A Short Study on Compressing Decoder-Based Language Models

Tianda Li*, **Yassir El Mesbahi***, **Ivan Kobyzev, Ahmad Rashid,**
Atif Mahmud, Nithin Anchuri, Habib Hajimolahoseini, Yang Liu,
Mehdi Rezagholizadeh
Huawei Noah's Ark Lab
{tianda.li, yassir.el.mesbahi1,ivan.kobyzev, ahmad.rashid, mehdi.rezagholizadeh}@huawei.com

Abstract

Pre-trained Language Models (PLMs) have been successful for a wide range of natural language processing (NLP) tasks. The state-of-the-art of PLMs, however, are extremely large to be used on edge devices. As a result, the topic of model compression has attracted increasing attention in the NLP community. Most of the existing works focus on compressing encoder-based models (tiny-BERT, distilBERT, distilRoBERTa, etc), however, to the best of our knowledge, the compression of decoder-based models (such as GPT-2) has not been investigated much. Our paper aims to fill this gap. Specifically, we explore two directions: 1) we employ current state-of-the-art knowledge distillation techniques to improve fine-tuning of DistilGPT-2. 2) we pre-train a compressed GPT-2 model using layer truncation and compare it against the distillation-based method (DistilGPT2). The training time of our compressed model is significantly less than DistilGPT-2, but it can achieve better performance when fine-tuned on downstream tasks. We also demonstrate the impact of data cleaning on model performance.

1 Introduction

Pre-trained Language Models (PLMs) have recently achieved great success on a wide variety of NLP problems [Peters et al., 2018, Devlin et al., 2019, Liu et al., 2019, Yang et al., 2020, Radford and Narasimhan, 2018, Radford et al., 2019]. With the rapidly increasing parameter count and training time, the state-of-the-art (SOTA) PLMs are becoming more challenging to be deployed on edge devices. In particular, RoBERTa-large has 355 million parameters, GPT-2-xl has 1.5 billion parameters, and the most recent GPT-3 [Brown et al., 2020] has 175 billion parameters. The importance of model compression methods is emergent in NLP [Gupta and Agrawal, 2021].

Generally speaking, the compression of a PLMs can be divided into two stages: initialization and fine-tuning. In the initialization stage, the compressed model's parameters can be either transferred from a larger pre-trained model [Sun et al., 2019, Passban et al., 2020] or pre-trained from scratch as a language model. Pre-training the smaller language models is cumbersome since typically knowledge is distilled from a larger teacher [Jiao et al., 2020, Sanh et al., 2020]. In the fine-tuning stage, the initialized compressed model is trained on a downstream task. In our work, we will investigate both stages of the compression.

A predominant solution for fine-tuning compressed PLMs is knowledge distillation (KD) [Rogers et al., 2020]. Most of the reported KD results in the literature [Hinton et al., 2015, Buciluǎ et al., 2006, Gao et al., 2018, Kamalloo et al., 2021, Rashid et al., 2020, Li et al., 2021, Haidar et al., 2021] are for encoder-based models such as BERT, RoBERTa. KD on decoder-based models [Radford et al.,

