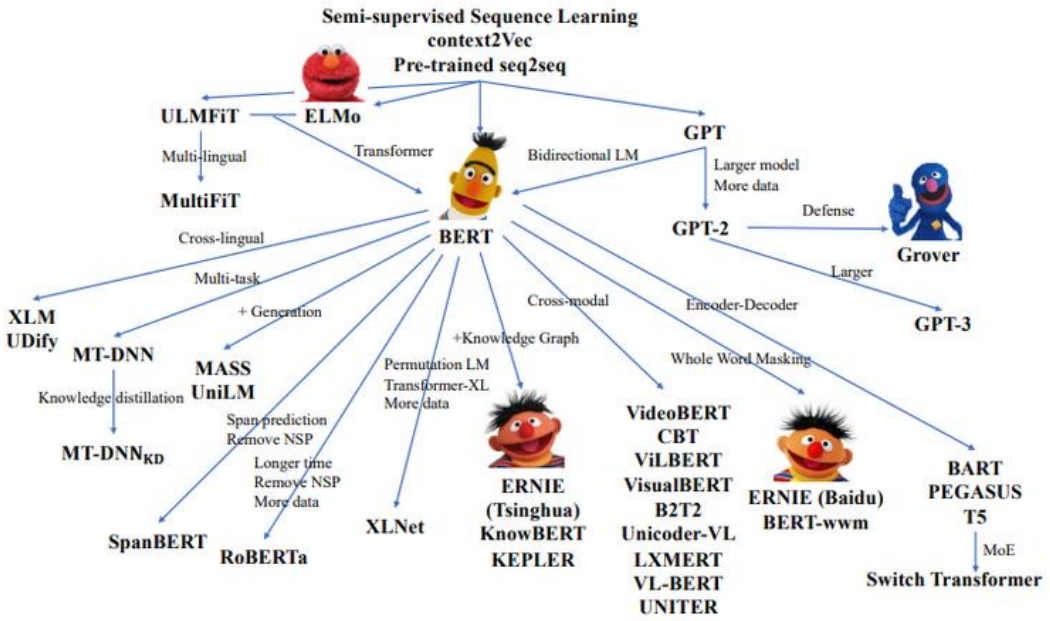
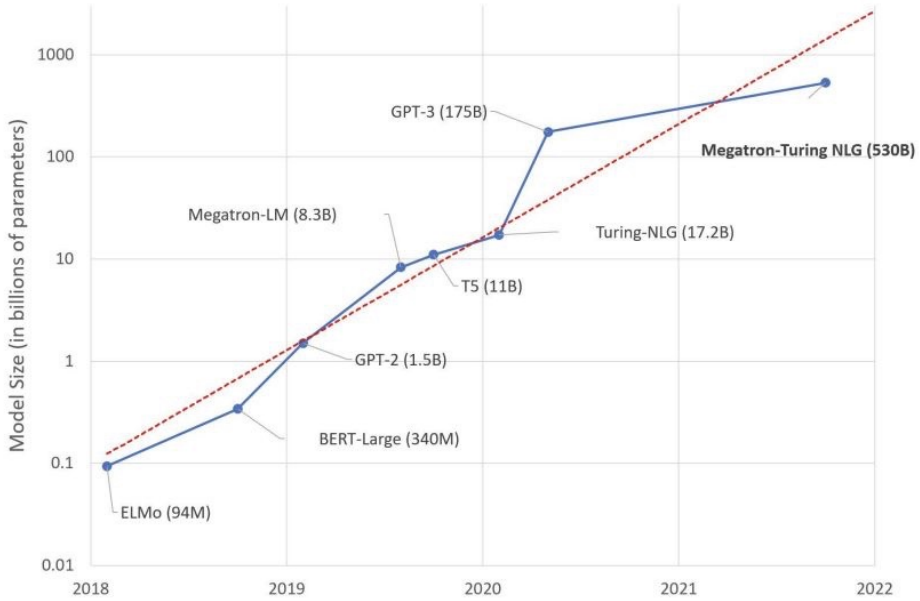
Knowledge Inheritance

Ding Xuanwen

Motivation

Despite the great follow-up efforts of exploring various pre-training techniques and model architectures, researchers find that simply enlarging the **model capacity, data size and training steps** can further improve the performance of PLMs





Motivation

- However, training large-scale PLMs requires tremendous computational resources, raising severe environmental concerns on the prohibitive computational costs
- Existing PLMs are generally trained from scratch individually, ignoring that many well-trained PLMs are available

	Organization	Model	Parameter	Data	Time
2018.6	OpenAI	GPT	110M	4GB	3 Day
2018.10	Google	BERT	330M	16GB	50 Day
2019.2	OpenAI	GPT-2	1.5B	40GB	200 Day
2019.7	Facebook	RoBERTa	330M	160GB	3 Year
2019.10	Google	T5	11B	800GB	66 Year
2020.6	OpenAI	GPT-3	175B	560GB	90 Year

How could existing PLMs benefit training larger PLMs in future?

Knowledge Inheritance

- **Knowledge Inheritance**
  - Through knowledge distillation [1]
  - Through parameter recycling [2][3]
  - …

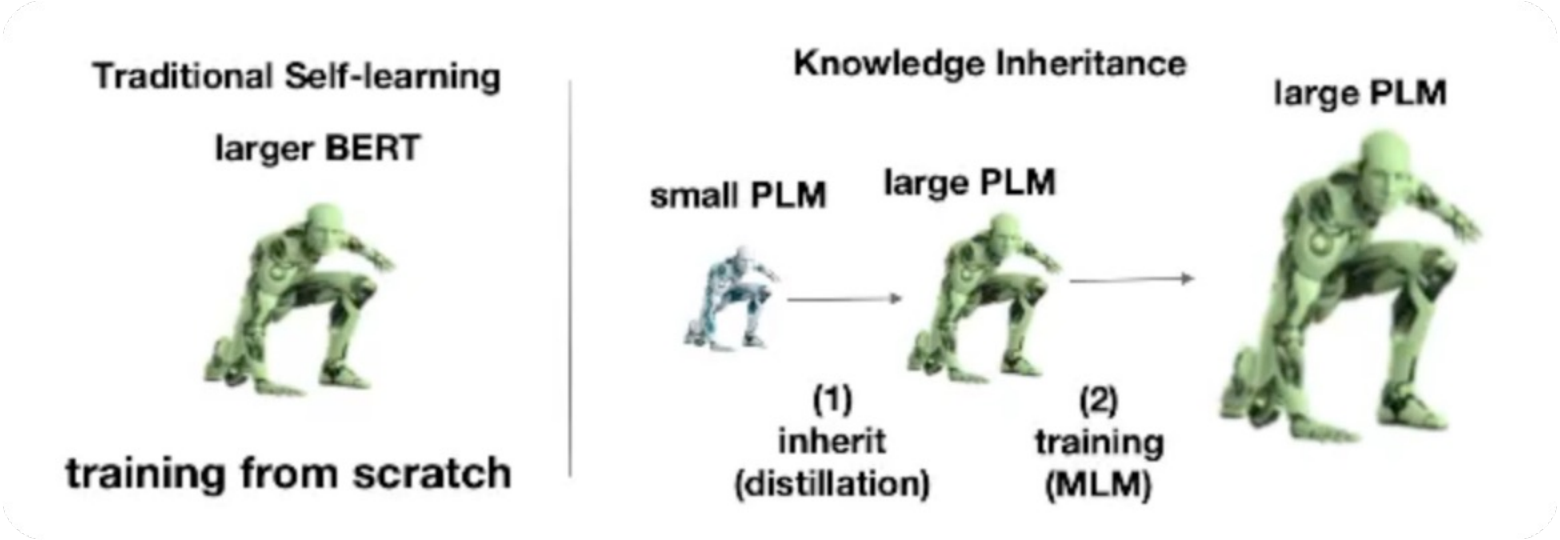
[1] Knowledge Inheritance for Pre-trained Language Models. Qin et al. NAACL 2022, oral
[2] bert2BERT: Towards Reusable Pretrained Lanruage Models. Chen et al. ACL 2022 poster
[3] ELLE: Efficient Lifelong Pre-training for Emerging Data. Qin et al. ACL 2022 findings

Knowledge Inheritance for Pre-trained Language Models

Yujia Qin[12], **Yankai Lin**[2], **Jing Yi**[1], **Jiajie Zhang**[1], **Xu Han**[1], **Zhengyan Zhang**[1], **Yusheng Su**[1], **Zhiyuan Liu**[1], **Peng Li**[2], **Maosong Sun**[1], **Jie Zhou**[2]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Pattern Recognition Center, WeChat AI, Tencent Inc.
yujiaqin16@gmail.com

Motivation

- Humans can leverage the knowledge summarized by their predecessors to learn new tasks
- Similarly, it is worth inheriting the implicit knowledge distributed in existing PLMs
- In this sense, we could **distill the knowledge** summarized by an existing small PLM during pretraining to efficiently learn larger PLMs
- We dub the above process as **knowledge inheritance (KI)**



Method

- Knowledge Inheritance Framework
- Train a large PLM M_L by inheriting knowledge from M_S

$$\begin{aligned}\mathcal{L}(\mathcal{D}_L;\mathcal{M}_S) &= \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}_L} (1-\alpha)\mathcal{L}_{\text{SELF}}(\mathbf{x}_i,\mathbf{y}_i) + \alpha\mathcal{L}_{\text{KI}}(\mathbf{x}_i;\mathcal{M}_S) \\ &= \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}_L} (1-\alpha)\mathcal{H}(\mathbf{y}_i,\mathcal{P}_{\mathcal{M}_L}(\mathbf{x}_i;1)) + \alpha\tau^2\mathbf{KL}(\mathcal{P}_{\mathcal{M}_S}(\mathbf{x}_i;\tau)||\mathcal{P}_{\mathcal{M}_L}(\mathbf{x}_i;\tau))).\end{aligned}$$

- Dynamic Inheritance Rate

$$\mathcal{L}(\mathcal{D}_L;\mathcal{M}_S) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}_L} (1-\alpha_t)\mathcal{L}_{\text{SELF}}(\mathbf{x}_i,\mathbf{y}_i) + \alpha_t\mathcal{L}_{\text{KI}}(\mathbf{x}_i;\mathcal{M}_S).$$

$$\alpha_t = \max(1-\alpha_T \times \frac{t}{T}, 0)$$

Research Questions

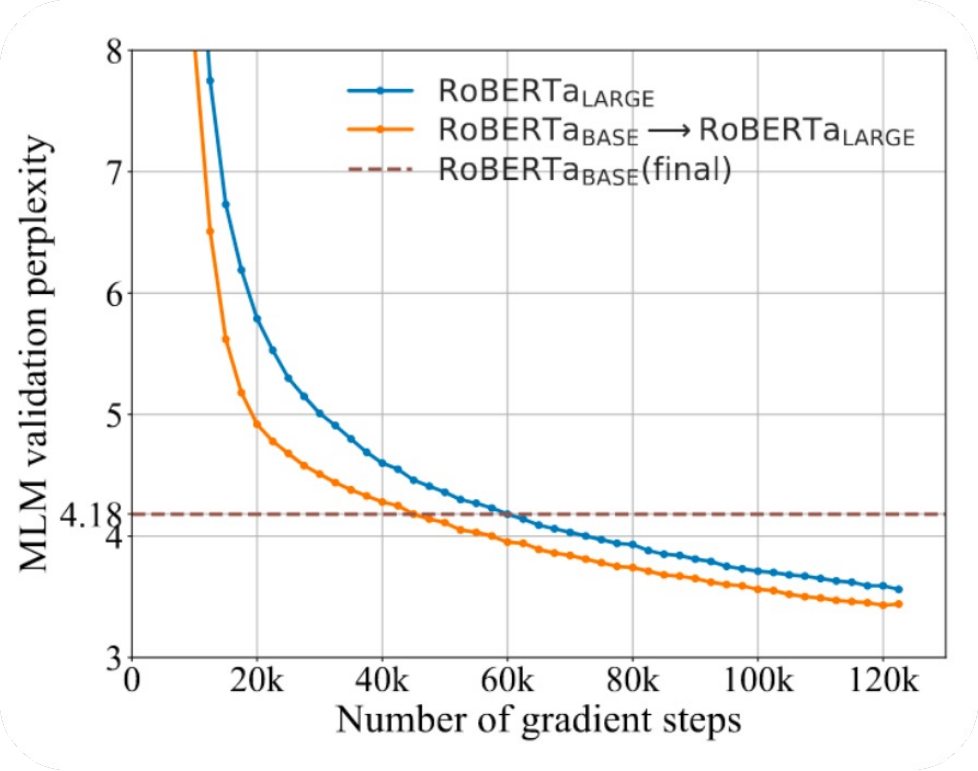
- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- (RQ2) Considering human beings are able to hand down knowledge **from generation to generation**, could KI similarly be sequentially performed among a series of PLMs with growing sizes?
- (RQ3) As more and more PLMs with different **pre-training settings** emerge, how would different settings affect the performance of KI?
- (RQ4) Besides training a large PLM from scratch, when adapting an already trained large PLM to a **new domain**, how could smaller domain teachers benefit such a process?

