Knowledge Inheritance for Pre-trained Language Models

Yujia Qin[1,2,3], **Yankai Lin**[4], **Jing Yi**[1,2,3], **Jiajie Zhang**[1,2,3], **Xu Han**[1,2,3], **Zhengyan Zhang**[1,2,3], **Yusheng Su**[1,2,3], **Zhiyuan Liu**[1,2,3,5,6]*, **Peng Li**[7]†, **Maosong Sun**[1,2,3,5]*, **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]Quan Cheng Laboratory
[7]Institute for AI Industry Research (AIR), Tsinghua University, China.
qyj20@mails.tsinghua.edu.cn

Abstract

Recent explorations of large-scale pre-trained language models (PLMs) have revealed the power of PLMs with huge amounts of parameters, setting off a wave of training ever-larger PLMs. However, it requires tremendous computational resources to train a large-scale PLM, which may be practically unaffordable. In addition, existing large-scale PLMs are mainly trained from scratch individually, ignoring that many well-trained PLMs are available. To this end, we explore the question how could existing PLMs benefit training large-scale PLMs in future. Specifically, we introduce a pre-training framework named "knowledge inheritance" (KI) and explore how could knowledge distillation serve as auxiliary supervision during pre-training to efficiently learn larger PLMs. Experimental results demonstrate the superiority of KI in training efficiency. We also conduct empirical analyses to explore the effects of teacher PLMs' pre-training settings, including model architecture, pre-training data, etc. Finally, we show that KI could be applied to domain adaptation and knowledge transfer. The implementation is publicly available at https://github.com/thunlp/Knowledge-Inheritance.

1 Introduction

Recently, it has become a consensus in the NLP community to use pre-trained language models (PLMs) as the backbone for various downstream tasks (Han et al., 2021; Min et al., 2021). 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 (Kaplan et al., 2020; Li et al., 2020b). This discovery sets off a wave of training large-scale PLMs (Raffel et al., 2019; Brown et al., 2020; Fedus et al., 2021).

Although huge PLMs have shown awesome performance (Bommasani et al., 2021), it requires tremendous computational resources to train large-scale PLMs (Schwartz et al., 2019), raising severe environmental concerns on the prohibitive computational costs. Moreover, existing PLMs are generally trained from scratch individually, ignoring that many well-trained PLMs are available. This leaves us an important question: *how could existing PLMs benefit training larger PLMs in future?*

Considering that humans can leverage the knowledge summarized by their predecessors to learn new tasks, so that the learning process could become efficient; 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 pre-training to efficiently learn larger PLMs. We dub the above process as *knowledge inheritance* (KI). This intuition is similar to reversed KD (Yuan et al., 2020) in the field of computer vision. They indicate that a delicate student model could still benefit from a teacher with an inferior architecture for a specific downstream task.

However, the success of reversed KD in supervised downstream tasks does not guarantee its feasibility under the scenario of large-scale self-supervised pre-training. Therefore, in this paper,

*Corresponding author.
† Part of the work was done while Peng Li was working at Tencent.

we strive to answer the following 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 (model architectures, training data, training strategies, etc) 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?

In conclusion, the contributions of this paper are summarized as follows: (1) we are the first to formulate the problem of knowledge inheritance, and demonstrate the feasibility of inheriting the knowledge from previously trained PLMs for efficiently training larger ones; (2) we show that the learned knowledge in PLMs could accumulate and further be passed down from generation to generation; (3) we systematically conduct empirical analyses to show the effects of various teacher pre-training settings, which may indicate how to select the most appropriate PLM as the teacher for KI; (4) we further show that during domain adaptation, an already trained large PLM could benefit from multiple small PLMs of different domains under the KI framework. The above empirical studies indicate that KI can well support cross-model knowledge transfer, providing a promising direction to share the knowledge learned by different PLMs and continuously promote their performance.

2 Related Work

Efficient Pre-training for NLP. Recently, researchers find that the performance of PLMs can be simply improved by increasing the model size, data size and training steps (Liu et al., 2019; Raffel et al., 2019; Kaplan et al., 2020), sparking a wave of training ever-larger PLMs. For instance, the revolutionary GPT-3 (Brown et al., 2020), which contains 175 billion parameters, shows strong capabilities for language understanding and generation. This means that utilizing PLMs with huge parameters for downstream tasks may greatly relieve the cost of manual labeling and model training for new tasks. However, larger models require greater computational demands (Patterson et al., 2021). To this end, researchers propose to accelerate pre-training by mixed-precision training (Shoeybi et al., 2019), distributed training (Shoeybi et al., 2019), large batch optimization (You et al., 2020), etc.

Another line of methods (Gong et al., 2019; Gu et al., 2021; Chen et al., 2022; Qin et al., 2022) proposes to pre-train larger PLMs progressively. They first train a small PLM, and then gradually increase the depth or width of the network based on parameter recycling (PR). Although PR could be used for the goal of KI, these methods typically have strict requirements on the architectures of both models, which is not flexible for practical uses; instead, we resort to KD as the solution for KI without architecture constraints. In addition, different from KI, PR is not applicable for absorbing knowledge from multiple teacher models and domain adaptation. More detailed comparisons between KI and PR are discussed in appendix E.

Knowledge Distillation for PLMs. Knowledge Distillation (KD) (Hinton et al., 2015) aims to compress a large model into a fast-to-execute one. KD has renewed a surge of interest in PLMs recently. Some explore KD at different training phases, e.g., pre-training (Sanh et al., 2019), downstream fine-tuning (Sun et al., 2019; Krishna et al., 2020), or both of them (Jiao et al., 2020); others explore distilling not only the final logits output by the large PLM, but also the intermediate hidden representations (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020). Conventional KD presumes that teacher models play pivotal roles in mastering knowledge, and student models generally cannot match their teachers in performance. When it comes to the scenario of KI, since student models have larger capacities, the performance of teacher models is no longer an "upper bound" of student models. Outside NLP, researchers recently demonstrate that a student model could also benefit from a poor teacher for a specific downstream task (Yuan et al., 2020) (reversed KD). Based on the prior explorations, in this paper, we investigate the application of reversed KD in pre-training.

3 Knowledge Inheritance

Task Formulation. Given a textual input $\mathbf{x} = \{x^1, \ldots, x^n\}$ and the corresponding label $\mathbf{y} \in \mathbb{R}^K$, where K is the number of classes for the specific pre-training task, e.g., the vocabulary size for masked language modeling (MLM) (Devlin et al., 2019), a PLM $\mathcal{M}$ converts each token

$x^j \in \mathbf{x}$ to task-specific logits $\mathbf{z}^j = [z_1^j, ..., z_K^j]$. $\mathbf{z}^j$ is then converted to a probability distribution $\mathcal{P}(x^j;\tau) = [p_1(x^j;\tau), ..., p_K(x^j;\tau)]$ using a softmax function with temperature τ. $\mathcal{M}$ is pre-trained with the objective $\mathcal{L}_{\text{SELF}}(\mathbf{x},\mathbf{y}) = \mathcal{H}(\mathbf{y}, \mathcal{P}(\mathbf{x};\tau))$, where $\mathcal{H}$ is the loss function, e.g., cross-entropy for MLM. Assume that we have a well-trained small PLM $\mathcal{M}_S$ optimized with the self learning objective $\mathcal{L}_{\text{SELF}}$ (such as MLM), our goal is leveraging $\mathcal{M}_S$'s knowledge to efficiently train a larger PLM $\mathcal{M}_L$ on the corpora $\mathcal{D}_L = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|\mathcal{D}_L|}$.