*Equal contribution

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

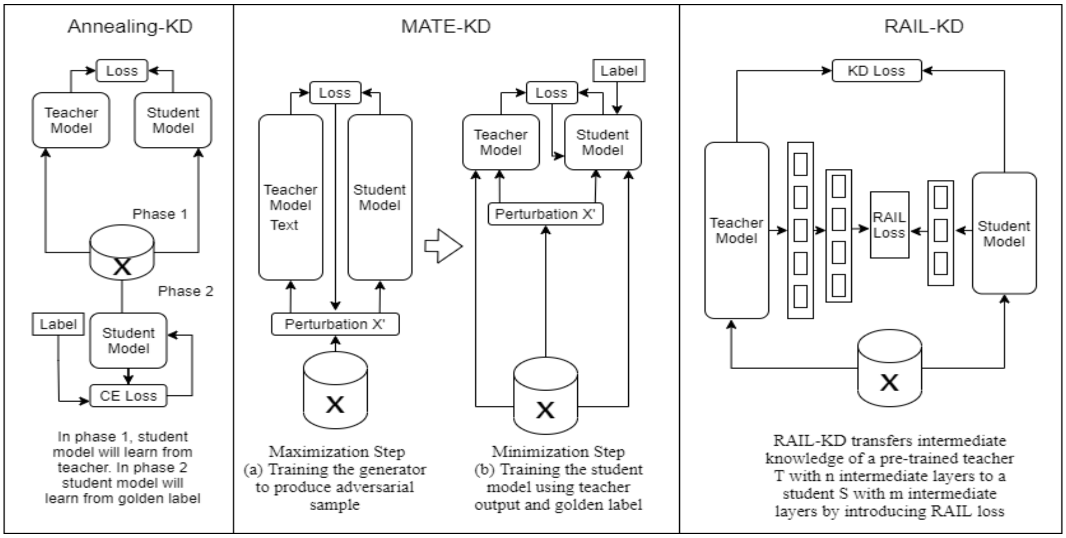

Figure 1: The overview of knowledge distillation techniques we apply in the paper: Annealing-KD, MATE-KD and RAIL-KD's.

2019] has not been investigated much. In this work we explore both teacher-based and teacher-free techniques aiming to improve compressed GPT-2 fine-tuning.

Pre-training of compressed encoder-based models has been extensively explored [Sanh et al., 2020, Xu et al., 2020, Sajjad et al., 2020]. However, DistilGPT2 [HuggingFace, 2019] is the only compressed GPT-2 model we found in the literature. The authors pre-trained DistilGPT-2 with KD using the original GPT-2 as a teacher, which results in a long training time. In our paper, we investigate pre-training without KD to significantly improve time efficiency.

Our contribution is three-fold:

1. We benchmark different SOTA teacher-based and teacher-free techniques for fine-tuning DistilGPT2 on downstream tasks.
2. We compare several truncation methods for pre-training initialization of the compressed GPT-2 model.
3. We conduct data cleaning of the OpenWebText dataset and pre-train our compressed model initialized with the best truncation technique. This pre-training scheme is time-efficient. At the same time, fine-tuning on downstream tasks reveal that our pre-trained model achieves better performance compared to DistilGPT-2.

2 Methodology

In this section, we introduce the techniques we applied for fine-tuning and pre-training compressed GPT-2 models. We start with knowledge distillation and teacher-free methods for fine-tuning, then we introduce layer truncation methods for initializing the student from the teacher, and finally, we discuss data-cleaning for efficient pre-training.

2.1 Fine-tuning with and without a teacher

Here, we discuss the techniques we applied to improve fine-tuning of DistilGPT-2 model on downstream tasks.

2.1.1 KD Methods

Hinton et al. [2015] proposed KD as a way to improve the training of a small neural network (student). Given a bigger model (teacher), KD adds a specific loss term to the loss function of the student aiming to push the student's predictions close to the teacher's. In this paper, we consider four different KD methods: 1) Vanilla KD, 2) Annealing-KD, 3) MATE-KD and 4) RAIL-KD. The overview of these models is given in Figure 1,

For **Annealing-KD** [Jafari et al., 2021], the student is trained in two phases. During phase 1, the student model learns only from the teacher. Here the temperature controls the smoothness of the teacher's output, annealing it from easy-to-learn to the actual sharp distribution. During phase 2, the student model is trained only on the ground-truth label.

For **MATE-KD** [Rashid et al., 2021], the training process has two steps: maximization and minimization. At the maximization step, a generator is trained to produce perturbed input for both student and teacher models. The target of this stage is to produce the input that can maximize the divergence between the teacher and student output. At the minimization step, the student model is trained to approximate the teacher's output.

For **RAIL-KD** [Haidar et al., 2021], during the training we transfer the knowledge from teacher's intermediate layers to student's intermediate layers. In our case, the 6-layers student model is distilled from 12 layers GPT2 model.

2.1.2 Teacher-free Methods

Here, we describe the most commonly used teacher-free techniques.