Experimental Setting

- Architecture: RoBERTa
- Teacher Model (BASE): $RoBERTa_{BASE}$ Student Model (LARGE): $ROBERTa_{LARGE}$
- Training Data: Wikipedia + BookCorpus (3400M tokens)
- Training Steps: 125k
- Batch Size: 2048
- Evaluation: Perplexity (pre-training), GLUE performance (fine-tuning)
- LARGE v.s. BASE → LARGE

RQ1

- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- Results of Pre-training:
  - Training LARGE under our KI framework converges faster than the self-learning baseline (27.3% FLOPs saved)



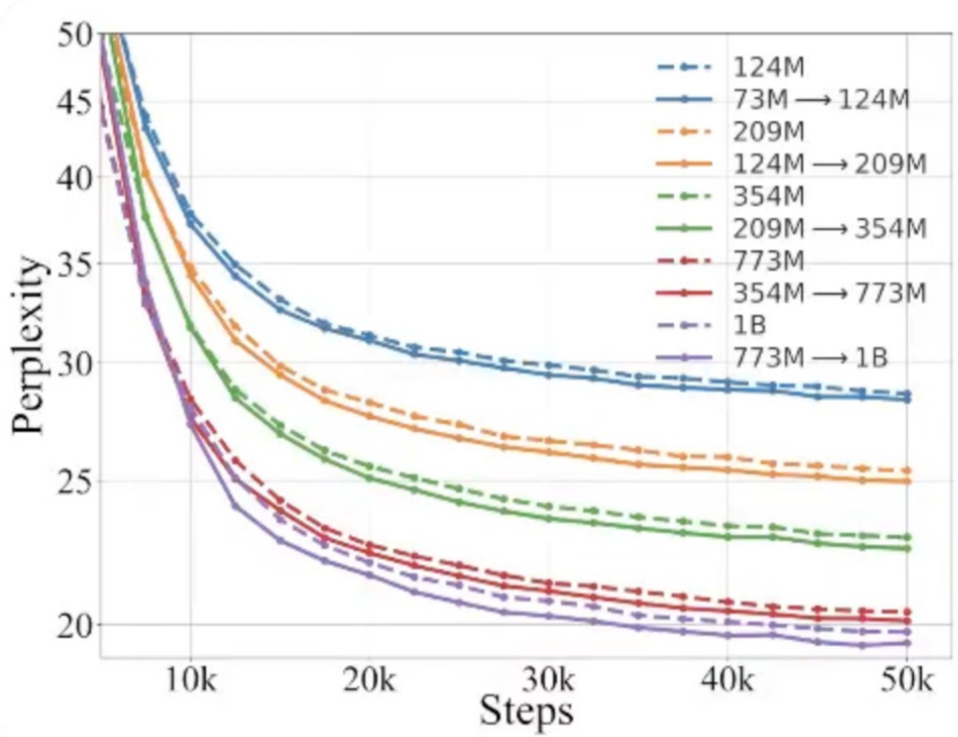
RQ1

- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- Results of Fine-training:
  - LARGE trained under KI framework achieves better performance than the baseline on downstream tasks at each step

Step	Model	CoLA	MNLI	QNLI	RTE	SST-2	STS-B	MRPC	QQP	Avg
5k	$\text{RoBERTa}_{\text{LARGE}}$	0.0	73.5	81.7	53.0	81.7	45.8	71.4	87.5	61.8
	$\text{RoBERTa}_{\text{BASE}} \rightarrow \text{RoBERTa}_{\text{LARGE}}$	**17.4**	**75.8**	**83.4**	**54.7**	**85.7**	**72.0**	**72.6**	**88.6**	**68.8**
45k	$\text{RoBERTa}_{\text{LARGE}}$	61.8	84.9	91.7	63.4	92.9	88.6	87.7	**91.5**	82.8
	$\text{RoBERTa}_{\text{BASE}} \rightarrow \text{RoBERTa}_{\text{LARGE}}$	**64.3**	**85.9**	**92.2**	**75.3**	**93.2**	**89.3**	**89.4**	91.5	**85.2**
85k	$\text{RoBERTa}_{\text{LARGE}}$	64.5	86.8	92.7	69.7	93.5	89.9	89.7	91.7	84.8
	$\text{RoBERTa}_{\text{BASE}} \rightarrow \text{RoBERTa}_{\text{LARGE}}$	**65.7**	**87.2**	**93.0**	**77.0**	**94.3**	**90.0**	**90.4**	**91.8**	**86.2**
125k	$\text{RoBERTa}_{\text{LARGE}}$	64.3	87.1	**93.2**	73.4	94.1	90.3	**90.1**	91.8	85.5
	$\text{RoBERTa}_{\text{BASE}} \rightarrow \text{RoBERTa}_{\text{LARGE}}$	**67.7**	**87.7**	93.1	**74.9**	**94.8**	**90.6**	88.2	**91.9**	**86.1**

RQ1

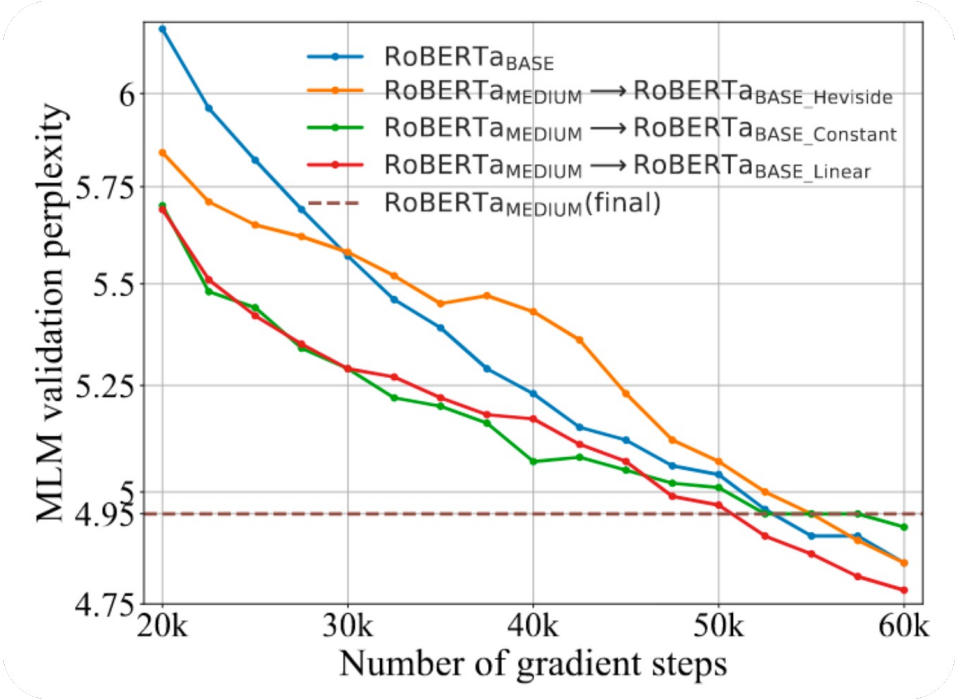
- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- Results of Pre-training:
  - The conclusion also holds for GPT across various sizes



Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{FFN}	lr (bs $= 2,048$)
$\text{GPT}_{73\text{M}}$	73M	9	576	12	3072	5.0×10^{-4}
$\text{GPT}_{124\text{M}}$	124M	12	768	12	3072	5.0×10^{-4}
$\text{GPT}_{209\text{M}}$	209M	18	864	12	3600	4.0×10^{-4}
$\text{GPT}_{354\text{M}}$	354M	24	1024	16	4096	3.5×10^{-4}
$\text{GPT}_{773\text{M}}$	773M	36	1280	20	5120	3.0×10^{-4}
$\text{GPT}_{1\text{B}}$	1068M	40	1440	20	5760	2.5×10^{-4}

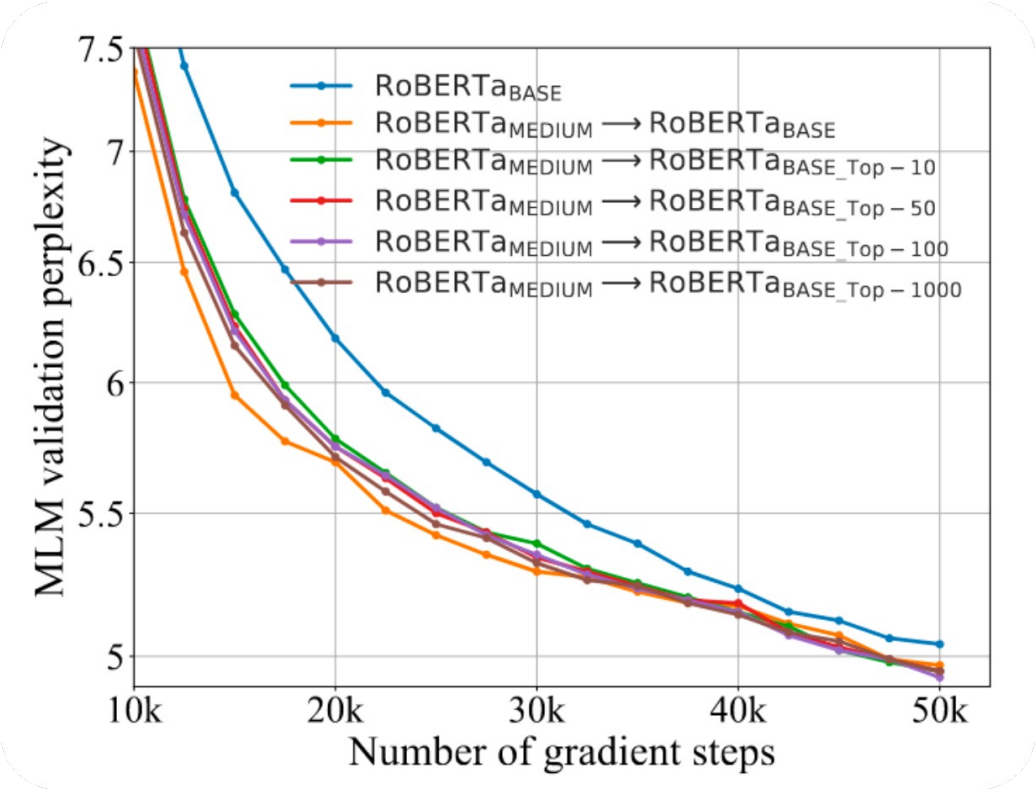
RQ1

- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- Effects of Inheritance Rate
  - Annealing at first is necessary
  - Teacher-guided learning is redundant after the student surpasses the teacher



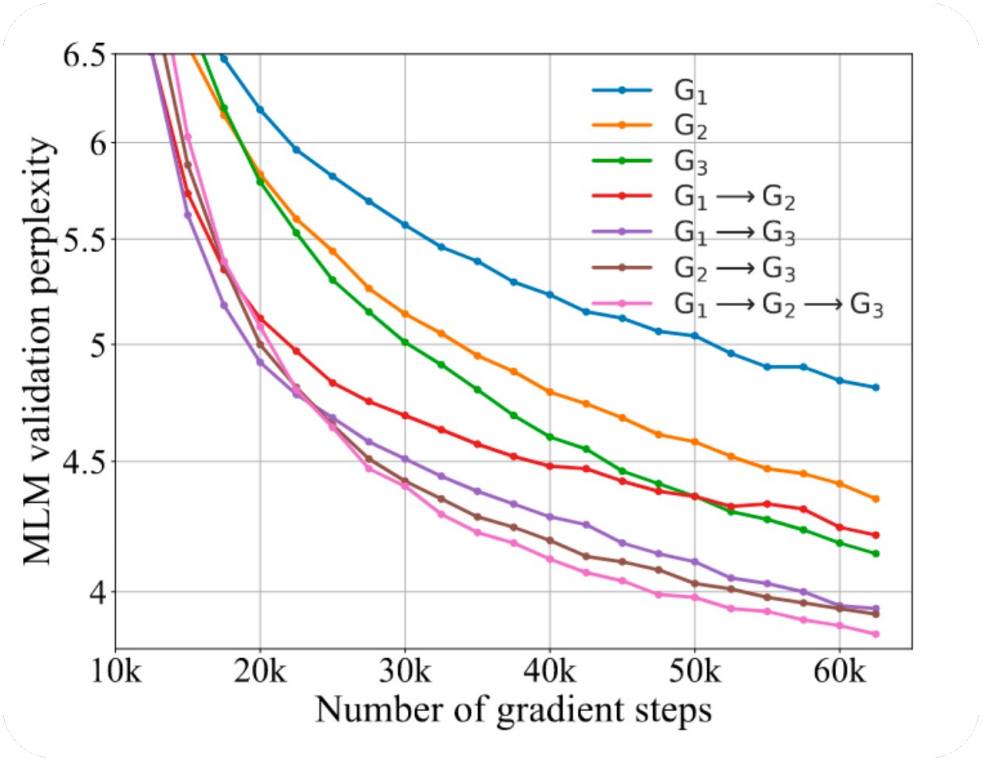
RQ1

- (RQ1) Could **distilling knowledge** from an existing trained PLM benefit large PLMs' training from scratch?
- Saving Storage Space with Top-K Logits
  - Top-K logits contain the vast majority of information