Investigated Methodology. Specifically, imparting $\mathcal{M}_S$'s knowledge to $\mathcal{M}_L$ on $\mathcal{D}_L$ is implemented by minimizing the Kullback-Leibler (KL) divergence between two probability distributions output by $\mathcal{M}_S$ and $\mathcal{M}_L$ on the same input $\mathbf{x}_i \in \mathcal{D}_L$, i.e., $\mathcal{L}_{\text{KI}}(\mathbf{x}_i;\mathcal{M}_S) = \tau^2 \text{KL}(\mathcal{P}_{\mathcal{M}_S}(\mathbf{x}_i;\tau)||\mathcal{P}_{\mathcal{M}_L}(\mathbf{x}_i;\tau)))$. In addition, $\mathcal{M}_L$ is also encouraged to conduct self-learning by optimizing $\mathcal{L}_{\text{SELF}}(\mathbf{x}_i, \mathbf{y}_i)$. Both $\mathcal{L}_{\text{SELF}}$ and $\mathcal{L}_{\text{KI}}$ are balanced with an inheritance rate α:

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

Since larger models generally converge faster and can achieve better final performance (Li et al., 2020b), $\mathcal{M}_L$ becomes more and more knowledgeable during the learning process, and would surpass the teacher eventually. Thus, it is necessary to encourage $\mathcal{M}_L$ increasingly learning knowledge on its own, not only following the teacher's instructions. Additionally, after $\mathcal{M}_L$ has surpassed its teacher, it no longer needs the guidance from $\mathcal{M}_S$ and should conduct pure self-learning from then on. Therefore, different from reversed KD, we dynamically change the inheritance rate α. Specifically, for a total training steps of T, we linearly decay α_t with a slope of $\frac{\alpha_T}{T}$. The student only inherits knowledge from the teacher for $\frac{T}{\alpha_T}$ steps, and then conducts pure self-learning, i.e., $\alpha_t = \max(1 - \alpha_T \times \frac{t}{T}, 0)$. Formally, at step t, the loss function for inheriting knowledge of $\mathcal{M}_S$ on $\mathcal{D}_L$ is formulated as:

$$\mathcal{L}(\mathcal{D}_L;\mathcal{M}_S) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}_L} (1-\alpha_t)\mathcal{L}_{\text{SELF}}(\mathbf{x}_i,\mathbf{y}_i) + \alpha_t\mathcal{L}_{\text{KI}}(\mathbf{x}_i;\mathcal{M}_S). \tag{2}$$

Note the logits of $\mathcal{M}_S$ on $\mathcal{D}_L$ can be pre-computed and saved offline so that we do not need to re-compute the logits of $\mathcal{M}_S$ when training $\mathcal{M}_L$. This process is done once and for all.

4 Empirical Analysis

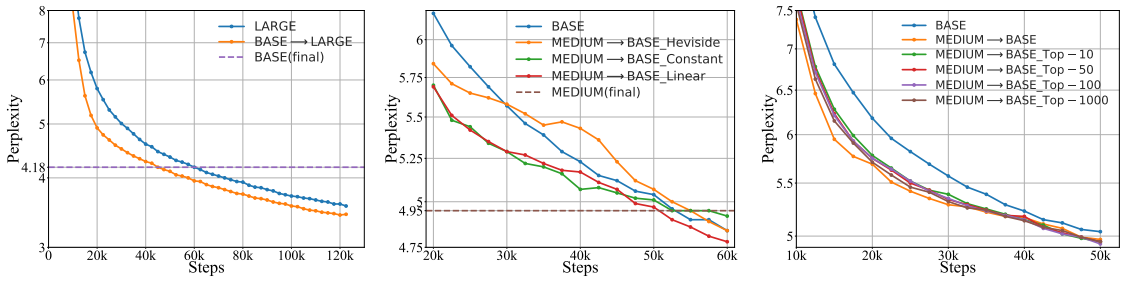
In this section, we answer our research questions proposed before. Specifically, (1) we first demonstrate the effectiveness of KI in § 4.1. (2) Then we show PLMs can accumulate knowledge over generations in § 4.2. (3) We also investigate the effects of different pre-training settings of the teacher models in § 4.3. (4) Finally, we show that KI could benefit domain adaptation, and a trained PLM can learn more efficiently with the help of multiple domain teachers in § 4.4. Detailed pre-training hyper-parameters are listed in appendix B.

4.1 RQ1: How Could Knowledge Inheritance Benefit Large PLMs' Training?

Setting. Our KI framework is agnostic to the specific self-supervised pre-training task and the PLM architecture. Without loss of generality, we mainly focus on the representative MLM task and use the model architecture of RoBERTa (Liu et al., 2019). Specifically, we first choose RoBERTa$_{\texttt{BASE}}$ (denoted as BASE) as the teacher ($\mathcal{M}_S$) and RoBERTa$_{\texttt{LARGE}}$ (denoted as LARGE) as the student ($\mathcal{M}_L$). We also experiment on auto-regressive language modeling using GPT (Radford et al., 2018) to show KI is model-agnostic.

For pre-training data, we use the concatenation of Wikipedia and BookCorpus (Zhu et al., 2015) same as BERT (Devlin et al., 2019), with roughly $3,400$M tokens in total. All models ($\mathcal{M}_S$ and $\mathcal{M}_L$) are trained for 125k steps, with a batch size of $2,048$ and a sequence length of 512. Note the whole training computations are comparable with those of BERT. We pre-train $\mathcal{M}_L$ by inheriting $\mathcal{M}_S$'s knowledge under KI (denoted as "BASE $\rightarrow$ LARGE"). We compare it with "LARGE" that only conducts self-learning from beginning to end.

For performance evaluation, we report the validation perplexity (PPL) during pre-training and the downstream performance on development sets of eight GLUE (Wang et al., 2019) tasks. Note compared with the self-learning baseline, in KI, the logits output by $\mathcal{M}_L$ are additionally used to calculate $\mathcal{L}_{\text{KI}}$, we empirically find that the additional computations caused by it are almost negligible compared with the cumbersome computations in Transformer blocks. Therefore, it requires almost the same computational cost between KI and the baseline for



(a) (b) (c)

Figure 1: (a) The validation PPL curve for pre-training $\mathcal{M}_L$ under KI framework (BASE $\rightarrow$ LARGE) and the self-learning baseline (LARGE). The teacher's (BASE) performance is 4.18. (b) Pre-training BASE under KI with three strategies for the inheritance rate α_t: Linear, Heviside and Constant. The teacher's (MEDIUM) performance is 4.95. (c) Pre-training BASE under KI with top-K logits, we vary K in $\{10, 50, 100, 1000\}$, respectively.

each step. Hence, we report the performance w.r.t training step (Li et al., 2020a), while the performance w.r.t. FLOPs (Schwartz et al., 2019) and wall-clock time (Li et al., 2020b) can be roughly obtained by stretching the figure horizontally.

Overall Results. As shown in Figure 1 (a), we conclude that: (1) **training $\mathcal{M}_L$ under KI converges faster than the self-learning baseline**, indicating that inheriting the knowledge from an existing teacher is far more efficient than solely learning such knowledge. That is, to achieve the same level of validation PPL, KI requires fewer computational costs. Specifically, under the guidance of $\mathcal{M}_S$, whose validation PPL is 4.18, BASE $\rightarrow$ LARGE achieves a validation PPL of 3.41 at the end of pre-training, compared with baseline (LARGE) 3.58. After BASE $\rightarrow$ LARGE stops learning from the teacher at the 40k-th step, it improves the validation PPL from 4.60 (LARGE) to 4.28, which is almost the performance when the baseline LARGE conducts self-learning for 55k steps, thus saving roughly 27.3% computational costs[1]. The results in Table 1 show that (2) **$\mathcal{M}_L$ trained under KI achieves better performance than the baseline on downstream tasks at each step**. We also found empirically that, under the same setting (e.g., data, hyper-parameters and model architectures), lower validation PPL generally indicates better downstream performance. Since the performance gain in downstream tasks is consistent with that reflected in PPL, we only show the latter for the remaining experiments. Concerning the energy cost, for the remaining experiments, unless otherwise specified, we choose MEDIUM (9 layers, 576 hidden size) as $\mathcal{M}_S$ and BASE as $\mathcal{M}_L$.

[1]If we load BASE and compute its logits during pre-training, 18.7% FLOPs can be saved roughly, since the forward passes of the small teacher also take up a small part.

Effects of Inheritance Rate. We set α_t in Eq. (2) to be linearly decayed (denoted as Linear) to gradually encourage $\mathcal{M}_L$ exploring knowledge on its own. We analyze whether this design is necessary by comparing it with two other strategies: the first is to only learn from the teacher at first and change to pure self-learning (denoted as Heviside) at the 35k-th step; the second is to use a constant ratio (1 : 1) between $\mathcal{L}_{\text{SELF}}$ and $\mathcal{L}_{\text{KI}}$ throughout the whole training process (denoted as Constant). We can conclude from Figure 1 (b) that: (1) **annealing at first is necessary**. The validation PPL curve of Linear converges the fastest, while Heviside tends to increase after $\mathcal{M}_L$ stops learning from the teacher, indicating that, due to the difference between learning from the teacher and self-learning, annealing at first is necessary so that the performance won't decay at the transition point (the 35k-th step). (2) **Supervision from the teacher is redundant after $\mathcal{M}_L$ surpasses $\mathcal{M}_S$**. Although Constant performs well in the beginning, its PPL gradually becomes even worse than the other two strategies. This indicates that after $\mathcal{M}_L$ has already surpassed $\mathcal{M}_S$, it will be encumbered by keeping following guidance from $\mathcal{M}_S$.