Label Smoothing (LS) Szegedy et al. [2015] proposed this method to improve the training of a classifier. For this, a cross-entropy loss should be calculated with smoothed labels rather than one-hot labels. The smoothed labels are given by:

$$y' = (1-\alpha)y + \alpha u, \tag{1}$$

where $u(K) = 1/K$ is the uniform distribution on K classes, y is the one-hot golden label, and α is a parameter between 0 and 1 controlling the sharpness of the resulting soft label.

TF-reg The TF-reg [Yun et al., 2020] technique is very similar to label smoothing, the only difference is that TF-reg switches the uniform distribution u in Equation 1 to the label-dependent distribution $p(k)$, defined by:

$$p^c(k) = \begin{cases} a, \text{ if } k = c \text{ (is the correct label)} \\ \frac{1-a}{K-1}, \text{ otherwise.} \end{cases} \tag{2}$$

Where a is a parameter between 0 and 1. TF-reg has two parameters (a and α) instead of just one (α) which allows for better tuning. The smoothed label for x in TF-reg is given by:

$$y' = (1-\alpha)y + \alpha p^{c(x)}, \tag{3}$$

where $c(x)$ is the correct label of the sample x.

Self-distillation (Self-KD) Self-KD [Furlanello et al., 2018] is a variation of the KD method, in which we first fine-tune a copy of the student on the dataset and then freeze it. This copy serves as a teacher during the training.

2.2 Student Layers Initialization

In this section, we introduce the student's layers initialization from the teacher. Sajjad et al. [2020] shows that an easy way of compressing pre-trained large models is to simply "truncate" them by dropping some layers. Inspired by that, we propose our pruning techniques and list the top 2 pruning strategies below (The overall six pruning techniques and results are introduced in Appendix A.1)

Uniform Selection (Uni) We select layers to copy uniformly, starting from the first layer. For example, if the teacher has 12 layers, we would initialize the student (6-layers model) by teacher layers 0, 2, 4, 6, 8 and 10.

Algorithm 1 Details of Pseudo-uniform selection for layer initialization from a larger model with n layers to a smaller model with k layers.

Require: $n > k$; $n \bmod k = 0$; $n \bmod 2 = 0$
Ensure: Pseudo-uniform selection of length k
  step $\leftarrow \lfloor \frac{n}{k} \rfloor$
  start $\leftarrow 0$
  end $\leftarrow n - 1$
  selection $\leftarrow []$
  while start $\leq$ end **do**
    selection $\leftarrow$ selection $+$[start]
    selection $\leftarrow$ selection $+$[end]
    start $\leftarrow$ start $+$ step
    end $\leftarrow$ end $-$ step
  end while

Pseudo-uniform Selection (Psudo) This strategy is inspired from DistilBert's paper [Sanh et al., 2020], where they initilaize their model (DistilBert) with teacher's (Bert-base) layers 0, 2, 4, 7, 9 and 11. In contrast with uniform selection, we make sure first and last layers are always selected. A generalization of this strategy can be described by the Algorithm 1, where n stands for the total number of teacher's layers and k is number of layers we want to select (also number of student's layers).

3 Experiments

3.1 Data

OpenWebText is an open-source recreation of the WebText corpus (on which the GPT-2 model was trained). We use this data for pre-training our compressed model. Original WebText contains over 8 million documents for a total of 40 GB of text. In our experiment, we only used a fraction of these data.

We assess compressed models on several downstream tasks. First, we employ the Wikitest103 dataset [Merity et al., 2016] to fine-tune a compressed model as a language model and measure the performance with perplexity score (the lower - the better). Then, we fine-tune a compressed model as a classifier on 6 out of 8 tasks in the SuperGLUE [Wang et al., 2019a] benchmark. Moreover, we evaluate the fine-tuning of a compressed model as a classifier on 7 out of 9 tasks of the General Language Understanding Evaluation (GLUE) [Wang et al., 2019b] benchmark.

