Efficient Distillation of Transformers via Self-Teaching

Tianjian Li, Bismarck Odoom
Department of Computer Science,
Johns Hopkins University
Baltimore, MD
{tli104, bodoom1}@jhu.edu

Abstract

Knowledge Distillation is a method to train compressed neural networks with fewer parameters that match the performance of large models. However, Knowledge Distillation requires users to pre-define the architecture of the smaller model and requires more GPU memory than direct training of the large model since during the distillation process, we need to additional store the parameters and gradients of the small model. In this work, we propose a novel distillation method called **self-teaching** for Transformer architecture models that require neither pre-definition of the student model nor additional GPU memory. We distill the full model into a sub-component of the full model that only uses the bottom few layers and train the intermediate hidden representations to match the output of the full model. We show the effectiveness of our distillation method in fine-tuning BERT-base on GLUE tasks, fine-tuning T5-large on summarization, as well as training a standard Transformer from scratch on IWSLT 14 machine translation. Our results indicate that self-teaching effectively distills neural networks trained from scratch and publicly available pre-trained models to a smaller size with a minimal performance drop and does not require additional compute. We release all of our code for reproducibility[1].

1 Introduction

Large language models exhibit extraordinary abilities in natural language understanding [28, 29] and generation [18]. Studies have shown that language models unlock more 'emergent abilities' when scaled up to billions of parameters [31]. The increase in parameters poses great demand on our resources. To compress these large models (i.e. the teacher model) into smaller models (i.e. the student model) with minimal performance drop, a standard approach is using Knowledge Distillation (KD) [6]. KD adds an auxiliary loss function that penalizes the KL divergence between the output probability distribution of the teacher model and the student model for being too large. By enforcing the output distribution of the teacher and student to be similar, KD effectively distills what the teacher learns to the student.

Despite the success of KD in compressing pre-trained large language models [22], it requires you to have additional computational resources to store both the teacher and the student model so that they can be trained and distilled simultaneously. In a practical scenario, we want to use the largest possible model that fits in our GPU memory. It becomes important to perform distillation without requiring additional memory.

One effective way of compressing models without using additional resources is **pruning**. Unstructured pruning removes individual parameters that do not contribute [16]. However, the model still needs to store the weight matrices and perform the full matrix multiplication during inference. Another

[1]https://github.com/tianjianl/self_supervised_sp23/tree/master/efficient-distillation

Preprint. Under review.

	No Additional Copies of Params	No Iterative Retraining
Knowledge Distillation [6]	✘	✔
Iterative Pruning [4]	✔	✘
Iterative Pruning + Distillation [34, 10]	✘	✘
Self Teaching (Ours)	✔	✔

Table 1: An overview of different methods to learn a smaller model with comparable performance. Our method does not require you to store an additional copy of the model parameters nor requires iterative estimation and retraining of the model.

line of work prunes individual components [4, 15, 12, 38] to achieve faster training and inference. However, structured pruning all require careful estimation of parameter importance [13, 14, 42], and some of the structure pruning methods require iterative pruning to compress to a desired amount of parameters [4, 12]. We provide a detailed comparison between our method and other state-of-the-art methods that learns smaller models in Table 1.

In this paper, we explore distilling a large transformer model by pruning individual layers. During training, we match the output probabilistic distribution of an intermediate layer to the final output distribution. Moreover, we utilize robust fine-tuning methods [1, 37] to tune the student model after distillation to close the performance gap even further.

To sum up, our contribution is three-fold:

- We propose a novel distillation framework that neither requires designing the architecture nor additional RAM to store the weights of the student model.
- We perform experiments on task-specific distillation on fine-tuning BERT [2] small (110M) and Large (336M) as well as T5-large [20] (1.3B) to verify that our method generalizes across different model architectures and sizes.
- We further close the performance gap between the student and the teacher model when the teacher model is large by proposing an optional enhancement of our method.