Saving Storage Space with Top-K Logits. Loading the teacher $\mathcal{M}_S$ repeatedly for KI is cumbersome, and an alternative way is to pre-compute and save the predictions of $\mathcal{M}_S$ offline once and for all. We show that using the information of top-K logits (Tan et al., 2019) can reduce the memory footprint without much performance decrease. Specifically, we save only top-K probabilities of $\mathcal{P}_S(x^j; \tau)$ followed by re-normalization, instead of the full distribution over all tokens. For RoBERTa,

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

Table 1: Downstream performances on GLUE tasks (dev). KI requires fewer pre-training steps to get a high score after fine-tuning. Detailed results at different training steps are illustrated in appendix A.6.

the dimension of $\mathcal{P}_S(x^j;\tau)$ is decided by its vocabulary size, which is around $50,000$. We thus vary K in $\{10, 50, 100, 1000\}$ to see its effects in Figure 1 (c), from which we observe that: **top-K logits contain the vast majority of information**. Choosing a relatively small K (e.g., 10) is already good enough for inheriting knowledge from the teacher without much performance decrease. Previous work also indicates the relation between KD and label smoothing (Shen et al., 2021), however, we show in appendix F that the improvements of KI are not because of benefiting from optimizing smoothed targets, which impose regularization.

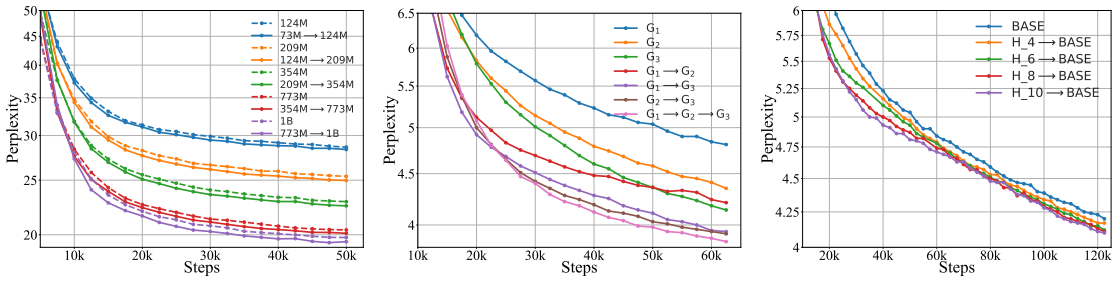
Experiments on GPT. To demonstrate that KI is model-agnostic, we conduct experiments on auto-regressive language modeling and choose GPT (Radford et al., 2018) architecture with growing sizes of $\{73\text{M}, 124\text{M}, 209\text{M}, 354\text{M}, 773\text{M}, 1\text{B}\}$ parameters in total, respectively. The detailed architectures are specified in Table 6. All the teacher models are pre-trained for 62.5k steps with a batch size of $2,048$. As reflected in Figure 2 (a), training larger GPTs under our KI framework converges faster than the self-learning baseline, which demonstrates **KI is agnostic to the specific pre-training objective and PLM architecture**.

4.2 RQ2: Could Knowledge Inheritance be Performed over Generations?

Human beings can inherit the knowledge from their antecedents, refine it and pass it down to their offsprings, so that knowledge can gradually accumulate over generations. Inspired by this, we investigate whether PLMs also have this kind of pattern. Specifically, we experiment with the knowledge inheritance among three generations of RoBERTa with roughly 1.7x growth in model size: G_1 (BASE, 125M), G_2 (BASE_PLUS, 211M) and G_3 (LARGE, 355M), whose architectures are listed in Table 6. All models are trained from scratch for 125k steps with a batch size of $2,048$ on the same corpus. We compare the differences among (1) self-learning for each generation (denoted as G_1, G_2 and G_3), (2) KI over two generations (denoted as $G_1 \rightarrow G_2$, $G_1 \rightarrow G_3$ and $G_2 \rightarrow G_3$), and (3) KI over three generations (denoted as $G_1 \rightarrow G_2 \rightarrow G_3$), where G_2 first inherits the knowledge from G_1, refines it by additional self-exploring and passes its knowledge down to G_3. The results are drawn in Figure 2 (b). Comparing the performance of G_2 and $G_1 \rightarrow G_2$, G_3 and $G_1 \rightarrow G_3$, or G_3 and $G_2 \rightarrow G_3$, we can again demonstrate the superiority of KI over self-training as concluded before. Comparing the performance of $G_1 \rightarrow G_3$ and $G_1 \rightarrow G_2 \rightarrow G_3$, or $G_2 \rightarrow G_3$ and $G_1 \rightarrow G_2 \rightarrow G_3$, it is observed that the performance of G_3 benefits from the involvements of both G_1 and G_2, which means knowledge could be accumulated through more generations' involvements.

4.3 RQ3: How Could $\mathcal{M}_S$'s Pre-training Setting Affect Knowledge Inheritance?

Existing PLMs are typically trained under quite different settings, and it is unclear how these different settings will affect the performance of KI. Formally, we have a series of well-trained smaller PLMs $\overline{\mathcal{M}_S} = \{\mathcal{M}_S^1, ..., \mathcal{M}_S^{N_S}\}$, each having been optimized on $\overline{\mathcal{D}_S} = \{\mathcal{D}_S^1, ..., \mathcal{D}_S^{N_S}\}$, respectively. Considering that the PLMs in $\overline{\mathcal{M}_S}$, consisting of varied model architectures, are pre-trained on different corpora of various sizes and domains with arbitrary strategies, thus the knowledge they master is also manifold. In addition, $\mathcal{M}_L$'s pre-training data $\overline{\mathcal{D}_L}$ may also consist of massive, heterogeneous corpora from multiple sources, i.e., $\overline{\mathcal{D}_L} = \{\mathcal{D}_L^1, ..., \mathcal{D}_L^{N_L}\}$. Due to the difference between $\overline{\mathcal{D}_L}$ and $\overline{\mathcal{D}_S}$, $\mathcal{M}_S$



(a) (b) (c)

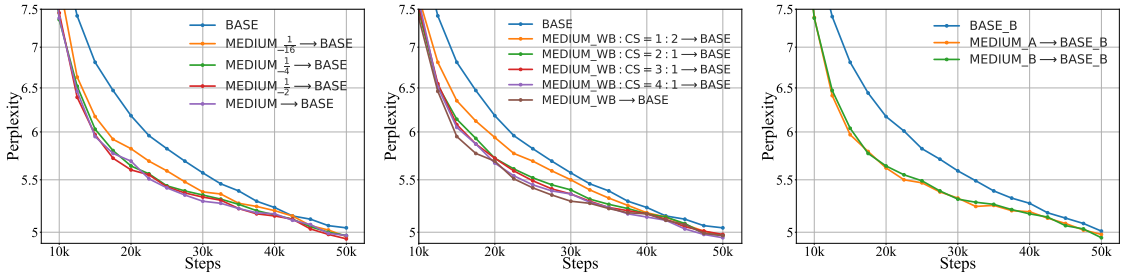
Figure 2: (a) Experiments on GPT. (b) KI over generations. (c) Effects of $\mathcal{M}_S$'s architecture (depth).

may be required to transfer its knowledge on instances unseen during its pre-training. Ideally, we want $\mathcal{M}_S$ to teach the courses it is skilled in. Therefore, it is essential to choose the most appropriate teacher for each composition $\mathcal{D}_L^* \in \overline{\mathcal{D}_L}$. To this end, we conduct thorough experiments to analyze the effects of several representative factors: model architecture, pre-training data, $\mathcal{M}_S$'s pre-training step (appendix A.2) and batch size (appendix A.3).