3.2 Fine-tuning on GLUE

We apply KD and teacher-free techniques described in Section 2.1 to fine-tune DistilGPT-2 model on GLUE tasks. The results are in Table 1 and Table 2. We can see that Annealing-KD, MATE-KD, and RAIL-KD all outperform VanillaKD. Interestingly, regular fine-tuning itself is a strong baseline that performs comparatively well to vanilla KD, and it even outperforms the LS and TF-reg techniques. Self-KD performance is comparable with other teacher-free techniques. RAIL-KD performs worse than MATE-KD and Annealing-KD, which indicates that distilling intermediate layers doesn't have an advantage over data augmentation or annealing scheduling. MATE-KD performs the best among four KD techniques. One should notice that this pattern is slightly different from the fine-tuning of Bert-based models [Li et al., 2021]. One possible explanation might be that decoder-based models are more sensitive to hyper-parameters. Data augmentation is a more robust way to improve the student model's performance.

3.3 Experiments on Layer Truncation

First, we initialize a 6-layer GPT-2 model with the initialization techniques described in section 2.2. Then, we pre-train the models on fraction of the OpenWebText dataset. For these experiments, we use either 4 or 8 GPUs and make use of the DeepSpeed framework [Rajbhandari et al., 2019, Rasley et al., 2020, Ren et al., 2021, Rajbhandari et al., 2021] to accelerate the training process. Then, we report

Table 1: Dev set results of teacher-free methods on GLUE. We benchmark pure finetuning of DistilGPT2 (first line) with teacher-free regularisation training and Self-KD. The last line is the performance of 12 layers GPT-2 model.

Evaluated Model	CoLA	RTE	MRPC(f1)	SST-2	MNLI	QNLI	QQP	Average
DistilGPT2	39.0	65.2	87.9	91.5	86.5	79.9	89.7	77.1
LS	38.9	64.8	87.3	91.6	86.6	80.1	89.6	77.0
TF-reg	38.7	65.1	87.4	91.4	86.9	80.2	89.6	77.0
Self-KD	39.7	64.7	87.3	90.9	87.0	80.5	89.8	77.2
GPT-2	43.2	66.8	87.6	92.2	82.3	88.6	89.5	78.6

Table 2: Dev set results of KD methods on GLUE. Here the student is DistilGPT2 and the teacher is 12 layers GPT-2. See Table 1 for the student's and teacher's performance.

Teacher	Evaluated Model	CoLA	RTE	MRPC(f1)	SST-2	MNLI	QNLI	QQP	Average
GPT-2	$\text{VanillaKD}_{\text{DistilGPT2}}$	39.3	65.7	88.0	90.7	79.6	86.8	89.4	77.1
GPT-2	$\text{RailKD}_{\text{DistilGPT2}}$	39.4	66.4	88.1	91.2	80.6	87.3	89.9	77.6
GPT-2	$\text{AnnealingKD}_{\text{DistilGPT2}}$	41.6	67.1	86.8	92.0	80.8	**87.8**	89.4	77.9
GPT-2	$\text{MateKD}_{\text{DistilGPT2}}$	**42.1**	**67.5**	**88.8**	**92.0**	**81.6**	87.7	**90.0**	**78.5**

the zero-shot performance of the pre-trained models in Table 3. Our compressed model outperforms DistilGPT-2 even when it's trained on 50% of the dataset. Also, our pre-training is tremendously time-efficient.

Table 3: Truncated models' zero-shot perplexity scores on Wikitext103 after pretraining. Models are truncated with techniques from Section 2.2, pre-trained on fraction of the OpenWebtext dataset, and then evaluated on Wikitext103 test set. All the truncated models in this table have 6 layers.

Models		Pretraining on fraction of OpenWebtext					
Model index	Teacher	Strategy	PPL	% of the dataset	Epochs	# GPUs	Time (h)
0	DistilGPT2	Pre-train	**45.26**	100	4	8	-
1	GPT2	Psudo	56.38	10	3	4	8
2	GPT2	Psudo	46.91	50	3	8	38
3	GPT2	Uni	**45.19**	50	3	8	35
4	GPT2-xl	Psudo	54.70	10	3	4	24
5	GPT2-large	Psudo	59.32	10	3	4	17

Next, we fine-tuned the pre-trained models on the Wikitext103 and put the perplexities in Table 4. We can see that the perplexity achieved by GPT2-psudo is still worse than DistilGPT2's. The 6-layer model truncated from GPT2-xl teacher and initialized with Pseudo-uniform truncation method (GPT2-xl-psudo) reaches a perplexity close to DistilGPT2's despite being pre-trained on a fraction (10%) of the OpenWebtext dataset (but it is not comparable to DistilGPT2 since it has three times more parameters).