2 Preliminaries

2.1 Importance Estimation of Parameters

The prominent way to estimate the importance of a single or group of parameters θ_j in both pruning and training is measured by the difference in loss before and after θ_j is zeroed out, we use $\theta_{j=0}$ to denote the set of parameters when $\theta_j = 0$.

$$\mathcal{I}(\theta_j) = |\mathcal{L}(\theta) - \mathcal{L}(\theta_{j=0})| \tag{1}$$

Approximating $\mathcal{L}(\theta_j)$ using a second-order Taylor expansion yields:

$$\mathcal{L}(\theta_{j=0}) \approx \mathcal{L}(\theta) - \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}(\theta_j - 0) + \frac{1}{2}(\theta_j - 0)^\top \frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta_j^2}(\theta_j - 0) \tag{2}$$

If we view the second-order term as negligible and substituting 2 into 1, we get

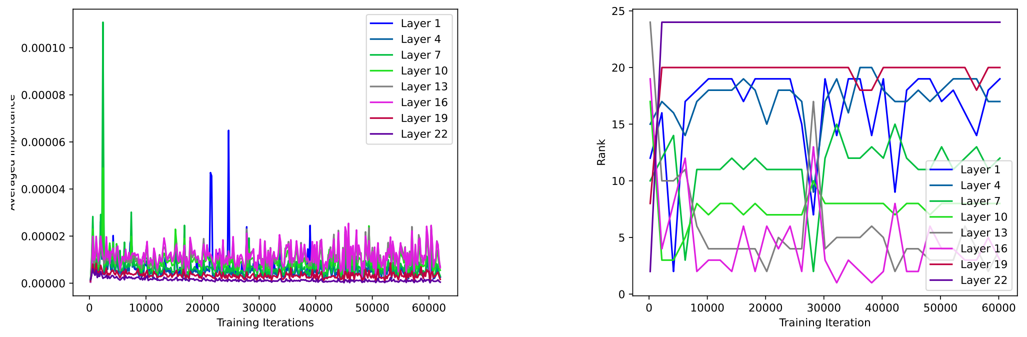
$$\begin{aligned}\mathcal{I}(\theta_j) = |\mathcal{L}(\theta) - \mathcal{L}(\theta_{j=0})| &\approx \left| \cancel{\mathcal{L}(\theta)} - \left(\cancel{\mathcal{L}(\theta)} - \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}(\theta_j - 0) \right) \right| \\ &= \left| \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}(\theta_j - 0) \right|\end{aligned}$$

This importance metric was originally proposed to prune convolutional neural networks [16, 17, 26], which is then extended to Transformer pruning [11, 37, 42]. In this paper, we use this loss-preserving metric $\mathcal{I}(\theta_j)$ as the importance metric of one or a set of parameters.

2.2 Preliminary Experiments: Deeper Layers are Unimportant

With the importance score of each parameter defined in the previous section, we plot out the averaged importance of each layer in BERT-large fine-tuned on the MNLI task [32] at 1a and their rank at 1b. We can see that the deeper layers have relatively low importance compared to the other layers from a loss-preserving perspective. In figure 1a the purple layer is always at the bottom throughout training and in figure 1b the purple and red, which correspond to layers 22 and 18 are at the top, meaning that their importance ranks the lowest during training.

Additionally, we plotted out the importance scores throughout the training of a T5-Large model trained on the CNN/DM dataset [23] at figure 5 in the appendix. We found that there is no significant importance difference between the encoder and decoder, but found that the deeper layers of the encoder and the shallow layers of the decoder are generally more important, which echoes our previous observations of the middle layers are more important than layers on both ends.



(a) Parameter Importance plot

(b) Importance Rank

Figure 1: Importance of Individual Layers of BERT-large (24 layers, 330M parameters)

The observation that deeper layers are generally unimportant motivates our distillation method of distilling from the full model to the first few layers to achieve model compression without additional GPU memory.

For a more intuitive view of how unimportant the deeper layers are, we plotted the importance score throughout training of BERT-large (24 layers) on the MNLI task in figure 2, where a deeper color corresponds to a larger training iteration. We observe that the importance of individual layers peaks around the middle layers (10-15) and plummets after layer 18, which motivates our method to prune out the deeper layers to construct a student model.

3 Method

We describe our self-teaching method at §3.1 and an optional more fine-grained version of self-teaching that achieves better performance with more compute in detail at §3.2.

3.1 Self-Teaching

In this section, we describe our self-teaching method in detail. Given a neural network f parameterized by θ, and an input x, the neural network maps the input to a probability distribution $f_\theta(x) = \hat{y}$. The standard training objective minimizes the cross-entropy between the output probability distribution and the ground-truth label y.

$$\mathcal{L}_{CE}(\theta) = \text{CE}(y, f_\theta(x)) = \sum_i y_i \log \hat{y}_i$$

Where y_i and $\hat{y}_i$ is the probability for the ith label.