Effects of Model Architecture. Large PLMs generally converge faster and achieve lower PPL, thus serving as more competent teachers. We experiment with two widely chosen architecture variations, i.e., depth (number of layers) and width (hidden size), to explore the effects of $\mathcal{M}_S$'s model architectures. We choose `BASE` (12 layer, 768 hidden size) as $\mathcal{M}_L$'s architecture, and choose the architecture of $\mathcal{M}_S$ to differ from $\mathcal{M}_L$ in either depth or width. Specifically, for $\mathcal{M}_S$, we vary the depth in $\{4, 6, 8, 10\}$, and the width in $\{384, 480, 576, 672\}$, respectively, and pre-train $\mathcal{M}_S$ under the same setting as $\mathcal{M}_L$. The PPL curve for each teacher model is shown in appendix A.7, from which we observe that deeper / wider teachers with more parameters converge faster and achieve lower PPL. After that, we pre-train $\mathcal{M}_L$ under KI leveraging these teacher models. As shown in Figure 2 (c) and appendix A.4, **choosing a deeper / wider teacher accelerates $\mathcal{M}_L$'s convergence**, demonstrating the benefits of learning from a more knowledgeable teacher. Since the performance of PLMs is weakly related to the model shape but highly related to the model size (Li et al., 2020b), it is always a better strategy to choose the larger teacher if other settings are kept the same. In experiments, we also find empirically that, the optimal duration of learning from the teacher is longer for larger teachers, which means it takes more time to learn from a more knowledgeable teacher.

Effects of Pre-training Data. In previous experiments, we assume $\mathcal{M}_L$ is pre-trained on the same corpus as $\mathcal{M}_S$, i.e., $\mathcal{D}_L = \mathcal{D}_S$. However, in real-world scenarios, it may occur that the pre-training corpus used by both $\mathcal{M}_L$ and $\mathcal{M}_S$ is mismatched, due to three main factors: (1) **data size.** When training larger models, the pre-training corpus is often enlarged to improve downstream performance, i.e., $|\mathcal{D}_S| \ll |\mathcal{D}_L|$; (2) **data domain.** PLMs are trained on heterogeneous corpora from various sources (e.g., news articles, literary works, etc.), i.e., $\mathcal{P}_{\mathcal{D}_S} \neq \mathcal{P}_{\mathcal{D}_L}$. The different knowledge contained in each domain may affect PLMs' generalization in downstream tasks; (3) **data privacy.** Even if both size and domain of $\mathcal{D}_S$ and $\mathcal{D}_L$ are ensured to be the same, it may be hard to retrieve the pre-training corpus used by $\mathcal{M}_S$ due to privacy concerns, with an extreme case: $\mathcal{D}_L \cap \mathcal{D}_S = \emptyset$. The gap between $\mathcal{D}_S$ and $\mathcal{D}_L$ may hinder $\mathcal{M}_S$'s successful knowledge transfer. We thus design experiments to analyze the effects of these factors, with three observations concluded:

- **Obs. 1: PLMs can image the big from the small for in-domain data**. To evaluate the effects of data size, we first pre-train teacher models on different partitions of the original training corpus under the same setting by randomly sampling $\{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{1}{1}\}$ of it, resulting in teacher models with final validation PPL of $\{5.43, 5.15, 5.04, 4.98, 4.92\}$, respectively. The final validation PPL increases as we shrink the size of $\mathcal{M}_S$'s pre-training corpus, which implies that training with less data weakens the teacher's ability. Next, we compare the differences when their knowledge is inherited by $\mathcal{M}_L$. As reflected in Figure 3 (a), however, the performance of KI is not substantially undermined until only $\frac{1}{16}$ of the original data is leveraged by the teacher. This indicates that PLMs can well image the overall data distribution even if it only sees a small part. Hence,



(a) (b) (c)

Figure 3: Effects of $\mathcal{M}_S$'s pre-training (a) data size, (b) data domain and (c) data privacy for KI.

when training larger PLMs, unless the data size is extensively enlarged, its impact can be ignored.

• **Obs. 2: Inheriting on similar domain improves performance**. To evaluate the effects of data domain, we experiment on the cases where $\mathcal{D}_S$ and $\mathcal{D}_L$ have domain mismatch. Specifically, keeping data size the same, we mix Wikipedia and BookCorpus (WB) used before with computer science (CS) papers from S2ORC (Lo et al., 2020), whose domain is distinct from WB, using different proportions, i.e., WB : CS $= \{1:2, 2:1, 3:1, 4:1\}$, respectively. We pre-train $\mathcal{M}_S$ on the constructed corpora, then test the performance when $\mathcal{M}_L$ inherits these teachers' knowledge on the WB domain data. As shown in Figure 3 (b), with the domain of the constructed corpus $\mathcal{M}_S$ is trained on becoming gradually similar to WB, the benefits from KI become more obvious, which means it is essential that both $\mathcal{M}_S$ and $\mathcal{M}_L$ are trained on similar domain of data, so that $\mathcal{M}_S$ can successfully impart knowledge to $\mathcal{M}_L$ by teaching the "right" course.

• **Obs. 3: Data privacy is not that important if the same domain is ensured**. To evaluate the effects of data privacy, we experiment in an extreme case where $\mathcal{D}_S$ and $\mathcal{D}_L$ have no overlap at all. To avoid the influences of size and domain, we randomly split the WB domain training corpus $\mathcal{D}$ into two halves ($\mathcal{D}_A$ and $\mathcal{D}_B$) and pre-train two teacher models (denoted as $\texttt{MEDIUM}_\texttt{A}$ and $\texttt{MEDIUM}_\texttt{B}$) on them. After pre-training, both of them achieve almost the same final PPL (4.99) on the same validation set. They are then inherited by the student model BASE on $\mathcal{D}_B$ (denoted as $\texttt{MEDIUM}_\texttt{A} \rightarrow \texttt{BASE}_\texttt{B}$ and $\texttt{MEDIUM}_\texttt{B} \rightarrow \texttt{BASE}_\texttt{B}$), which is exactly the pre-training corpus of $\texttt{MEDIUM}_\texttt{B}$ and has no overlap with that of $\texttt{MEDIUM}_\texttt{A}$. We also choose $\mathcal{M}_L$ that conducts pure self-learning on $\mathcal{D}_B$ as the baseline (denoted as $\texttt{BASE}_\texttt{B}$). It is observed from Figure 3 (c) that, there is little difference between the validation PPL curves of $\texttt{MEDIUM}_\texttt{A} \rightarrow \texttt{BASE}_\texttt{B}$ and $\texttt{MEDIUM}_\texttt{B} \rightarrow \texttt{BASE}_\texttt{B}$, indicating that whether the pre-training corpus of $\mathcal{M}_S$ and $\mathcal{M}_L$ has data overlap or not is not a serious issue as long as they share the same domain. This is meaningful when organizations aim to share the knowledge of their PLMs without exposing either the pre-training data or the model parameters due to privacy concerns.

4.4 RQ4: How Could Knowledge Inheritance Benefit Domain Adaptation?

With streaming data of various domains continuously emerging, training domain-specific PLMs and storing the model parameters for each domain can be prohibitively expensive. To this end, researchers recently demonstrated the feasibility of adapting PLMs to the target domain through continual pre-training (Gururangan et al., 2020). In this section, we further extend KI and demonstrate that domain adaptation for PLM can benefit from inheriting knowledge of existing domain experts.

Specifically, instead of training large PLMs from scratch, which is the setting used before, we focus on adapting $\texttt{BASE}_\texttt{WB}$, which has been well-trained on the WB domain for 125k steps, to two target domains, i.e., computer science (CS) and biomedical (BIO) papers from S2ORC (Lo et al., 2020). The proximity (vocabulary overlap) of three domains is listed in appendix D. We assume there exist two domain experts, i.e., $\texttt{MEDIUM}_\texttt{CS}$ and $\texttt{MEDIUM}_\texttt{BIO}$. Each model has been trained on CS / BIO domain for 125k steps. Note their training computation is far less than $\texttt{BASE}_\texttt{WB}$ due to fewer model parameters. Hence, either $\texttt{MEDIUM}_\texttt{CS}$ or $\texttt{MEDIUM}_\texttt{BIO}$ is no match for $\texttt{BASE}_\texttt{WB}$ in WB domain but has richer knowledge in CS / BIO domain. For evaluation, we compare both (1) the validation PPL on the target domain and (2) the performance (test F1) on downstream tasks, i.e. ACL-ARC (Jurgens et al., 2018) for CS domain and CHEMPROT (Kringelum et al., 2016) for BIO domain. Before adaptation,

$\mathbf{N_{tokens}}$		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**

Table 2: The validation PPL (PPL) and downstream performance (F1) on the target domain (CS / BIO) after $\texttt{BASE}_{\texttt{WB}}$ is post-trained for 4k steps with self-learning (SL) or knowledge inheritance (KI). We experiment with different sizes of domain corpus. All downstream experiments are repeated 10 times with different seeds.