3.4 Effect of Data Cleaning on Pre-training

We found that the OpenWebText dataset contains a significant amount of noisy samples. Some of these are HTML code, others are pure noise (concatenation of special characters). To alleviate the problem of noisy samples, we implemented a program that automatically inspects the samples, clears out HTML code and short sentences, eliminates sentences with a high ratio of non-alphanumerical characters (more than 10%) and duplicates. Using the above algorithm, we managed to dramatically

Table 4: Truncated models' perplexity scores on Wikitext103 after fine-tuning. Once models are pre-trained on fractions of OpenWebText (table 3), they are fine-tuned on Wikitext103 train set and then evaluated on Wikitext103 test set. All the models compared in this table have 6 layers and the model index indicates corresponds the one in table 3.

Models		Fine-tuning pretrained models				
Model index	Teacher	Strategy	PPL	Epochs	# GPUs	Time (h)
0	DistilGPT2	Pre-train	**21.13**	6	4	2
1	GPT2	Psudo	23.44	12	4	5
2	GPT2	Psudo	22.67	6	8	2
3	GPT2	Uni	22.61	6	4	5
4	GPT2-xl	Psudo	**21.30**	6	4	5
5	GPT2-large	Psudo	23.44	6	4	5

reduce the size of the OpenWebText dataset (from $332,011,430$ to $114,366,559$ samples, or by 65.5%).

We pre-train a 6-layer GPT2 model (initialized with the pseudo-uniform strategy) on the cleaned dataset, then we fine-tune it on the Wikitext103 and several datasets from the SuperGLUE and compare the results with the model that has been pre-trained on the full dataset. For Wikitext103, we measure zero-shot (ZS) and post-fine-tuning (FT) perplexities (PPL). Results are shown in Table 5.

Table 5: Results after pre-training on regular/cleaned OpenWebText dataset. Truncated models are pretrained on both datasets (original and cleaned) and their performance measured on several tasks.

Models	Pretrain dataset	Pretrain time (h)	Scores after fine-tuning									
			Wikitext103		BoolQ	Copa	CB		Rte	Wic	Wsc	
			PPL (ZS)	PPL (FT)	Acc.	Acc.	Acc.	F1	Acc.	Acc.	F1	Acc.
DistilGPT2	Regular	768	45.26	21.13	71.16	56	73.21	61.45	62.45	63.6	63.46	77.64
GPT2-Psudo	Cleaned	**42**	45.93	22.42	**70.67**	**58**	**78.57**	**65.15**	**63.53**	**60.3**	**63.46**	**77.64**
	Regular	49	**44.51**	**22.29**	68.07	56	71.42	49.77	58.48	59.24	59.61	73.74

We can see that cleaning the dataset helps reducing training time while allowing for achieving comparable or better performance.

4 Conclusion

In this work, we aim to compress GPT-2. First, we benchmark current SOTA KD and teacher-free methods on DistilGPT2 and pick the best performing one. Then, we explore truncation methods for the initialization of the student model from the teacher model's parameters. Specifically, we propose a pseudo-uniform strategy that outperforms alternative initializations in the language modeling experiments on Wikitext-103. Finally, we conduct data cleaning on the OpenWebText dataset and pre-trained our compressed model. To test the effectiveness of our strategy we carried out the experiments on Wikitext-103 and 6 out of 8 SuperGLUE Benchmark datasets. Our pre-trained model outperforms DistilGPT-2 on 5 out of 7 downstream tasks, yet it is significantly more time-efficient. For the future direction, we will evaluate our initialization strategy along with KD methods and investigate if the pre-training of our compressed GPT-2 can be improved even more.

References

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess,

J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks, 2018.

J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56, 2018. doi: 10.1109/SPW.2018.00016.

M. Gupta and P. Agrawal. Compression of deep learning models for text: A survey, 2021.

M. A. Haidar, N. Anchuri, M. Rezagholizadeh, A. Ghaddar, P. Langlais, and P. Poupart. Rail-kd: Random intermediate layer mapping for knowledge distillation, 2021.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.