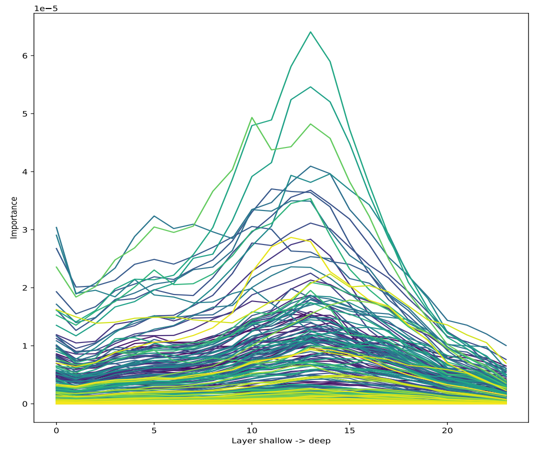


Figure 2: Importance scores of individual layers of BERT-large throughout training on the MNLI dataset. A deeper color corresponds to a higher training iteration.

Knowledge Distillation [6] minimizes the KL Divergence between the output of a teacher model f and the output of a student model g for the same input x, assume $g'_\theta(x) = \hat{z}$:

$$\mathcal{L}_{KD}(\theta, \theta') = \mathrm{KL}(f_\theta(x) \| g_{\theta'}(x)) = \sum_i \hat{y}_i \log \frac{\hat{y}_i}{\hat{z}_i}$$

In our self-teaching method, the student model is composed of the first n layers of the teacher model. An illustration of our proposed method and a comparison between other distillation pipelines is available in figure 2. 2a depicts the standard knowledge distillation method. 2b depicts self-distillation, which is knowledge distillation with the teacher and student model being architecturally identical. 2c depicts our self-teaching method that jointly minimizes the KL divergence between the final output and the output of an intermediate layer:

$$\mathcal{L}(\theta) = \mathcal{L}_{CE}(\theta) + \alpha \cdot \mathrm{KLDiv}(f_\theta(x), f_\theta^n(x))$$

where f_θ^n is the sub-model composed of only the first n layers of f_θ.

3.2 Gradual Self-Teaching

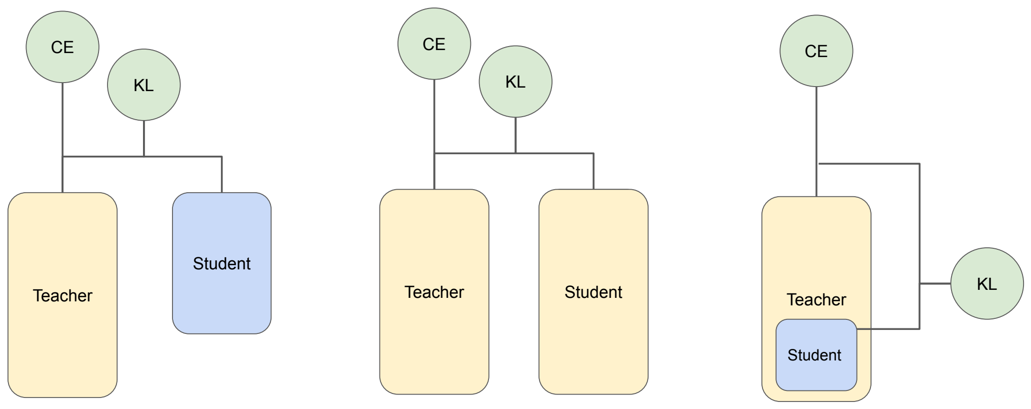
Empirically, we find that pruning too many layers from larger models results in a larger performance drop. We hypothesize that even the unimportant parameters, when aggregated, can contain important information for the model to make predictions. Therefore, we propose a more fine-grained version of self-teaching: gradual self-teaching, which iteratively prunes layers from the large model to achieve minimize the performance drop from pruning too many layers.

Formally, we first pre-define a sequence of student layers $s_0 > s_1 > s_2 > ... > s_n$. Then we first distill the large model to the first student s_0 by jointly minimizing the prediction loss and distillation loss until convergence

$$\mathcal{L}(\theta) = \mathcal{L}_{CE}(\theta) + \alpha \mathcal{L}_{KD}(f_\theta(x), f_\theta^{s_0}(x))$$

then on the next iteration, we treat the first student s_0 as the teacher, and we match the output of the second student to the teacher $f_\theta^{s_0}$. After a student is trained until convergence, we treat it as the teacher in the next iteration. At iteration t, we train on the following objective to convergence:

$$\mathcal{L}_t(\theta) = \mathcal{L}_{CE}\left(f_\theta^{s_{t-1}}(x)\right) + \alpha \mathcal{L}_{KD}\left(f_\theta^{s_{t-1}}(x), f_\theta^{s_t}(x)\right)$$



(a) Knowledge Distillation [6] (b) Self-Distillation [5] (c) Self-Teaching (Ours)

Figure 3: Illustration of different distillation methods: Self-Distillation is when the Teacher and student are models with identical structures but different initializations.