$\mathbf{N_{tokens}}$	3,400M				200M				100M				40M			
Metrics	$F1_C$	PPL_C	$F1_B$	PPL_B	$F1_C$	PPL_C	$F1_B$	PPL_B	$F1_C$	PPL_C	$F1_B$	PPL_B	$F1_C$	PPL_C	$F1_B$	PPL_B
SL	71.7	3.15	83.7	2.71	70.5	3.97	82.7	3.36	67.7	5.95	81.7	4.84	68.3	11.7	81.1	10.5
KI	**72.2**	**3.15**	**83.9**	**2.70**	**71.8**	**3.42**	**83.1**	**2.92**	**69.8**	**3.90**	**82.6**	**3.32**	**69.1**	**5.70**	**81.3**	**4.64**

Table 3: The results when $\texttt{BASE}_{\texttt{WB}}$ is post-trained on two new domains simultaneously with self-learning (SL) or knowledge inheritance (KI). We report both validation PPL (PPL_B / PPL_C) and downstream performance ($F1_B$ / $F1_C$) for BIO / CS domain. We observe that SL exhibits severe overfitting when data is relatively scarce.

$\texttt{BASE}_{\texttt{WB}}$ achieves a PPL of 5.41 / 4.86 and F1 of 68.5 / 81.6 on CS / BIO domain, while $\texttt{MEDIUM}_{\texttt{CS}}$ achieves 2.95 (PPL) and 69.4 (F1) on CS domain, $\texttt{MEDIUM}_{\texttt{BIO}}$ achieves 2.55 (PPL) and 83.6 (F1) on BIO domain. This demonstrates the superiority of two teachers over the student in their own domain despite their smaller model capacity.

We compare two strategies for domain adaptation: (1) only conducting self-learning on the target domain and (2) inheriting knowledge from well-trained domain teachers. Specifically, $\texttt{BASE}_{\texttt{WB}}$ is post-trained for additional 4k steps on either CS or BIO domain to learn new knowledge. In addition, considering that in real-world scenarios, it can be hard to retrieve enough pre-training data for a specific domain, due to some privacy issues. Hence, we conduct experiments with different sizes of domain corpus. In Table 2, $\texttt{BASE}_{\texttt{WB}}$ is post-trained on either CS or BIO domain while in Table 3, it is trained on synthetic domain data (BIO : CS = 1 : 1) to absorb knowledge from two domains simultaneously (we assume $\mathcal{M}_L$ is trained with the optimal teacher selection strategy, i.e., each teacher imparts the knowledge on its own domain data). It can be concluded from Table 2 and Table 3 that:

(1) **KI is more training-efficient**. Compared with self-learning, inheriting knowledge from domain teachers achieves lower final PPL and improved performance in domain-specific downstream tasks, indicating that, for domain adaptation, KI is more training-efficient so that an already trained large PLM could absorb more knowledge from new domain with the same training budget. (2) **KI is more data-efficient**. The PPL gap between KI and SL is further enlarged when there is less domain-specific data available for adaptation, which means KI is more data-efficient especially under the low-resource setting, where domain data is scarce. In other words, only providing a small portion of domain-specific data is enough for satisfactory adaptation performance under KI, while self-learning exhibits overfitting to some extent. (3) **Large PLMs can simultaneously absorb knowledge from multiple domains and thus become omnipotent**. From Table 3, we observe $\texttt{BASE}_{\texttt{WB}}$ achieves improved performance on both domains after being taught by two teachers simultaneously. KI shows superiority over self-learning. However, simultaneous learning overfits training data more easily and its performance on either domain is no match for learning only one domain at a time.

5 Conclusion and Future Work

In this work, we propose a general knowledge inheritance (KI) framework that leverages previously trained PLMs for training larger ones. We conduct sufficient empirical studies to demonstrate its feasibility. In addition, we show that KI could well support knowledge transfer over a series of PLMs with growing sizes. We also comprehensively analyze various pre-training settings of the teacher model that may affect KI's performance, the results shed light on how to choose the most appropriate teacher PLM for KI. Finally, we ex-

tend KI and show that, during domain adaptation, an already trained large PLM could benefit from smaller domain teachers. In general, we provide a promising direction to share and exchange the knowledge learned by different models and continuously promote their performance.

In future, we aim to explore the following directions: (1) the **efficiency** of KI, i.e., given limited computational budget and pre-training corpus, how to more efficiently absorb knowledge from teacher models. Potential solutions include denoising teacher models' predictions and utilizing more information from the teacher. How to select the most representative data points for KI is also an interesting topic; (2) the **effectiveness** of KI under different settings, i.e., how can KI be applied if the teachers and the students are pre-trained on different vocabularies, languages, pre-training objectives and modalities.

Finally, we believe it is vital to use fair benchmarking that can accurately and reliably judge each KI algorithm. Thus, we suggest future work to: (1) conduct all experiments under the same computation environment and report the pre-training hyper-parameters and hardware deployments in detail, (2) evaluate the downstream tasks with multiple different random seeds and choose tasks that give relatively stable and consistent results, which could serve as better indicators for PLMs' effectiveness. In addition, it is also essential that PLMs are tested on diverse downstream tasks which evaluate PLMs' different abilities, (3) save the checkpoint more frequently during pre-training and evaluate the downstream performance, which can better indicate the trend of PLMs' effectiveness, and (4) open-source all the codes and model parameters for future comparisons.

Acknowledgments

This work is supported by the National Key R&D Program of China (No. 2020AAA0106502), Institute Guo Qiang at Tsinghua University, NExT++ project from the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative, and Beijing Academy of Artificial Intelligence (BAAI). This work is also supported by the Pattern Recognition Center, WeChat AI, Tencent Inc. Yujia Qin and Yankai Lin designed the methods and the experiments. Yujia Qin, Jing Yi and Jiajie Zhang conducted the experiments. Yujia Qin, Yankai Lin and Xu Han wrote the paper. Zhiyuan Liu, Peng Li, Maosong Sun and Jie Zhou advised the project. All the authors participated in the discussion.

References

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2022. bert2bert: Towards reusable pretrained language models. In *Association for Computational Linguistics: ACL 2022*, Online.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv preprint*, abs/2101.03961.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1602–1611. PMLR.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Efficient training of BERT by progressively stacking. In *Proceedings*

of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2337–2346. PMLR.

Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2021. On the transformer growth for progressive BERT training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5174–5180, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *ArXiv preprint*, abs/2106.07139.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *ArXiv preprint*, abs/1503.02531.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv preprint*, abs/2001.08361.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on sesame street! model extraction of bert-based apis. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mengtian Li, Ersin Yumer, and Deva Ramanan. 2020a. Budgeted training: Rethinking deep neural network training under resource constraints. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020b. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5958–5968. PMLR.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *ArXiv preprint*, abs/2106.04554.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv preprint arXiv:2111.01243*.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *ArXiv preprint*, abs/2104.10350.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, Online.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2019. Green ai. *ArXiv preprint*, abs/1907.10597.

Zhiqiang Shen, Zechun Liu, Dejia Xu, Zitian Chen, Kwang-Ting Cheng, and Marios Savvides. 2021. Is label smoothing truly incompatible with knowledge distillation: An empirical study. *arXiv preprint arXiv:2104.00676*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *ArXiv preprint*, abs/1909.08053.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online. Association for Computational Linguistics.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L Yuille. 2019. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5628–5635.

Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Reducing bert pre-training time from 3 days to 76 minutes. *ArXiv preprint*, abs/1904.00962.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2020. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

Appendices

A Additional Experiments and Analysis

A.1 Effects of Model Size

We experiment on four PLMs with roughly 1.7x growth in model size: $\mathcal{M}_1$ (RoBERTa$_{\text{MEDIUM}}$, 73.5M), $\mathcal{M}_2$ (RoBERTa$_{\text{BASE}}$, 125M), $\mathcal{M}_3$ (RoBERTa$_{\text{BASE_PLUS}}$, 211M) and $\mathcal{M}_4$ (RoBERTa$_{\text{LARGE}}$, 355M), whose architectures are listed in Table 6. We first pre-train a teacher PLM $\mathcal{M}_i$ ($\mathcal{M}_S$) for 125k steps with a batch size of $2,048$ under the same setting then train a larger one $\mathcal{M}_{i+1}$ ($\mathcal{M}_L$) by inheriting $\mathcal{M}_i$'s knowledge under KI framework (denoted as $\mathcal{M}_i \rightarrow \mathcal{M}_{i+1}, i \in \{1,2,3\}$). We compare $\mathcal{M}_i \rightarrow \mathcal{M}_{i+1}$ with $\mathcal{M}_{i+1}$ that conducts self-learning from beginning to end. As shown in Figure 4, the superiority of KI is observed across all models. In addition, with the overall model size of $\mathcal{M}_S$ and $\mathcal{M}_L$ gradually increasing, the benefits of KI become more evident, reflected in the broader absolute gap between the PPL curve of $\mathcal{M}_i \rightarrow \mathcal{M}_{i+1}$ and $\mathcal{M}_{i+1}$ when i gradually grows. This implies that with the advance of computing power in future, training larger PLMs will benefit more and more from our KI framework.