HuggingFace. Distilgpt2, 2019. URL `https://huggingface.co/distilgpt2`.

A. Jafari, M. Rezagholizadeh, P. Sharma, and A. Ghodsi. Annealing knowledge distillation, 2021.

X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372. URL `https://www.aclweb.org/anthology/2020.findings-emnlp.372`.

E. Kamalloo, M. Rezagholizadeh, P. Passban, and A. Ghodsi. Not far away, not so close: Sample efficient nearest neighbour data augmentation via MiniMax. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3522–3533, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.309. URL `https://aclanthology.org/2021.findings-acl.309`.

T. Li, A. Rashid, A. Jafari, P. Sharma, A. Ghodsi, and M. Rezagholizadeh. How to select one among all? an extensive empirical study towards the robustness of knowledge distillation in natural language understanding, 2021.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. URL `http://arxiv.org/abs/1609.07843`.

P. Passban, Y. Wu, M. Rezagholizadeh, and Q. Liu. ALP-KD: Attention-Based Layer Projection for Knowledge Distillation, 2020.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054, 2019.

S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He. Zero-infinity: Breaking the GPU memory wall for extreme scale deep learning. *CoRR*, abs/2104.07857, 2021. URL `https://arxiv.org/abs/2104.07857`.

A. Rashid, V. Lioutas, A. Ghaddar, and M. Rezagholizadeh. Towards zero-shot knowledge distillation for natural language processing. *arXiv preprint arXiv:2012.15495*, 2020.

A. Rashid, V. Lioutas, and M. Rezagholizadeh. MATE-KD: Masked adversarial TExt, a companion to knowledge distillation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1062–1071, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.86. URL `https://aclanthology.org/2021.acl-long.86`.

J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He. *DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters*, page 3505–3506. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379984. URL `https://doi.org/10.1145/3394486.3406703`.

J. Ren, S. Rajbhandari, R. Y. Aminabadi, O. Ruwase, S. Yang, M. Zhang, D. Li, and Y. He. Zero-offload: Democratizing billion-scale model training. *CoRR*, abs/2101.06840, 2021. URL `https://arxiv.org/abs/2101.06840`.

A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. doi: 10.1162/tacl_a_00349. URL `https://www.aclweb.org/anthology/2020.tacl-1.54`.

H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov. Poor man's BERT: smaller and faster transformer models. *CoRR*, abs/2004.03844, 2020.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

S. Sun, Y. Cheng, Z. Gan, and J. Liu. Patient knowledge distillation for bert model compression, 2019.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537, 2019a. URL `http://arxiv.org/abs/1905.00537`.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019b.

C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou. Bert-of-theseus: Compressing bert by progressive module replacing. In *EMNLP*, 2020.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

S. Yun, J. Park, K. Lee, and J. Shin. Regularizing class-wise predictions via self-knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13873–13882, 2020.

A Appendix

A.1 Student Layer Initialization

Overall, we tried 6 pruning strategies as listed below:

Uniform selection We select layers to copy uniformly, starting from the first layer. For example, if the teacher has 12 layers, we would initialize the student (6-layers model) by teacher layers 0, 2, 4, 6, 8 and 10.

Variant of uniform selection We select layers to copy uniformly, but the last layer we select should always be the one before the last of the teacher's layers.

Pseudo-uniform selection This strategy is inspired from DistilBert's paper, where they initilaize their model(DistilBert) with teacher's (Bert-Base) layers 0, 2, 4, 7, 9 and 11. In contrast with uniform selection, we make sure first and last layer are always selected. where n represents the total number of teacher's layers and k the number of layers we want to select (also number of student's layers).

Bottom half selection This is a generalization of one of the strategies describe in their paper. We uniformly select from the bottom-half section of teacher's layers. As an example, for a 36-layer teacher, we select uniformly from its first 13 layers. In the particular case of a 12-layer teacher, we pick the first six layers.