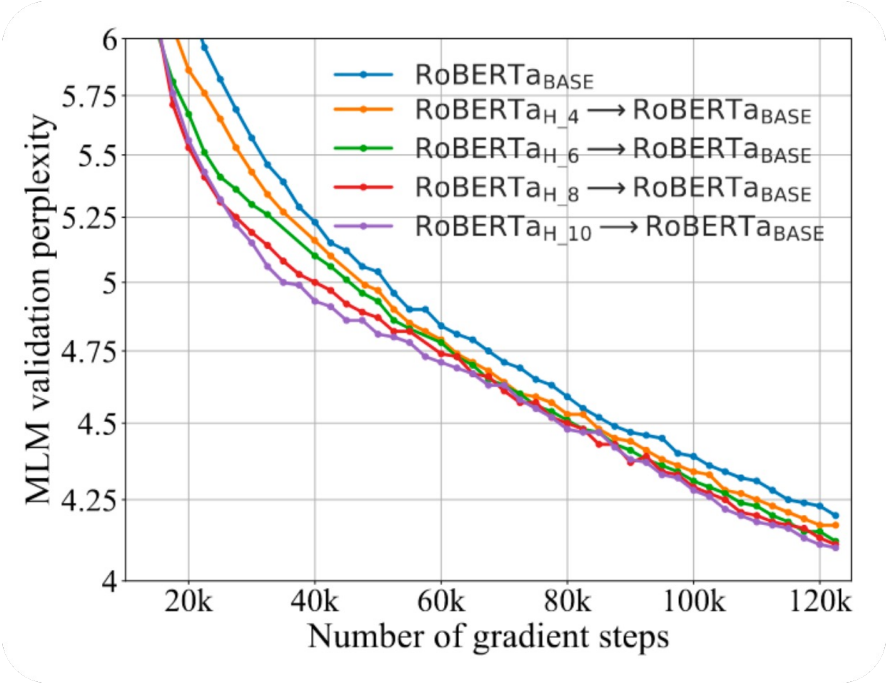
RQ2

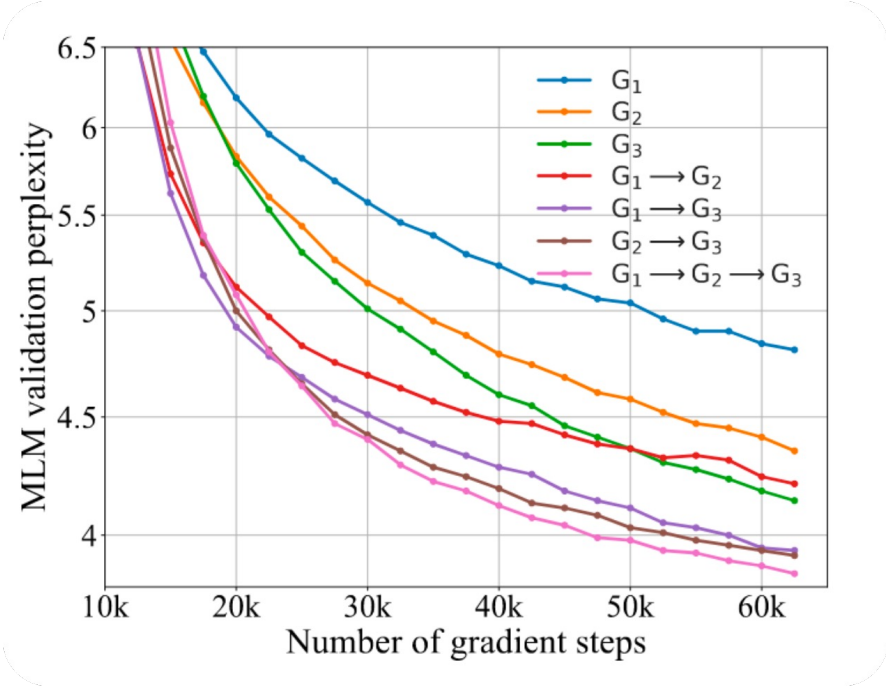
- (RQ2) Considering human beings are able to hand down knowledge **from generation to generation**, could KI similarly be sequentially performed among a series of PLMs with growing sizes?
- Knowledge Inheritance over Generations
  G_1(BASE, 125M), G_2(BASE_PLUS, 211M) G_3(LARGE, 355M)
- (1) Self-learning (G_1, G_2, G_3)
- (2) KI over two generations ($G_1 \rightarrow G_2$, $G_2 \rightarrow G_3$, $G_2 \rightarrow G_3$)
- (3) KI over three generations ($G_1 \rightarrow G_2 \rightarrow G_3$)



RQ3

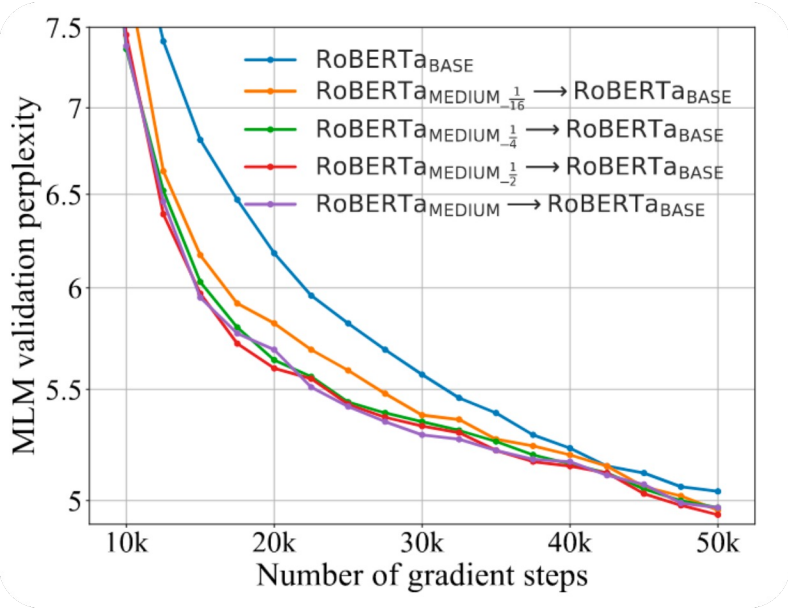
- (RQ3) As more and more PLMs with different **pre-training settings** emerge, how would different settings affect the performance of KI?
- Effects of Model Architecture
  - Choosing a wider / deeper teacher, which has lower validation PPL, further accelerates ML's convergence

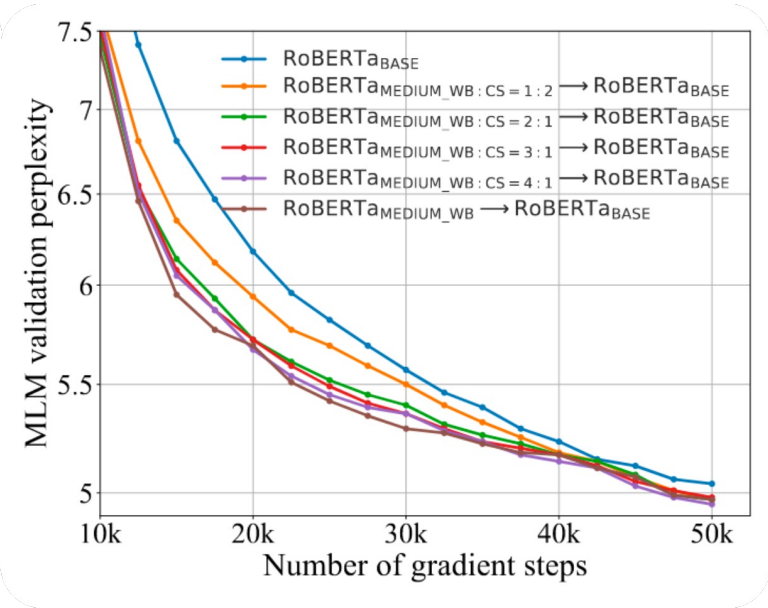


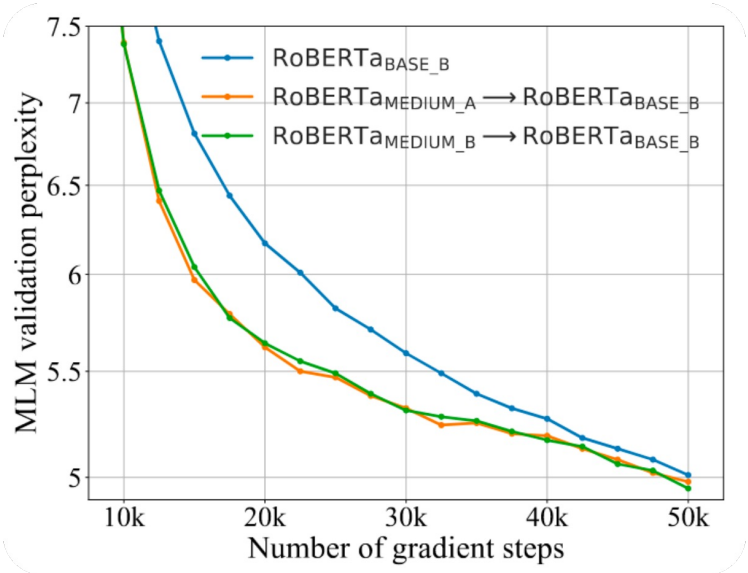


RQ3

- (RQ3) As more and more PLMs with different **pre-training settings** emerge, how would different settings affect the performance of KI?
- Effects of Data Size / Domain / Privacy
  - PLMs can image the big from the small for in-domain data
  - Inheriting on similar domain improves performance
  - Data privacy is not important if the same domain is ensured







RQ4

- (RQ4) Besides training a large PLM from scratch, when adapting an already trained large PLM to a **new domain**, how could smaller domain teachers benefit such a process?
- KI is more training-efficient and data-efficient
- Catastrophic Forgetting on the Source Domain

N_{tokens}		3, 400M		200M		100M		40M		20M	
Metrics		F1	PPL	F1	PPL	F1	PPL	F1	PPL	F1	PPL
CS	SL	69.8	3.12	71.7	3.17	71.4	3.24	68.3	3.51	67.5	4.07
	KI	**72.9**	**3.06**	**72.6**	**3.09**	**71.9**	**3.11**	**71.1**	**3.21**	**70.8**	**3.37**
BIO	SL	84.0	2.67	82.8	2.72	83.2	2.83	83.3	3.16	82.7	3.81
	KI	**84.5**	**2.65**	**83.4**	**2.66**	**83.9**	**2.69**	**83.6**	**2.82**	**83.5**	**3.01**

Domain	**Strategy**	3, 400M	200M	100M	40M
CS	SL	6.71	7.01	7.39	8.77
	KI	8.63	9.39	9.48	9.87
BIO	SL	7.29	6.61	8.16	10.34
	KI	10.74	10.78	10.93	11.66

Conclusion

- Propose a general knowledge inheritance framework that leverages previously trained PLMs for training larger ones
- Conduct sufficient empirical studies to demonstrate its feasibility in knowledge transfer, domain adaptation, etc.
- Comprehensively analyze various pre-training settings of the teacher model

bert2BERT: Towards Reusable Pretrained Language Models

Cheng Chen[1†], Yichun Yin[2], Lifeng Shang[2], Xin Jiang[2], Yujia Qin[1], Fengyu Wang[1], Zhi Wang[3,4‡], Xiao Chen[2], Zhiyuan Liu[1], Qun Liu[2]