A.2 Effects of $\mathcal{M}_S$'s Pre-training Steps

Longer pre-training has been demonstrated as an effective way for PLMs to achieve better performance (Liu et al., 2019) and thus become more knowledgeable. To evaluate the benefits of more pre-training steps for $\mathcal{M}_S$, we first vary RoBERTa$_{\text{MEDIUM}}$'s pre-training steps in $\{62.5\text{k}, 125\text{k}, 250\text{k}, 500\text{k}\}$, and keep all other settings the same. After pre-training, these teacher models achieve the final validation PPL of $\{5.25, 4.92, 4.72, 4.51\}$, respectively. Then we compare the performances when RoBERTa$_{\text{BASE}}$ learn from these teacher models and visualize the results in Figure 4, from which we can conclude that, inheriting knowledge from teachers with longer pre-training time (steps) helps $\mathcal{M}_L$ converge faster. However, such a benefit is less and less obvious as $\mathcal{M}_S$'s pre-training steps increase, which means after enough training computations are invested, the teacher model enters a plateau of convergence in validation PPL, and digging deeper in knowledge becomes even harder. The bottleneck lies in other factors, e.g., the size and diversity of pre-training data, which hinder $\mathcal{M}_S$ from becoming more knowledgeable. We also found empirically that, after being pre-trained for 125k steps on the corpus with a batch size of $2,048$, all the models used in this paper have well converged, and longer pre-training only results in limited performance gain in either PPL or downstream performance.

A.3 Effects of $\mathcal{M}_L$'s Batch Size

Batch size is highly related to PLM's training efficiency, and previous work (Liu et al., 2019; Li et al., 2020b; You et al., 2019) found that slow-but-accurate large batch sizes can bring improvements to model training, although the improvements become marginal after increasing the batch size beyond a certain point (around $2,048$). BERT (Devlin et al., 2019) is pre-trained for $1,000$k steps with a batch size of 256, and the computational cost is equivalent to training for 125k steps with a batch size of $2,048$ (Liu et al., 2019), which is the pre-training setting chosen in our main paper. Choosing RoBERTa$_{\text{MEDIUM}}$ as the teacher model and RoBERTa$_{\text{BASE}}$ as the student model, in Figure 4 we compare the validation PPL as we vary the batch size in $\{256, 512, 1024, 2,048\}$, controlling for the number of passes through the pre-training corpus. We also vary the peak learning rate in $\{1.0 \times 10^{-4}, 2.5 \times 10^{-4}, 3.8 \times 10^{-4}, 5.0 \times 10^{-4}\}$ and pre-train for $\{1,000\text{k}, 500\text{k}, 250\text{k}, 125\text{k}\}$ steps, respectively, when increasing the batch size. We observe that increasing the batch size results in improved final validation PPL, which is aligned with previous findings (Liu et al., 2019). When adjusting batch size, KI accelerates the convergence unanimously, and its benefits become more evident when training with a smaller batch size, reflected in the absolute improvement in final validation PPL. We hypothesize that this is because learning from the smoothed target probability of KI, containing rich *secondary information* (Yang et al., 2019) or *dark knowledge* (Furlanello et al., 2018), makes the pre-training process more stable. The student PLM is prevented from fitting to unnecessarily strict distributions and can thus learn faster.

A.4 Additional Experiments of the Effects of Teacher Model $\mathcal{M}_S$'s architecture (width)

We show in Figure 5 the validation PPL of $\mathcal{M}_L$ when choosing the teacher PLM $\mathcal{M}_S$ with different hidden sizes ($\{384, 480, 576, 672\}$). As mentioned in our main paper, choosing a wider teacher

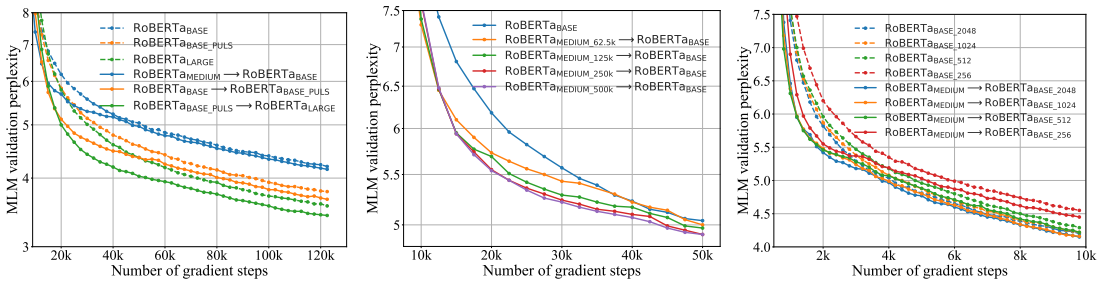


Figure 4: Left: effects of $\mathcal{M}_L$'s model size. Middle: effects of $\mathcal{M}_S$'s number of pre-training steps. Right: effects of $\mathcal{M}_L$'s batch size.

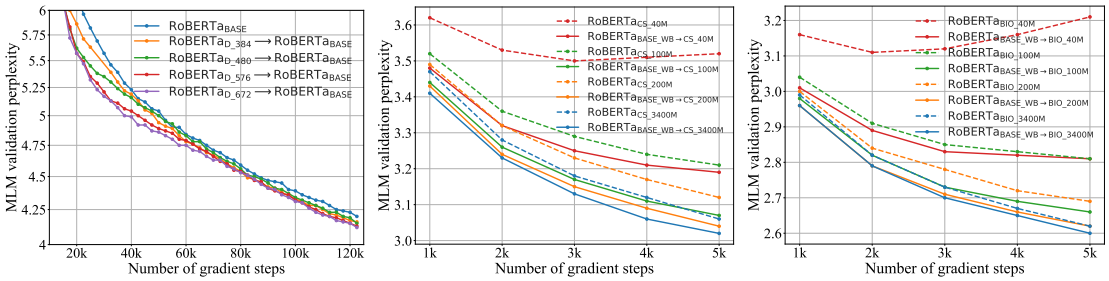


Figure 5: Left: the PPL curve when choosing the teacher PLM with different hidden sizes. Middle & Right: adapting $\text{RoBERTa}_{\texttt{BASE_WB}}$ to CS (middle) / BIO (right) domain with different number of training steps on different sizes of domain data. We compare two strategies: self-learning and KI. For example, $\text{RoBERTa}_{\texttt{CS_3400M}}$ denotes post-training $\text{RoBERTa}_{\texttt{BASE_WB}}$ with the self-learning strategy on the $3,400$M token CS domain corpus. $\text{RoBERTa}_{\texttt{BASE_WB}\rightarrow\texttt{CS_3400M}}$ denotes post-training $\text{RoBERTa}_{\texttt{BASE_WB}}$ with the KI strategy on the $3,400$M token CS domain corpus.

model improves the training efficiency of the student PLM.

A.5 Additional Experiments of Knowledge Inheritance for Domain Adaptation

Domain	Strategy	$3,400$M	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

Table 4: The validation PPL on the source domain (WB) after $\text{RoBERTa}_{\texttt{BASE_WB}}$ is post-trained on the target domain (CS / BIO) with self-learning (SL) and knowledge inheritance (KI).

Different Number of Post-training Steps. In the main paper, we adapt $\text{RoBERTa}_{\texttt{BASE_WB}}$ to either CS or BIO domain by post-training it for 4k steps. We further vary the number of training steps in $\{1\text{k}, 2\text{k}, 3\text{k}, 4\text{k}, 5\text{k}\}$ and visualize the validation PPL in Figure 5. We also experiment on different sizes of domain corpus, i.e., $3,400$M, 200M, 100M, 40M tokens, respectively, as done in the main paper. We observe that generally the validation PPL on each domain decreases with the training step growing, and the performance of KI is always better than self-learning. The improvement of KI over self-learning is further enlarged when there is less target domain data available, demonstrating that KI is more data-efficient and can work well in low-resource settings. In addition, self-learning exhibits overfitting problems when the data size of the target domain is relatively small, which is not observed under our KI framework, which means KI can mitigate overfitting under low-resource settings.