Top half selection Similar to the bottom half selection, this strategy consists in selecting layers uniformly from the top layers of the teacher. For example, for a 48-layer teacher, we would select uniformly from its top 24 layers. In the particular case of a 12-layer teacher, we pick the last six layers.

Random selection We implement this method to have a baseline to compare with. We randomly pick layers from the teacher, sort them by index and use them to initialize the student.

We apply the above pruning techniques on the GPT-2 models and measure their perplexities on Wikitext103 after fine-tuning. To easily identify the models we are training, we add to the names of original GPT-2 a suffix indicating which layer selection strategy was used to initialize it. Table 6 shows the correspondence between suffixes and pruning strategies. Table 7 displays some characteristics of the resulting models, as well as their performance on Wikitext103 test set. We list several

Table 6: Pruning strategies suffixes

Strategy	Suffix
Uniform	uniform
Uniform (variant)	uniform-2
Pseudo-uniform	psudo
Bottom-half	6bh
Top-half	6th
Random	random

observations from fine-tuning results:

The best validation curves come from the "uniform/bottom-half/pseudo-uniform" strategies We observe a better convergence in these settings, which is similar to previous paper reported results.

The "pseudo-uniform" strategy achieves the best test results after fine-tuning Perplexities are the lowest in this setting, as shown in table 7.

The "bottom-half" strategy outperforms the "top-half" According to previous report, this is due to the fact that bottom layers tend to learn embeddings and general representations while top layers are more task-specialized.

Overall, as a conclusion of this experiment the pseudo-uniform initialization scheme clearly allows for better generalization (table 7). We can also conclude that pre-training plays a significant role in

Table 7: Truncated models' perplexity scores on Wikitext103 test set

Models	# layers	Teacher's layers	# heads	Hidden size	# parameters	Epochs	PPL
DistilGPT2	6	———	12	768	81 M	6	**21.13**
GPT2-xl-psudo	6	$\{0, 8, 16, 31, 39, 47\}$	25	1,600	266 M	6	**22.8**
GPT2-xl-uniform		$\{0, 9, 18, 27, 36, 45\}$					25.37
GPT2-xl-uniform-2		$\{0, 10, 20, 30, 40, 46\}$					26.07
GPT2-xl-6bh		$\{0, 4, 9, 13, 18, 23\}$					25.54
GPT2-xl-6th		$\{24, 28, 33, 37, 42, 47\}$					47.51
GPT2-xl-random		$\{2, 11, 17, 25, 34, 40\}$					42.48
GPT2-large-psudo	6	$\{0, 6, 12, 23, 29, 35\}$	20	1,280	183 M	6	**23.79**
GPT2-large-uniform		$\{0, 7, 14, 21, 28, 35\}$					27.68
GPT2-large-uniform-2		$\{0, 8, 16, 24, 32, 34\}$					27.78
GPT2-large-6bh		$\{0, 2, 4, 7, 9, 12\}$					35.88
GPT2-large-6th		$\{13, 17, 21, 26, 30, 35\}$					74.15
GPT2-large-random		$\{5, 25, 26, 29, 32, 34\}$					79.68
GPT2-medium-psudo	6	$\{0, 4, 8, 15, 19, 23\}$	16	1,024	128 M	6	**28.09**
GPT2-medium-uniform		$\{0, 4, 8, 12, 16, 20\}$					35.83
GPT2-medium-uniform-2		$\{0, 5, 10, 15, 20, 23\}$					35.4
GPT2-medium-6bh		$\{0, 2, 4, 6, 8, 10\}$					36.2
GPT2-medium-6th		$\{12, 14, 16, 18, 20, 23\}$					88.71
GPT2-medium-random		$\{1, 3, 7, 11, 15, 21\}$					75.13
GPT2-psudo	6	$\{0, 2, 4, 7, 9, 11\}$	12	768	81 M	6	**26.14**
GPT2-uniform		$\{0, 2, 4, 6, 8, 10\}$					26.6
GPT2-6bh		$\{0, 1, 2, 3, 4, 5\}$					29.86
GPT2-6th		$\{6, 7, 8, 9, 10, 11\}$					49.61

aligning the weights and making convergence faster: the performance of DistilGPT2 supports this claim.