[1]Department of Computer Science and Technology, Tsinghua University

[2]Huawei Noah's Ark Lab, [3]Tsinghua Shenzhen International Graduate School

[4]Peng Cheng Laboratory

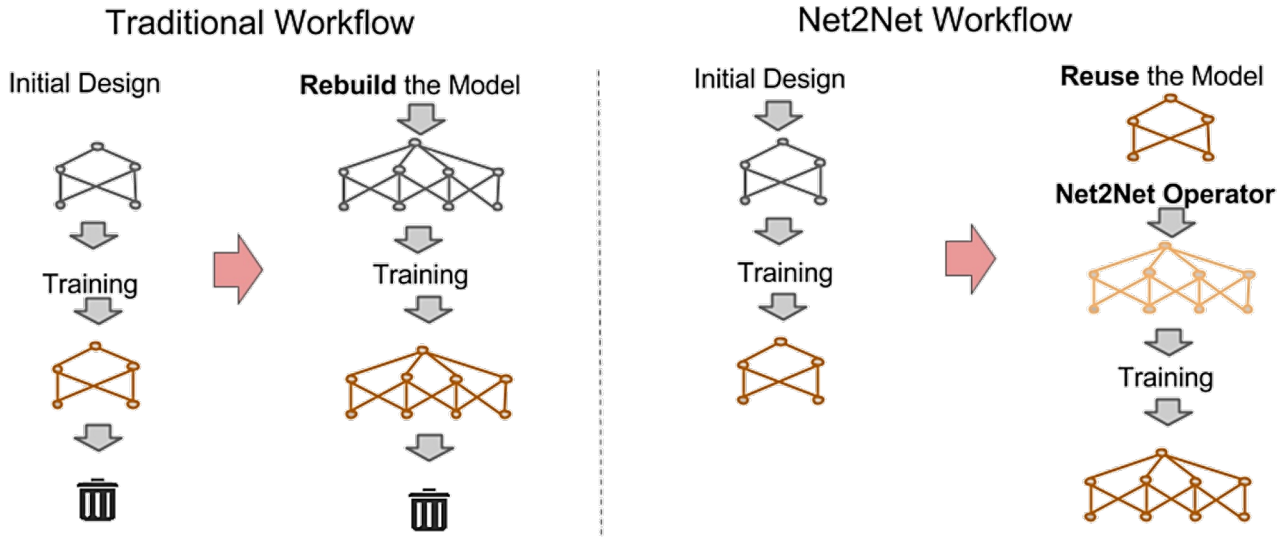
{c-chen19,qyj20,wangfy20}@mails.tsinghua.edu.cn

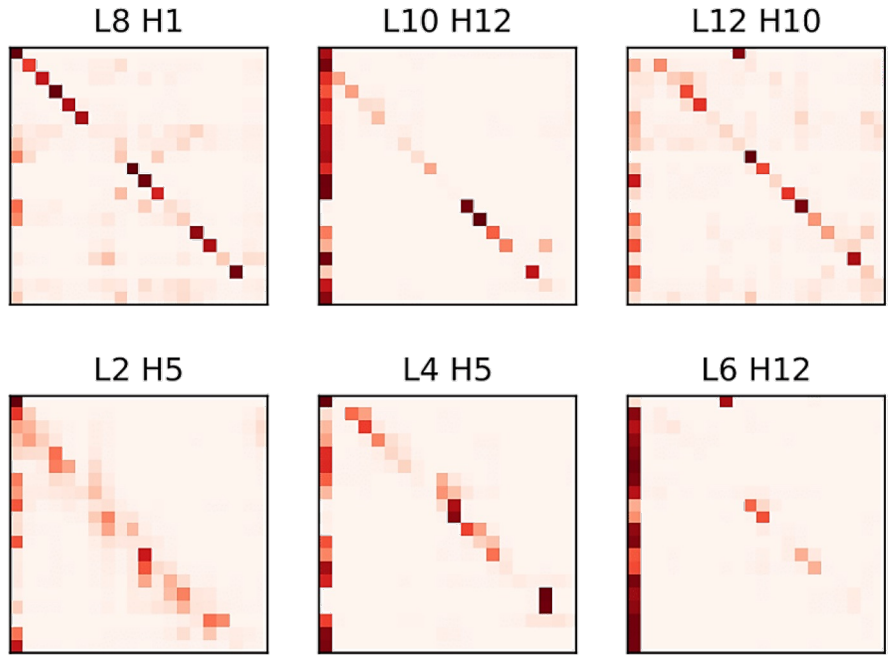
{yinyichun,shang.lifeng,jiang.xin,chen.xiao2,qun.liu}@huawei.com

wangzhi@sz.tsinghua.edu.cn,liuzy@tsinghua.edu.cn

Motivation

- Related work
  - Net2Net[1] : use **function-preserving transformation** to make neural networks reusable.
  - StackingBert[2] : different layers have some similar knowledge

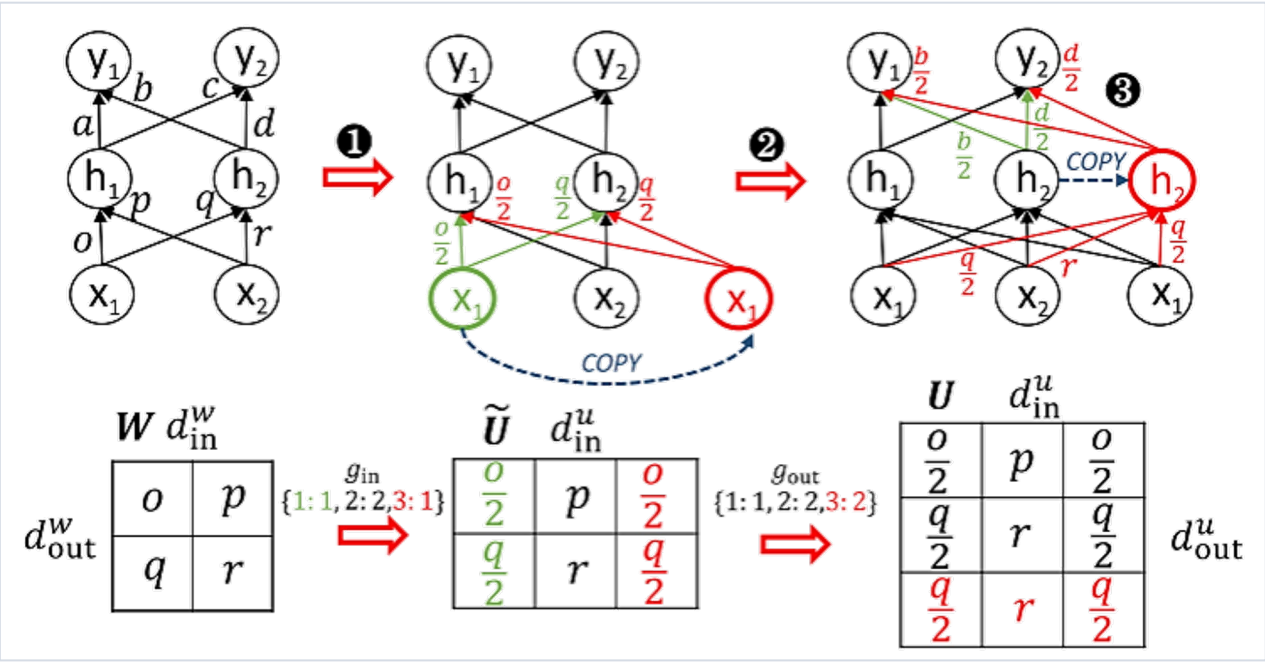




Method

- Objective : $S(L^s, D^s) \rightarrow T(L^t, D^t)$
- Parameter initialization
  - function-preserving initialization (FPI), ensures that the initialized large model has almost the same behavior as the small model
  - advanced knowledge initialization (AKI), leads to a faster convergence rate and achieves higher training efficiency
- Two-stage training strategy
  - train sub-models with different layers
  - full-model training

FPI



$$g_{\text{in}}(i) = \begin{cases} i & i \in [1, d^w_{\text{in}}] \\ f(\{1, 2, ..., d^w_{\text{in}}\}) & i \in (d^w_{\text{in}}, d^u_{\text{in}}], \end{cases} \tag{5}$$

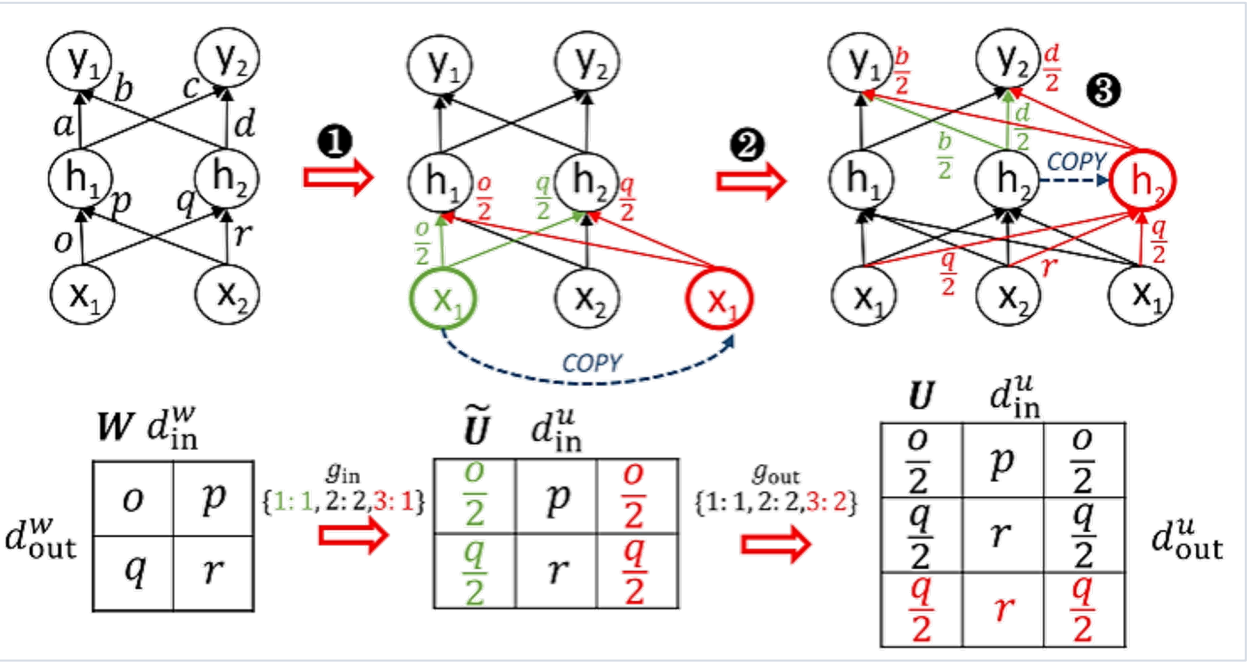
$$g_{\text{out}}(j) = \begin{cases} j & j \in [1, d^w_{\text{out}}] \\ f(\{1, 2, ..., d^w_{\text{out}}\}) & j \in (d^w_{\text{out}}, d^u_{\text{out}}], \end{cases} \tag{6}$$

$$C_{g_{\text{in}}(i)} = \sum_{i'=1}^{d^u_{\text{in}}} \mathbb{I}(g_{\text{in}}(i') = g_{\text{in}}(i)) \tag{7}$$

$$\widetilde{\boldsymbol{U}}_{(i,*)} = \frac{1}{C_{g_{\text{in}}(i)}} \boldsymbol{W}_{(g_{\text{in}}(i),*)},$$

$$\boldsymbol{U}_{(*,j)} = \widetilde{\boldsymbol{U}}_{(*,g_{\text{out}}(j))}, \tag{8}$$

FPI



Embedding

$$\boldsymbol{U}^E_{(*,j)} = \boldsymbol{W}^E_{(*,g^e_{\text{out}}(j))}. \tag{9}$$

MHA

$$\boldsymbol{U}^{Q|K|V} = \text{EXPN}(\boldsymbol{W}^{Q|K|V}; g^{q|k|v}_{\text{in}}, g^{q|k|v}_{\text{out}}),$$

$$\boldsymbol{U}^{O} = \text{EXPN}(\boldsymbol{W}^{O}; g^{o}_{\text{in}}, g^{o}_{\text{out}}).$$

$$g^{q|k|v}_{\text{out}}(j) = \begin{cases} j & j \in [1, a^s] \\ f(\{1, 2, ..., a^s\}) & j \in (a^s, a^t], \end{cases} \tag{10}$$

(11)

$$g^e_{\text{out}} = g^{q|k|v}_{\text{in}}\ ;\ g^{q|k|v}_{\text{out}} = g^o_{\text{in}};\ g^{q|k|v}_{\text{in}} = g^o_{\text{out}}$$