Catastrophic Forgetting on the Source Domain. Table 4 lists the validation PPL on the source domain (WB) after $\text{RoBERTa}_{\texttt{BASE_WB}}$ is post-trained on the target domain (CS / BIO) with self-learning (SL) and knowledge inheritance (KI) for 4k steps. We show the results w.r.t. different sizes of domain corpus ($3,400$M, 200M, 100M and 40M tokens). We observe that after domain adaptation, the validation PPL on the source domain increases, which means PLMs may forget some key knowledge on the source domain when learning new knowledge

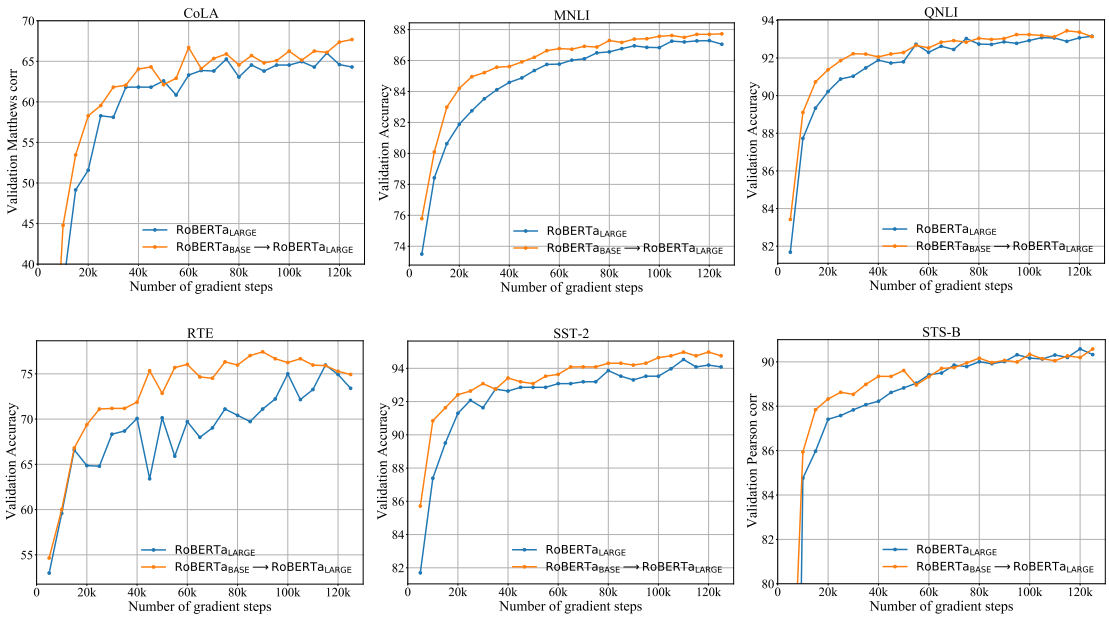


Figure 6: Downstream performance visualization on six GLUE tasks comparing RoBERTa$_{\text{LARGE}}$ and RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$. For CoLA, RTE, SST-2 and STS-B, we repeat fine-tuning for 5 times; for MNLI and QNLI, we repeat fine-tuning for 3 times.

in the target domain, i.e., the catastrophic forgetting problem. In addition, we find that the problem is more evident for KI than self-learning. We expect future work to further explore how to mitigate the catastrophic forgetting.

A.6 Detailed Downstream Performances on GLUE Tasks

Figure 6 visualizes the downstream performance of RoBERTa$_{\text{LARGE}}$ and RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$ on the dev sets of six GLUE tasks at different pre-training steps with an interval of 5k. It can be observed that the downstream performance of RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$ rises faster than the baseline, which means it takes fewer pre-training steps for our KI framework to get a high score in downstream tasks. Aligned with previous findings (Li et al., 2020b), we found MNLI and SST-2 to be the most stable tasks in GLUE, whose variances are lower.

We also list the average GLUE performance for RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$ and the baseline RoBERTa$_{\text{LARGE}}$ in Table 5, from which we observe that the baseline at 70k-th step achieves almost the same GLUE performance as our method at 40k-th step, which means our framework saves around 42.9% FLOPs, much higher than the reported 27.3% FLOPs saved based on the pre-training PPL metric in the main paper. In addition, our method achieves almost the same GLUE performance as the baseline at the final step (125k) with only 70k steps, which means our framework saves 44% FLOPs in total. Both the perplexity in the pre-training stage and performance in downstream tasks can be chosen as the evaluation metric for measuring the computational cost savings. However, in this paper, we choose the former because it is more stable and accurate than the latter. We find empirically that some GLUE tasks like CoLA have higher variances than others, which might make the measurement inaccurate.

Besides, when discussing the effects of model architectures in the main paper, we only show the validation PPL of each model during pre-training, we visualize the corresponding downstream performance (MNLI) in Figure 7, from which it can be observed that learning from teacher models with more parameters helps achieve better downstream performance at the same pre-training step. In general, we observe that, under our setting, the performance gain in downstream tasks is aligned with that reflected in validation PPL during pre-training.

A.7 Teacher Models' Validation PPL Curves during Pre-training for "Effects of Model Architecture"

Figure 7 visualizes the validation PPL curves for all the teacher models used in the experiments

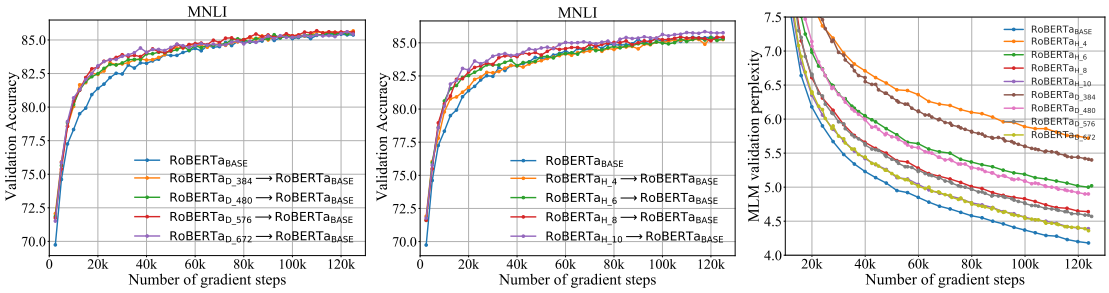


Figure 7: Left & Middle: downstream performances corresponding to the experiments on effects of $\mathcal{M}_S$'s model architecture (width (left) & depth (middle)). Right: validation PPL during pre-training for the teacher models used in experiments of effects of teacher model architecture.

Step	RoBERTa$_{\text{BASE}}$	RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$
5k	61.8	68.8
10k	75.6	78.1
15k	79.3	81.5
20k	80.4	82.8
25k	81.7	83.6
30k	82.4	83.9
35k	83.1	84.1
40k	83.6	84.5
45k	82.8	85.2
50k	83.9	84.6
55k	83.4	85.2
60k	84.0	85.7
65k	84.1	85.3
70k	84.3	85.5
75k	85.0	85.8
80k	84.7	85.8
85k	84.8	86.2
...	...	...
125k	85.5	86.1

Table 5: Average GLUE performance comparing both RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{LARGE}}$ at different pre-training steps.

on the effects of model architecture. The teacher models differ from RoBERTa$_{\text{BASE}}$ in either the depth or width. Specifically, we vary the depth in $\{4, 6, 8, 10\}$ (denoted as $\{$RoBERTa$_{\text{H_4}}$, RoBERTa$_{\text{H_6}}$, RoBERTa$_{\text{H_8}}$, RoBERTa$_{\text{H_10}}\}$), and the width in $\{384, 480, 576, 672\}$ (denoted as $\{$RoBERTa$_{\text{D_384}}$, RoBERTa$_{\text{D_480}}$, RoBERTa$_{\text{D_576}}$, RoBERTa$_{\text{D_672}}\}$). Generally, PLMs with larger model parameters converge faster and achieve better final performance.

B Pre-training Hyper-parameters

In Table 6, we list the architectures we used for all models, covering the details for the total number of trainable parameters (n_{params}), the total number of layers (n_{layers}), the number of units in each bottleneck layer (d_{model}), the total number of attention heads (n_{heads}), the inner hidden size of FFN layer (d_{FFN}) and the learning rate when batch size is set to $2,048$ (lr). The training-validation ratio of pre-training data is set to $199:1$. We set the weight decay to 0.01, dropout rate to 0.1, and use linear learning rate decay. Adam is chosen as the optimizer. The learning rate is warmed up for the first 10% steps. The hyper-parameters for Adam optimizer is set to $1 \times 10^{-6}, 0.9, 0.98$ for $\epsilon, \beta_1, \beta_2$, respectively. For a fair comparison, all experiments are done in the same computation environment with 8 NVIDIA 32GB V100 GPUs. Table 7 describes the total number of pre-training steps for each ($\mathcal{M}_L$, $\mathcal{M}_S$) pair chosen in our experiments.