FFN

$$\boldsymbol{U}^1 = \text{EXPN}(\boldsymbol{W}^1; g^1_{\text{in}}, g^1_{\text{out}}),$$

$$\boldsymbol{U}^2 = \text{EXPN}(\boldsymbol{W}^2; g^2_{\text{in}}, g^2_{\text{out}}). \tag{12}$$

$$g^o_{\text{out}} = g^1_{\text{in}};\ g^1_{\text{out}} = g^2_{\text{in}};\ g^1_{\text{in}} = g^2_{\text{out}}$$

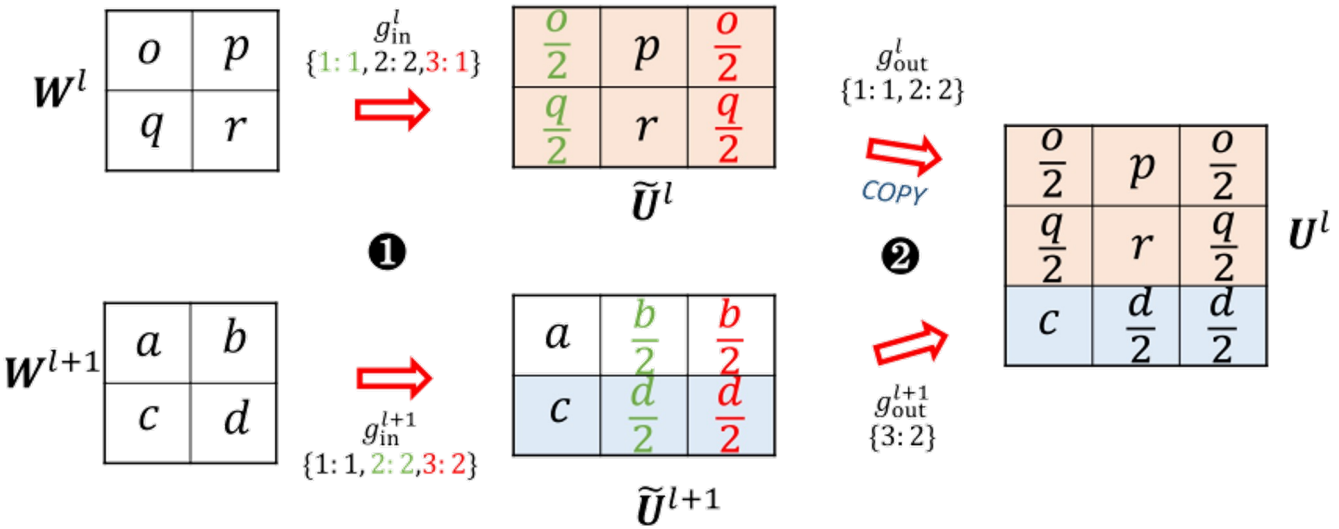
LN

$$\boldsymbol{U}^{\text{ln}}_j = \boldsymbol{W}^{\text{ln}}_{g^2_{\text{out}}(j)}. \tag{13}$$

$$\text{LayerNorm}(\boldsymbol{H}) = (\frac{\boldsymbol{H} - \mu_{\boldsymbol{H}}}{\sigma_{\boldsymbol{H}}}) \odot \boldsymbol{W}^{LN} + \boldsymbol{b}^{LN}$$

AKI

- Breaks FPI **symmetry** that hinders model convergence
- Upper-layer information can be used as **high-level** knowledge



$$\boldsymbol{U}^l = \mathrm{EXPN}(\boldsymbol{W}^l, \boldsymbol{W}^{l+1}; g_{\text{in}}, g_{\text{out}}). \tag{14}$$

$$C_{g^l_{\text{in}}(i)} = \sum_{i'=1}^{d^u_{\text{in}}} \mathbb{I}(g^l_{\text{in}}(i') = g^l_{\text{in}}(i))$$
$$\widetilde{\boldsymbol{U}}^l_{(i,*)} = \frac{1}{C_{g^l_{\text{in}}(i)}} \boldsymbol{W}^l_{(g^l_{\text{in}}(i),*)}. \tag{15}$$

$$\boldsymbol{U}^l_{(*,j)} = \begin{cases} \widetilde{\boldsymbol{U}}^l_{(*,j)} & j \in [1, d^s_{\text{out}}] \\ \widetilde{\boldsymbol{U}}^{l+1}_{(*,g^{l+1}_{\text{out}}(j))} & j \in (d^s_{\text{out}}, d^t_{\text{out}}], \end{cases} \tag{16}$$

Depth-wise expansion

Algorithm 1 bert2BERT initialization

Input: the target model $\mathcal{T}(L^t, D^t)$ and the source model $\mathcal{S}(L^s, D^s)$.

1: $\mathcal{T}_1(L^s, D^t) \leftarrow$ do AKI or FPI with $\mathcal{S}(L^s, D^s)$
2: $k \leftarrow \lfloor L^t / L^s \rfloor$
3: **for** $t = 2 \rightarrow k$ **do**
4: $\quad \mathcal{T}_t(L^s \cdot t, D^t) \leftarrow$ stack $\mathcal{T}_1$ on top of $\mathcal{T}_{t-1}$
5: **end for**
6: $\mathcal{T} \leftarrow$ stack top $L^t - L^s \cdot k$ layers of $\mathcal{T}_1$.

Output: the initialized model $\mathcal{T}(L^t, D^t)$

Two-stage Pre-training

- Stage1: Train sub-modules
- Stage2: Train full-model

Algorithm 2 Two-stage Pre-training

Input: the initialized model $\mathcal{T}$, large-scale unsupervised dataset $\mathcal{D}$, the epoch number of sub-model training E_b and the epoch number of whole training process E, the layer number l_b.

Construct sub-models and these models have the layer numbers of $\{l_b, 2 \cdot l_b, \ldots, L^t\}$.
for $e = 1 \rightarrow E_b$ **do**
  for $batch$ in D **do**
    $\mathcal{T}' \leftarrow$ sample one sub-model.
    Perform forward and backward of $\mathcal{T}'$.
    Update only top l_b layers of $\mathcal{T}'$.
  end for
end for

Algorithm 2 Two-stage Pre-training

Input: the initialized model $\mathcal{T}$, large-scale unsupervised dataset $\mathcal{D}$, the epoch number of sub-model training E_b and the epoch number of whole training process E, the layer number l_b.

for $e = E_b \rightarrow E$ **do**
  for $batch$ in D **do**
    Perform forward and backward of $\mathcal{T}$.
    Update whole model $\mathcal{T}$.
  end for
end for

Output: the pre-trained model $\mathcal{T}$

Experiment Setting

- Pre-training Details
  - Datasets : English Wikipedia , Toronto Book
  - Two-stage : E_b=5, l_b=3
  - Target model : $T(12, 768)$
  - Source model : $S(12, 512)$
- Fine-tuning Details
  - Downstream task : GLUE benchmark, SQuADv1.1
- Baseline
  - DirectCopy, Stack-BERT, MSLT

Result

Model	FLOPs (× 1e19)	Ratio (Cost Saving)	SQuADv1.1 (F1)	SST-2 (Acc)	MNLI (Acc)	MRPC (Acc)	CoLA (Mcc)	QNLI (Acc)	QQP (Acc)	STS-B (Acc)	Avg.
$\text{BERT}_{\text{BASE}}$ (Google)	-	-	88.4(0.1)	93.6(0.2)	84.7(0.1)	87.9(0.9)	59.6(1.5)	91.6(0.1)	91.4(0.1)	89.6(0.5)	85.8(0.1)
$\text{BERT}_{\text{BASE}}$ † (Ours)	7.3	0%	89.6(0.1)	92.7(0.2)	84.6(0.2)	88.6(0.5)	57.3(4.0)	90.6(0.7)	90.6(0.1)	89.9(0.3)	85.5(0.5)
Progressive Training											
MSLT†	6.5	10.7%	90.4(0.2)	92.9(0.2)	85.1(0.2)	87.9(2.1)	55.6(4.1)	90.7(0.2)	90.6(0.2)	88.2(0.6)	85.2(0.7)
StackBERT†	5.5	24.3%	90.4(0.2)	92.6(0.4)	85.3(0.1)	88.2(1.0)	63.2(0.9)	91.0(0.4)	91.0(0.1)	86.7(0.7)	86.0(0.2)
bert2BERT : $\mathcal{S}(12, 512) \rightarrow \mathcal{T}(12, 768)$											
DirectCopy	6.4	12.2%	89.8(0.2)	92.9(0.3)	84.7(0.2)	86.2(0.6)	62.2(0.7)	90.2(0.6)	90.4(0.1)	89.2(0.1)	85.7(0.1)
FPI	5.1	30.4%	90.0(0.2)	92.6(0.4)	85.2(0.1)	87.1(0.5)	61.5(0.9)	90.9(0.6)	90.8(0.2)	89.7(0.2)	86.0(0.1)
AKI	4.5	38.4%	90.4(0.1)	92.5(0.4)	85.3(0.4)	87.8(0.9)	61.0(1.4)	91.2(0.2)	90.5(0.1)	89.5(0.2)	86.0(0.2)
bert2BERT	**4.0**	**45.2%**	90.0(0.2)	92.9(0.1)	85.1(0.1)	87.7(0.7)	60.0(1.2)	90.5(0.8)	90.4(0.1)	89.2(0.2)	85.7(0.4)

Result

Settings	Model	FLOPs (× 1e19)	Ratio (Saving)	Loss (MLM)	Avg.
$\mathcal{S}(6, 512)$	DirectCopy	7.3	0%	1.440	89.1
	bert2BERT	5.6	23.3%	1.435	89.3
$\mathcal{S}(8, 512)$	bert2BERT	4.6	36.8%	1.435	89.2
$\mathcal{S}(10, 512)$	bert2BERT	4.2	42.7%	1.434	89.1

Table 3: bert2BERT with smaller source model. Avg means the average score of SST-2/MNLI/SQuADv1.1.

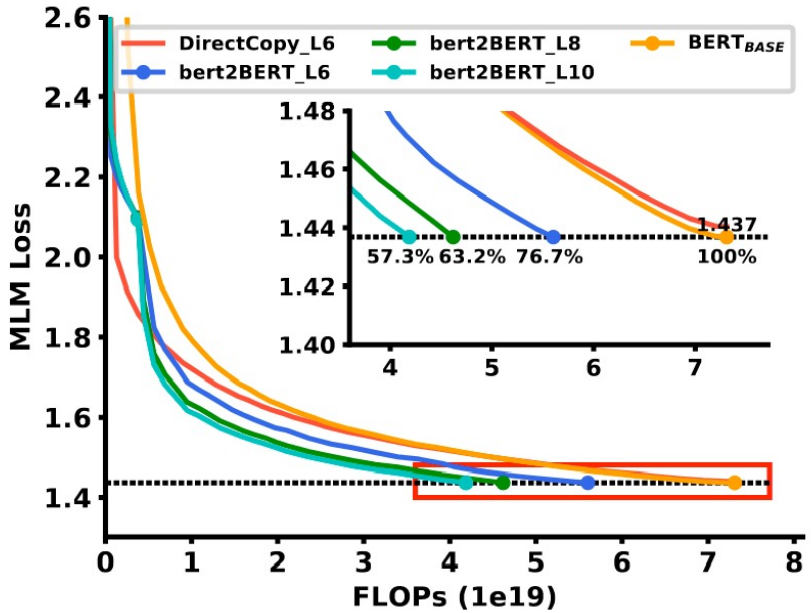
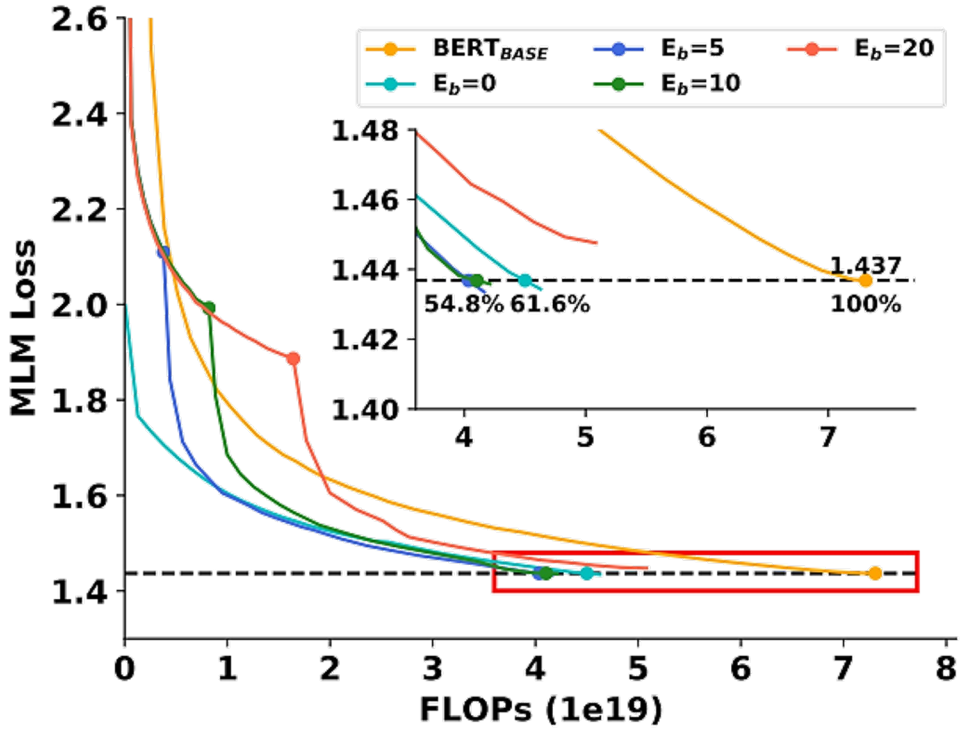


Figure 6: Loss curves of bert2BERT and baselines with smaller source models.

Result

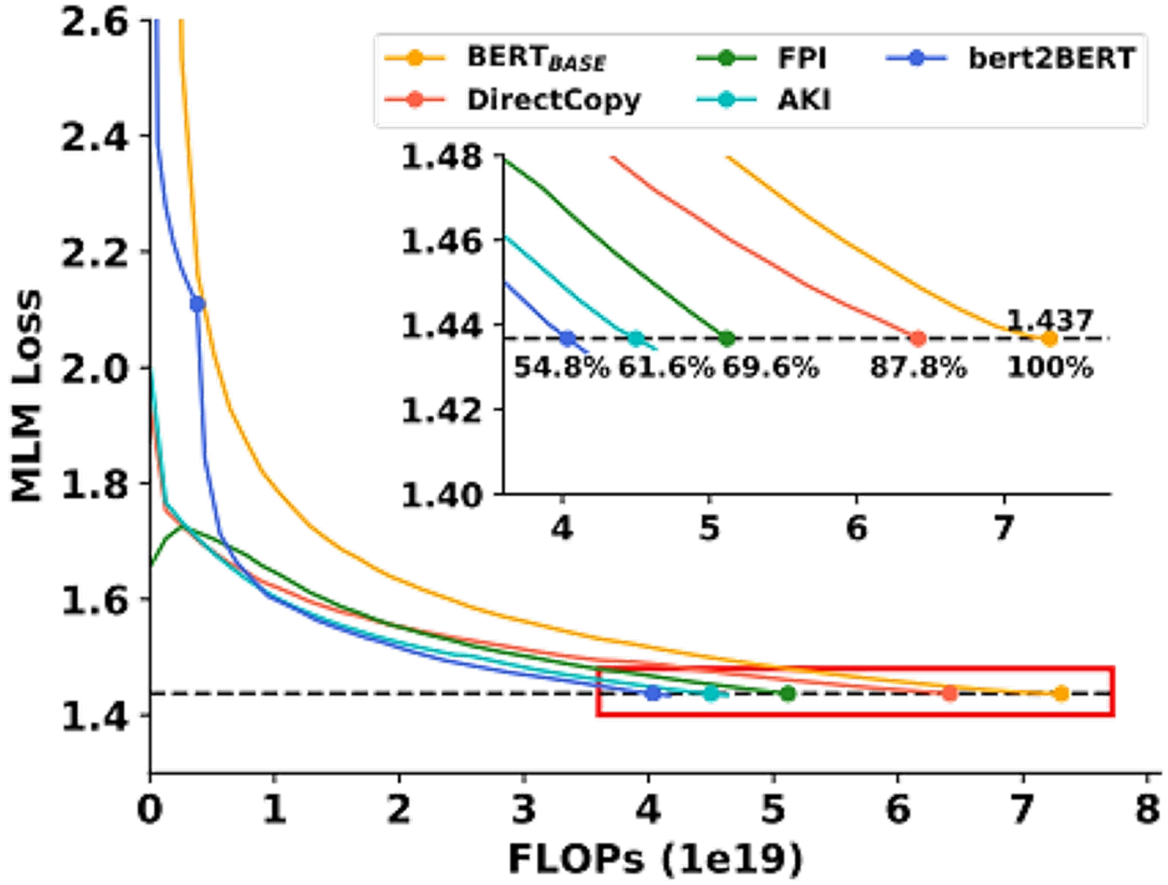
Model	FLOPs (× 1e19)	Ratio (Cost Saving)	Avg.
bert2BERT : $\mathcal{S}(12, 512) \rightarrow \mathcal{T}(12, 768)$			
bert2BERT ($E_b = 0$)	4.5	38.4%	86.0
bert2BERT ($E_b = 5$)	**4.0**	**45.2%**	85.7
bert2BERT ($E_b = 10$)	4.1	43.9%	85.3
bert2BERT ($E_b = 20$)	5.4	25.4%	84.3



Result

Model	FLOPs ($\times$ 1e19)	PTB (w/o FT)	WikiText-2 (w/o FT)	WikiText103 (w/o FT)
bert2BERT : $\mathcal{S}(12, 512) \rightarrow \mathcal{T}(12, 768)$				
GPT	4.9	133.8	47.0	53.5
bert2BERT	**2.6** (47% $\downarrow$)	132.1	47.9	53.0

Result


BERT$_{BASE}$
FPI
bert2BERT
DirectCopy
AKI
MLM Loss
FLOPs (1e19)
1.437
54.8%
61.6%
69.6%
87.8%
100%

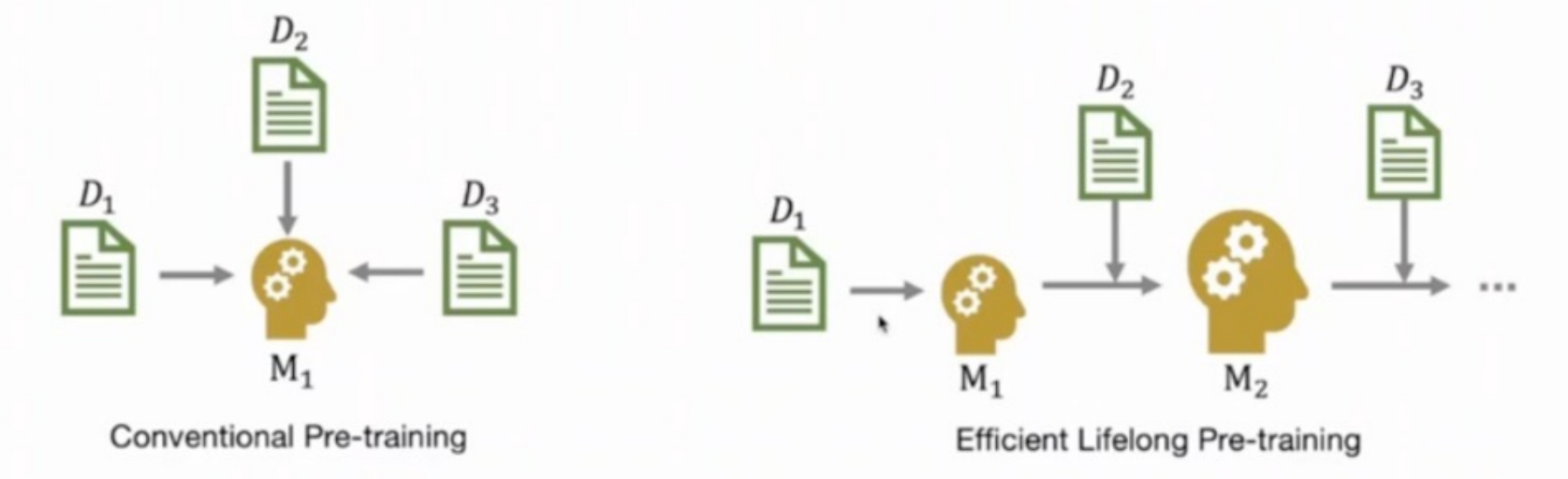
ELLE: Efficient Lifelong Pre-training for Emerging Data

Yujia Qin[1,2,3*], **Jiajie Zhang**[1,2,3*], **Yankai Lin**[4], **Zhiyuan Liu**[1,2,3,5,6†], **Peng Li**[7‡], **Maosong Sun**[1,2,3,5,6,8†], **Jie Zhou**[4]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Beijing National Research Center for Information Science and Technology
[3]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[4]Pattern Recognition Center, WeChat AI, Tencent Inc.
[5]International Innovation Center of Tsinghua University, Shanghai, China
[6]Beijing Academy of Artificial Intelligence
[7]Institute for AI Industry Research (AIR), Tsinghua University, China.
[8]Jiangsu Collaborative Innovation Center for Language Ability, Xuzhou, China
{qyj20,jiajie-z19}@mails.tsinghua.edu.cn

Motivation

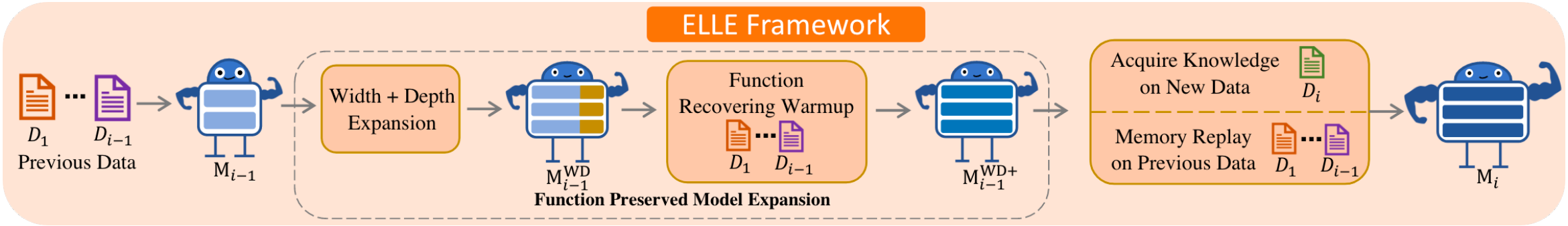
- Question: with limited computational resources, **how can we efficiently adapt PLMs in a lifelong manner**?
- **Efficient Lifelong Pre-training**: PLMs continually absorb knowledge from emerging data, and in the meantime, mitigate the catastrophic forgetting



Motivation

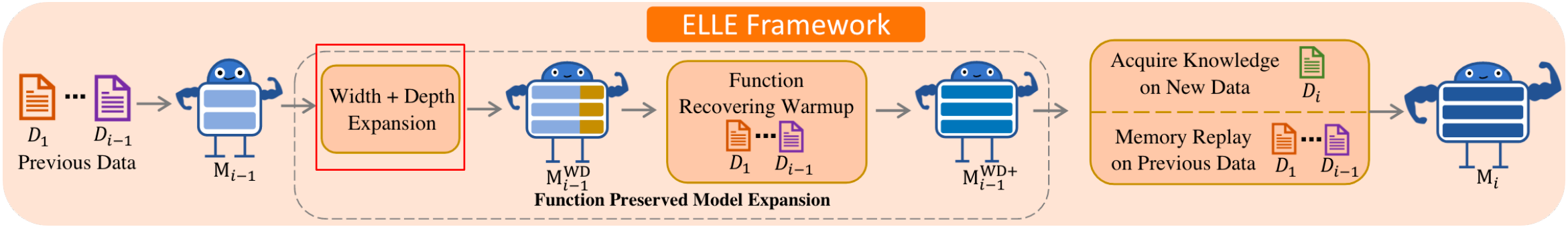
- Two challenges for efficient lifelong pre-training
  - (1) **Efficient knowledge growth**: with the data scale growing, packing more knowledge into a fixed-sized PLM becomes increasingly hard
  - (2) **Proper knowledge stimulation**: during pre-training, various knowledge from all domains is packed into PLMs hastily. However, a certain downstream task may largely require the knowledge from a specific domain
- We propose ELLE, targeting at **E**fficient **L**ife**L**ong pre-training for **E**merging data