C Fine-tuning Hyper-parameters

Table 8 describes the hyper-parameters for ACL-ARC, CHEMPROT and GLUE tasks. The selection of these hyper-parameters closely follows (Liu et al., 2019) and (Gururangan et al., 2020).

D Domain Proximity of WB, CS and BIO

Table 9 lists the domain proximity (vocabulary overlap) of WB, CS and BIO used in this paper.

E Comparison between Knowledge Inheritance and Parameter Recycling

Parameter recycling (i.e., progressive training) first trains a small PLM, and then gradually increases the depth or width of the network based on parameter initialization. It is an orthogonal research direction against our KI, and has many limitations as follows:

Architecture Mismatch. Existing parameter recycling methods (Gong et al., 2019; Gu et al., 2021) require that the architectures of both small PLMs

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{FFN}	lr (bs $= 2,048$)
RoBERTa$_{\text{MEDIUM}}$	74M	9	576	12	3072	5.0×10^{-4}
RoBERTa$_{\text{D_d}}$	-	12	d	12	3072	5.0×10^{-4}
RoBERTa$_{\text{H_h}}$	-	h	768	12	3072	5.0×10^{-4}
RoBERTa$_{\text{BASE}}$	125M	12	768	12	3072	5.0×10^{-4}
RoBERTa$_{\text{BASE_PLUS}}$	211M	18	864	12	3600	3.5×10^{-4}
RoBERTa$_{\text{LARGE}}$	355M	24	1024	16	4096	2.5×10^{-4}
GPT$_{\text{73M}}$	73M	9	576	12	3072	5.0×10^{-4}
GPT$_{\text{124M}}$	124M	12	768	12	3072	5.0×10^{-4}
GPT$_{\text{209M}}$	209M	18	864	12	3600	4.0×10^{-4}
GPT$_{\text{354M}}$	354M	24	1024	16	4096	3.5×10^{-4}
GPT$_{\text{773M}}$	773M	36	1280	20	5120	3.0×10^{-4}
GPT$_{\text{1B}}$	1068M	40	1440	20	5760	2.5×10^{-4}

Table 6: Model architectures for all the models we used in this paper.

$\mathcal{M}_L$	$\mathcal{M}_S$	Steps of teacher-guided learning
RoBERTa$_{\text{BASE}}$	RoBERTa$_{\text{MEDIUM}}$	35k
	RoBERTa$_{\text{D_384}}$	28k
	RoBERTa$_{\text{D_480}}$	40k
	RoBERTa$_{\text{D_576}}$	70k
	RoBERTa$_{\text{D_672}}$	85k
	RoBERTa$_{\text{H_4}}$	22k
	RoBERTa$_{\text{H_6}}$	35k
	RoBERTa$_{\text{H_8}}$	55k
	RoBERTa$_{\text{H_10}}$	65k
RoBERTa$_{\text{BASE_PLUS}}$	RoBERTa$_{\text{BASE}}$	55k
RoBERTa$_{\text{LARGE}}$	RoBERTa$_{\text{BASE}}$	40k
	RoBERTa$_{\text{BASE_PLUS}}$	65k
	RoBERTa$_{\text{BASE}}$ $\rightarrow$ RoBERTa$_{\text{BASE_PLUS}}$	75k
GPT$_{\text{124M}}$	GPT$_{\text{73M}}$	10k
GPT$_{\text{209M}}$	GPT$_{\text{124M}}$	15k
GPT$_{\text{354M}}$	GPT$_{\text{209M}}$	18k
GPT$_{\text{773M}}$	GPT$_{\text{354M}}$	16k
GPT$_{\text{1B}}$	GPT$_{\text{773M}}$	20k

Table 7: The total number of steps for teacher-guided learning for different ($\mathcal{M}_L$, $\mathcal{M}_S$) pairs.

and large PLMs are matched to some extent, however, our KI does not have such a requirement. For example, Gong et al. (2019); Gu et al. (2021) either requires the number of layers, or the hidden size/embedding size of a large PLM to be the integer multiples of that of a small PLM. Hence, it is not flexible to train larger PLMs with arbitrary architectures, making parameter recycling hard to be implemented practically. Besides, there are more and more advanced non-trivial Transformer modifications appearing (we refer to Lin et al. (2021) for details), e.g., pre-normalization, relative embedding, sparse attention, etc. It is non-trivial to directly transfer the parameters between two PLMs if they have different inner structures. Nevertheless, our KI framework will not be influenced by such architectural mismatches.

Inability for Multi-to-one Knowledge Inheritance. It is non-trivial to support absorbing knowledge from multiple teacher models by jointly recycling their model parameters. Instead, it is easy to implement for KI. As shown in our experiments, we demonstrate that under our framework, large PLMs can simultaneously absorb knowledge from multiple teachers.

Inability of Knowledge Inheritance for Domain Adaptation. Parameter recycling is hard to support continual learning, which makes large PLMs absorb knowledge from small ones in a lifelong manner. In real-world scenarios, numerous PLMs of different architectures are trained locally with

HyperParam	ACL-ARC & CHEMPROT	GLUE
Learning Rate	2×10^{-5}	$\{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}\}$
Batch Size	256	$\{16, 32\}$
Weight Decay	0.1	0.1
Max Epochs	10	10
Learning Rate Decay	Linear	Linear
Warmup Ratio	0.06	0.06

Table 8: Hyper-parameters for fine-tuning RoBERTa on ACL-ARC, CHEMPROT and GLUE.

	WB	CS	BIO
WB	100%	19.1%	25.6%
CS	19.1%	100%	22.5%
BIO	25.6%	22.5%	100%

Table 9: Domain proximity (vocabulary overlap) among three domains (WB, CS, BIO) discussed in this paper. Following (Gururangan et al., 2020), we create the vocabulary for each domain by considering the top 10k most frequent words (excluding stopwords).

Step	20k	40k	60k	80k	100k
$\alpha = 0.3$	8.68	7.29	6.90	6.57	6.26
$\alpha = 0.2$	7.27	6.47	5.95	5.68	5.46
$\alpha = 0.1$	6.71	5.74	5.35	5.06	4.86
$\alpha = 0$	6.13	5.21	4.83	4.57	4.36
KI	**5.69**	**5.17**	**4.78**	**4.52**	**4.32**

Table 10: Validation PPL for training $\text{RoBERTa}_{\text{BASE}}$ with different strategies. **KI** denotes our knowledge inheritance framework, where $\text{RoBERTa}_{\text{MEDIUM}}$ is chosen as the teacher.

different data. These small PLMs can be seen as domain experts, and it is essential that larger PLMs can continuously benefit from these existing PLMs efficiently by incorporating their knowledge so that larger PLMs can become omnipotent. As described before, it is easy to implement for our framework and we have demonstrated the effectiveness.

Model Privacy. Parameter recycling requires the availability of the parameters of an existing PLM, which may be impractical due to some privacy issues, e.g., GPT-3 only provides API access for prediction instead of the model parameters. Instead, our KI framework does not presume access to an existing model parameter since the predictions of the small model can be pre-computed and saved offline. This superiority will further make it possible for API-based online knowledge transfer.

F Comparing Label Smoothing and Knowledge Inheritance

Previous work shows the relation between label smoothing and knowledge distillation to some extent (Shen et al., 2021). To demonstrate that the success of our KI is not because of learning from a more smoothed target, we conduct experiments comparing both label smoothing and our KI in Table 10. Specifically, for label smoothing, PLMs optimize a smoothed target $\mathbf{y}_i^S = (1-\alpha) * \mathbf{y}_i + \alpha * \vec{\mathbf{1}}/(K-1)$, where $\alpha = 0$ denotes learning from scratch with no label smoothing, larger α means a more smoothed target for PLMs to learn from, K denotes the vocabulary size. Specifically, we choose α from $\{0.1, 0.2, 0.3\}$. It can be concluded from the results in Table 10 that adding label smoothing into the pre-training objectives of PLMs leads to far worse performance than the vanilla baseline, which shows that the improvements of our knowledge inheritance framework are non-trivial: larger PLMs are indeed inheriting the "knowledge" from smaller ones, instead of benefiting from optimizing a smoothed target, which imposes regularization.