Methodology



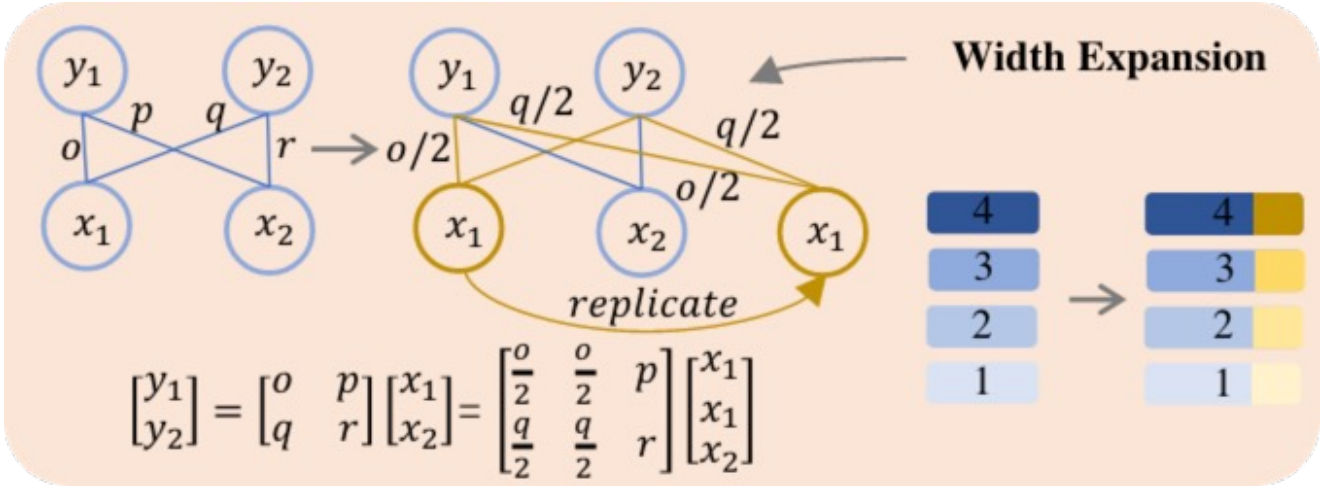
Methodology

- (1) **Function preserved model expansion**: flexibly expand an existing PLM's width and depth to improve the efficiency of knowledge acquisition



Methodology

- Width Expansion



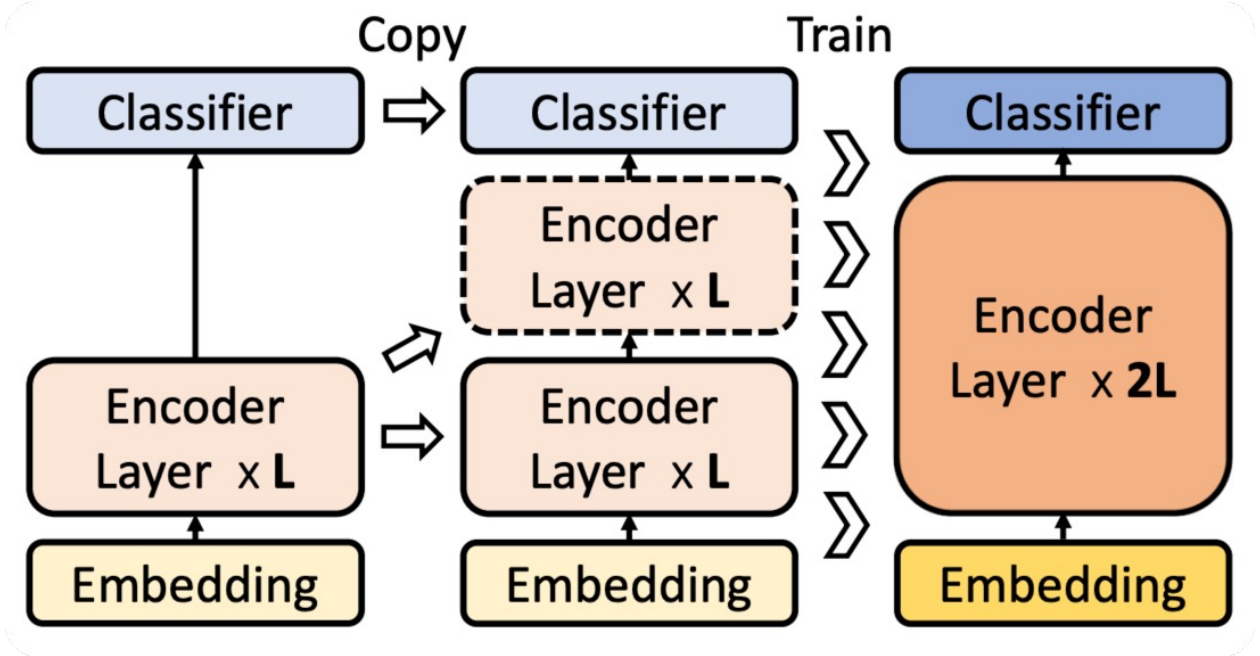
$$m(i) = \begin{cases} i & i \in [1, h_1] \\ U(\{1, \ldots, h_1\}) & i \in (h_1, h_1 + \Delta_{h_1}], \end{cases}$$

$$C_i = \sum_{i'=1}^{h_1+\Delta_{h_1}} \mathbb{I}(m(i') = m(i)),$$

$$\boldsymbol{W}'_{(i,*)} = \frac{1}{C_i} \cdot \boldsymbol{W}_{(m(i),*)} + \mathbb{I}(C_i > 1) \cdot \boldsymbol{\delta}_i,$$

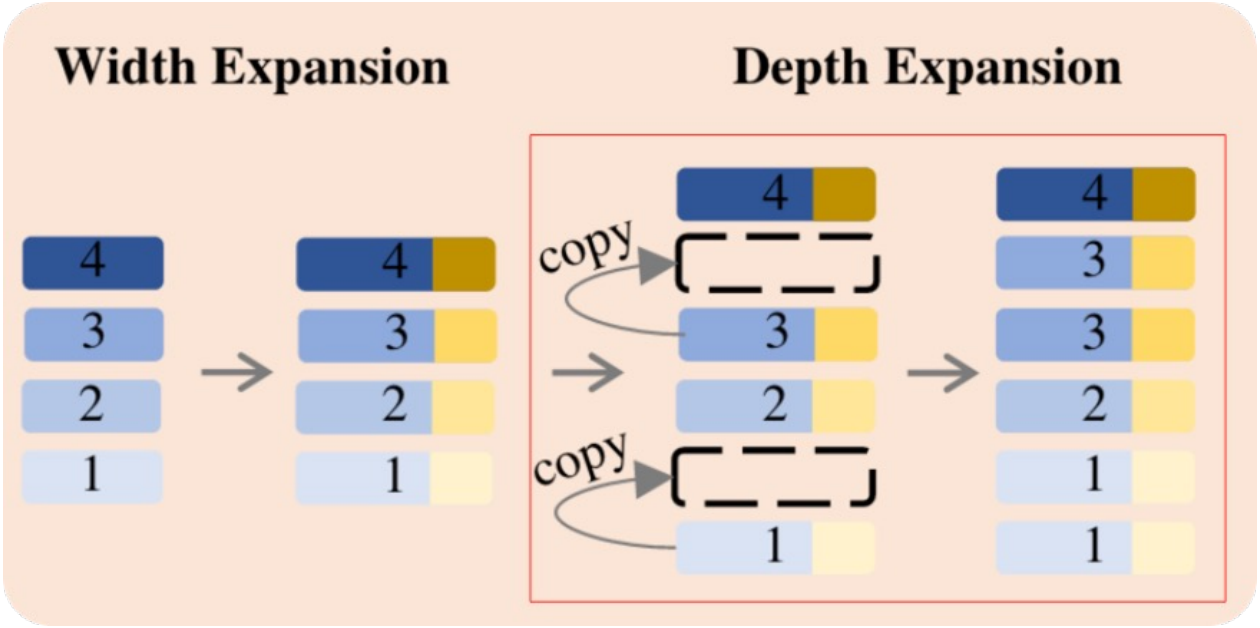
Methodology

- Depth Expansion(previous work)



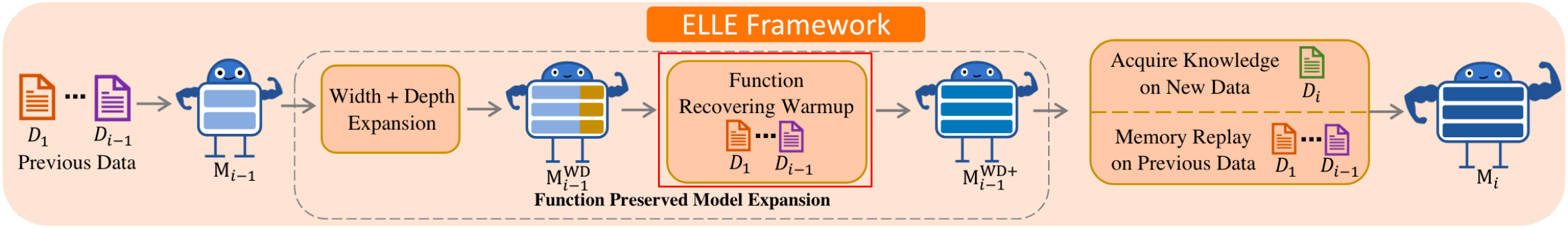
Methodology

- Depth Expansion



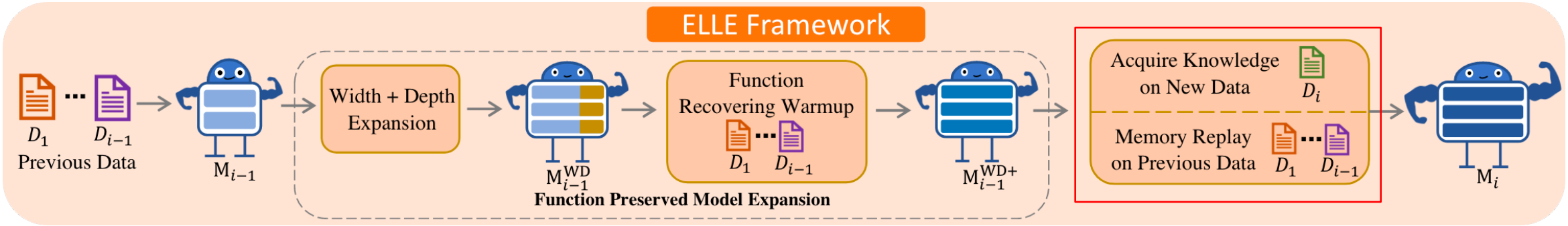
Methodology

- (2) **Function Recovering Warmup**: pre-train the initialized PLM on the previous corpora conserved in the memory to recover the language abilities lost during model expansion



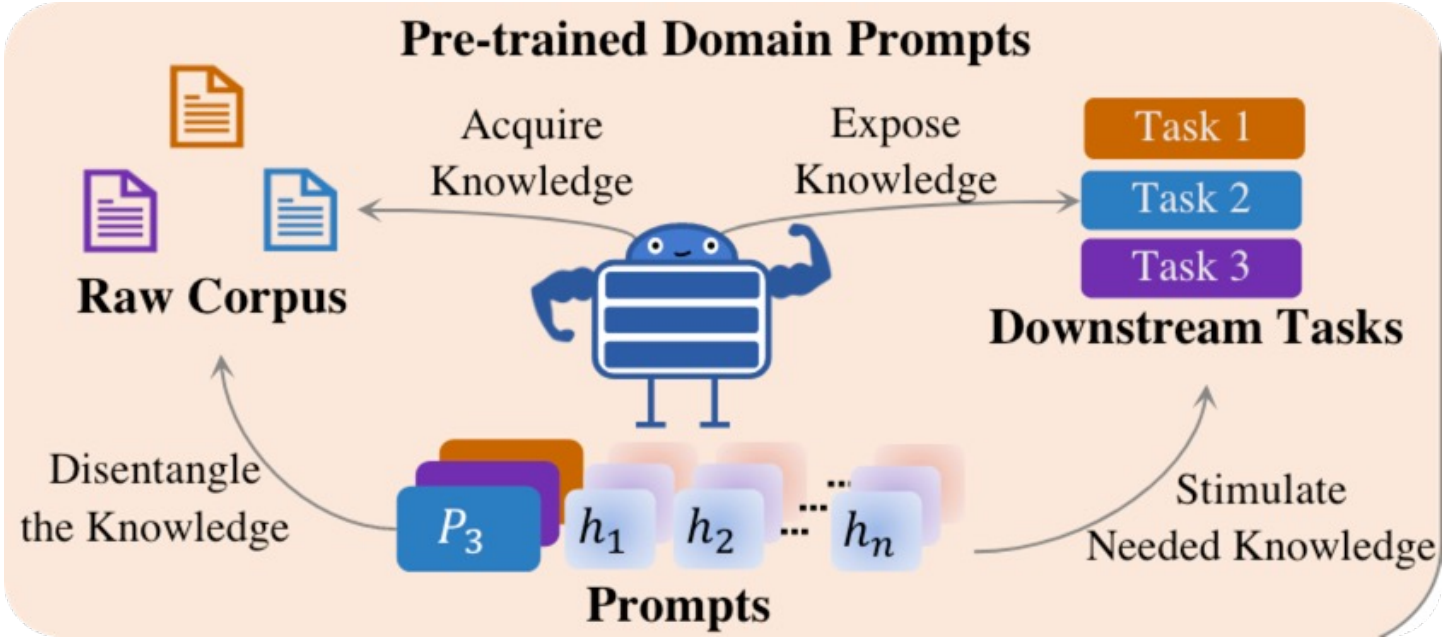
Methodology

- (3) **Acquiring New Knowledge Memory Replay**



Methodology

- (4) **Pre-trained domain prompts**: disentangle the versatile knowledge learned during pre-training and stimulate the proper knowledge for downstream tasks



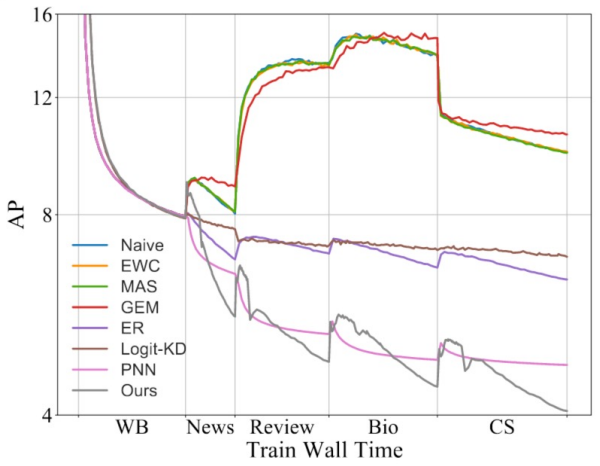
Experimental Setting

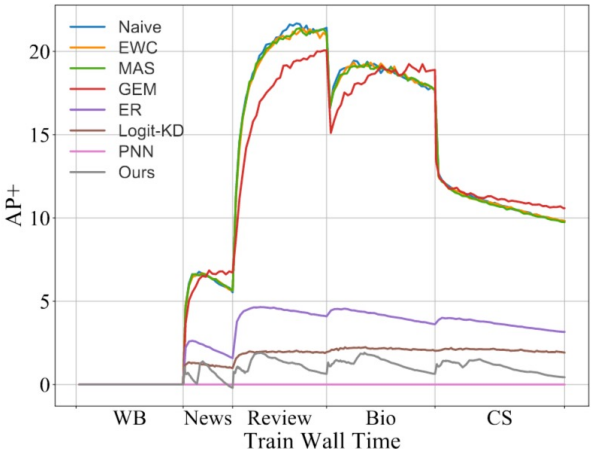
- Pre-train on 5 sequential domains
  - Wikipedia, News, Reviews, Biomedical and Computer Science
- Evaluate ELLE on both BERT and GPT architecture
- Growing PLMs from both (1) 30M to 125M and (2) 125M to 355M
- Metrics:
  - Pre-training: average perplexity (AP) & averaged increased perplexity (AP+)
    - $AP = \exp(\frac{1}{j}\sum_{i=1}^{j} logPPL_{T,i})$
    - $AP^{+} = \frac{1}{j-1}\sum_{i=1}^{j-1}(PPL_{T,i} - PPL_{i,i}^{f})$
  - Fine-tuning: downstream performance on each domain
    - MNLI, HYPERPARTISAN, HELPFULNESS, CHEMPROT, ACL-ARC

Experimental Results

- **Pre-training**
  - Lower average perplexity (AP) and less catastrophic forgetting (AP+)

Domain	WB		NS		REV		BIO		CS	
Metrics	AP	AP+	AP	AP+	AP	AP+	AP	AP+	AP	AP+
Growing from $\text{BERT}_{\text{L6\_D384}}$ *to* $\text{BERT}_{\text{L12\_D768}}$										
Naive (Lower Bound)	7.96	-	8.03	5.54	13.52	21.42	13.86	17.67	9.93	9.81
EWC	7.96	-	8.09	5.65	13.40	20.98	13.92	17.75	9.94	9.82
MAS	7.96	-	8.08	5.65	13.44	21.17	13.87	17.67	9.91	9.75
A-GEM	7.96	-	8.82	6.72	13.31	20.06	14.73	18.89	10.56	10.58
ER	7.96	-	6.85	1.59	6.99	4.09	6.66	3.62	6.39	3.16
Logit-KD	7.96	-	7.60	0.99	7.19	1.95	7.08	2.02	6.92	1.92
PNN	7.96	-	6.52	0.00	5.29	**0.00**	4.84	**0.00**	4.76	**0.00**
ELLE (ours)	**7.92**	-	**5.62**	**-0.20**	**4.81**	0.64	**4.41**	0.64	**4.06**	0.44
Growing from $\text{BERT}_{\text{L12\_D768}}$ *to* $\text{BERT}_{\text{L24\_D1024}}$										
ER	4.54	-	4.33	1.31	4.02	1.46	3.73	1.15	3.82	1.28
ELLE (ours)	**4.52**	-	**3.89**	**0.47**	**3.61**	**0.75**	**3.66**	**0.97**	**3.29**	**0.54**
Growing from $\text{GPT}_{\text{L6\_D384}}$ *to* $\text{GPT}_{\text{L12\_D768}}$										
Naive (Lower Bound)	46.54	-	52.91	37.96	81.28	177.22	94.44	160.51	60.64	80.48
MAS	46.54	-	53.12	38.44	81.23	177.20	93.21	157.93	60.62	80.28
ER	46.54	-	44.49	12.42	35.46	21.78	33.24	23.38	31.94	19.83
Logit-KD	46.54	-	48.93	5.41	37.60	9.97	34.60	11.74	33.67	11.19
PNN	46.54	-	39.90	**0.00**	26.84	**0.00**	**22.19**	**0.00**	21.43	**0.00**
ELLE (ours)	**46.50**	-	**36.84**	2.25	**25.60**	4.38	22.29	5.88	**20.49**	4.31





Experimental Results

- **Fine-tuning**
  - Better downstream performance on each domain

Domain	WB	NS	REV	BIO	CS	AVG
Growing from $\text{BERT}_{\text{L6\_D384}}$ *to* $\text{BERT}_{\text{L12\_D768}}$						
Naive	77.2	72.8	60.6	77.1	64.8	70.5
EWC	77.4	72.8	61.6	77.5	59.6	69.8
MAS	77.1	73.7	60.7	77.5	68.2	71.5
A-GEM	76.6	71.4	61.5	76.9	67.5	70.8
ER	77.6	72.2	61.9	78.3	63.5	70.7
Logit-KD	77.2	69.5	63.9	76.8	58.9	69.2
PNN	76.0	76.3	68.0	79.5	65.2	73.0
ELLE	**83.2**	**81.8**	**68.5**	**82.9**	**72.7**	**77.8**
Growing from $\text{BERT}_{\text{L12\_D768}}$ *to* $\text{BERT}_{\text{L24\_D1024}}$						
ER	84.7	83.3	68.0	82.7	71.4	78.0
ELLE	**86.3**	**90.4**	**70.5**	**84.2**	**73.8**	**81.0**

Experimental Results

- **Ablation Study (pre-training)**

Domain					WB		Ns		Rev		Bio		CS	
WE	**DE**	**FRW**	δ_N	**PT**	AP	AP+	AP	AP+	AP	AP+	AP	AP+	AP	AP+
					7.96	-	6.85	1.59	6.99	4.09	6.66	3.62	6.39	3.16
✓		✓			7.96	-	6.23	0.78	5.34	1.42	4.98	1.20	4.48	0.89
	✓	✓			7.96	-	5.81	0.03	5.49	1.43	5.16	1.32	4.79	0.94
✓	✓	✓			7.96	-	5.78	0.02	4.91	0.76	4.49	0.73	4.13	0.52
✓	✓				7.96	-	5.79	0.09	5.09	1.13	4.58	0.88	4.22	0.65
✓	✓	✓	✓		7.96	-	5.69	−0.13	4.85	0.67	4.45	0.69	4.09	0.47
✓	✓	✓	✓	✓	**7.92**	-	**5.62**	**-0.20**	**4.81**	**0.64**	**4.41**	**0.64**	**4.06**	**0.44**

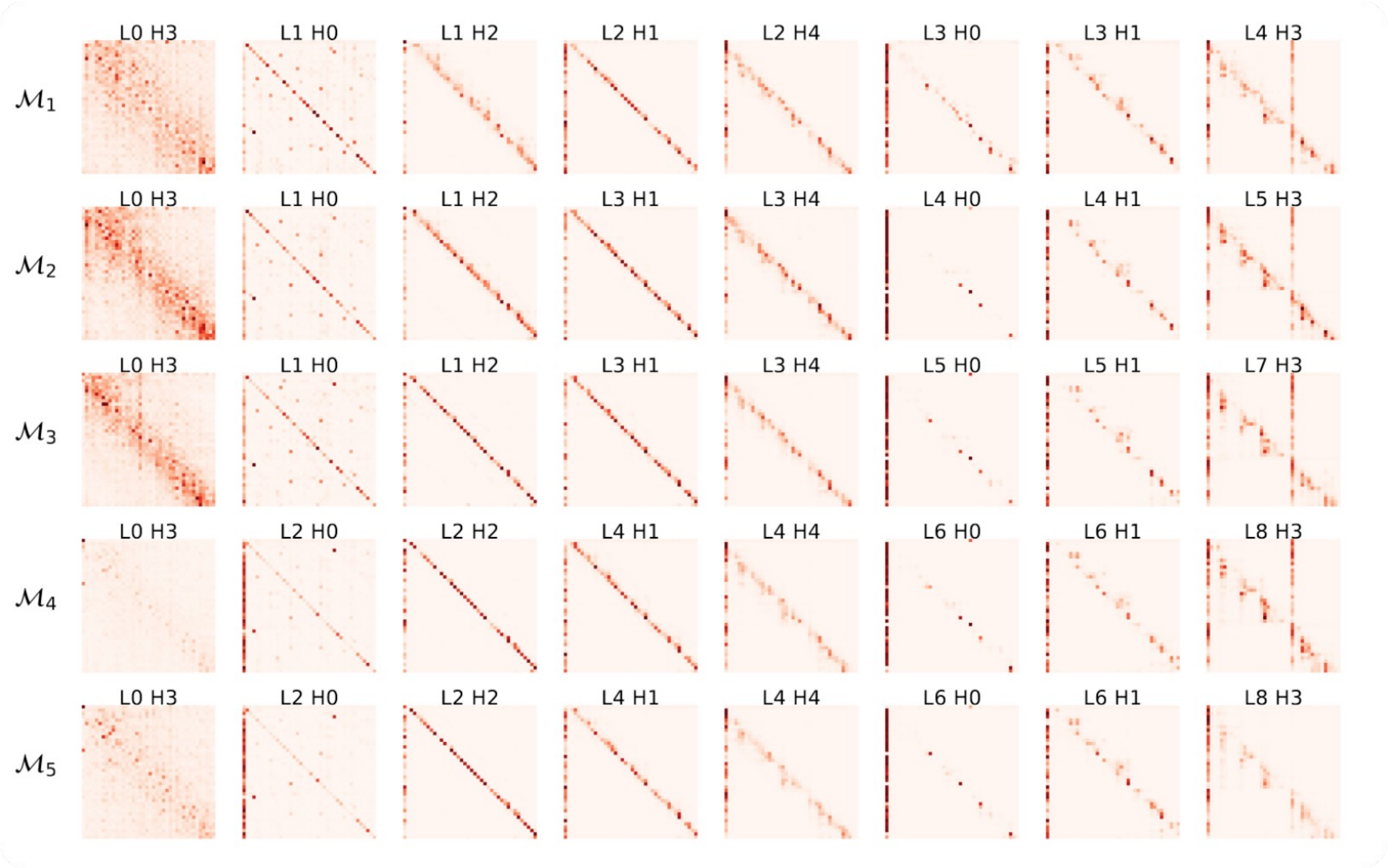
Experimental Results

- **Effects of Domain Prompts during Fine-tuning**

Domain	WB	NS	REV	BIO	CS	AVG
ELLE $-$ $\text{PT}_{\text{fine-tune}}$	82.9	79.9	67.0	82.1	67.7	75.9
ELLE $+$ $\neg\text{PT}_{\text{fine-tune}}$	83.1	80.6	68.1	81.7	70.8	76.9
ELLE	**83.2**	**81.8**	**68.5**	**82.9**	**72.7**	**77.8**

Experimental Results

- **Visualization of the Attention Patterns**



Thanks