4 Experiments

We evaluate self-teaching on two different scenarios: task-specific distillation: distilling pre-trained encoder models and distilling encoder-decoder models trained for machine translation from scratch.

4.1 Setup

Distillation of Pre-trained Encoders: We use the Huggingface [33] implementation of BERT [2] and experiment on two different sizes of BERT: BERT-base (110M parameters) and BERT-large (340M) parameters. We train task-specific student models on five different tasks in the GLUE benchmark [28] to verify that self-teaching generalizes across different tasks and model sizes. We report the detailed hyper-parameter configuration at table 6 in the appendix.

Distillation of Encoder-Decoder Models: We use the official fairseq [19] implementation of Transformers [27]. We experiment by pruning layers of pruning the same number of layers in both the encoder and the decoder. We use five different languages (De, Ar, Fa, Es, He) from the IWSLT 14 machine translation dataset and train machine translation models from selected languages to English.

We also experiment on task-specific distillation on T5-large [20], a state-of-the-art encoder-decoder model with 1.3B parameters. We use the official Huggingface implementation [33] and experiment with two summarization tasks: CNN/Daily Mail [23], which contains 90k news articles and their summaries, and WikiLingua [3], which contains over 100k English WikiHow instructions and summaries.

4.2 Baselines

Table 2 reports the results of fine-tuning BERT-base and BERT-Large on six GLUE tasks. Our reproduced results are in line with the reported results from [2].

	RTE	MRPC	CoLA	SST-2	QNLI	QQP
BERT-Base (Ours)	66.4	88.1	55.0	93.0	90.7	90.2
BERT-Base (Reported)	66.4	88.9	52.1	93.5	90.5	89.6
BERT-Large (Ours)	70.1	89.1	61.3	94.2	92.4	91.1
BERT-Large (Reported)	70.1	89.3	60.5	94.9	92.7	91.3

Table 2: Baseline results on the development set fine-tuning of GLUE tasks, reported results are from [2].

Model	RTE	QNLI	CoLA	MRPC	SST-2	QQP	Avg.
Full 12 Layers (110M)	66.4	90.7	55.0	88.1	93.0	90.2	80.6
DistillBERT [21] (66M)	59.9	89.2	51.3	87.5	91.3	88.5	78.0
First 9 Layers (66M)	**66.0**	**90.3**	**54.7**	**87.7**	**92.1**	**89.4**	**80.0**
First 8 Layers (58M)	65.7	89.1	55.2	86.2	91.9	90.2	79.7
First 7 Layers (51M)	63.5	88.1	51.2	85.6	91.3	89.9	78.3
First 6 Layers (44M)	63.2	86.9	43.3	84.5	89.4	87.2	76.0

Table 3: Results of self-teaching on BERT-base. Our method outperforms DistillBERT using the same amount of parameters.

Model	CoLA (Mcc.)	RTE (Acc.)	MRPC (Acc.)
Full 24 Layers (336M)	61.3	72.6	89.3
First 20 Layers (254M)	58.6	71.1	85.2
First 16 Layers (203M)	60.4	70.0	83.7
First 12 Layers (183M)	54.1	63.9	72.6
First 12 Layers w/o gradual (183M)	31.0	57.4	66.5

Table 4: Results on BERT-large

4.3 Distillation of BERT-base

We report the results of distilling BERT-base (110M parameters) with different student layers at Table 3. Self-teaching outperforms DistillBERT by using fewer parameters. Moreover, we highlight that we do not require additional GPU memory. When only half of the layers are kept after distillation (6 layers), we see that the average performance drop is 5.7% (80.6 $\rightarrow$ 76.0). Furthermore, if we constrain the number of parameters to be the same as DistillBERT [21] (66M), self-teaching outperforms DistillBERT by 2 points on average on our tested tasks. However, we acknowledge that our method is doing task-specific distillation while DistillBERT is task-agnostic.

4.4 Gradual Self-Teaching on BERT-large

We empirically observe that directly distilling from the top layer leads to large performance degradation when the model is larger. Direct self-teaching does not scale well to larger models. To mitigate this issue, we propose to gradually distill from the top layer to our final layer by defining a distillation schedule. For example, if the schedule is $24 \rightarrow 20 \rightarrow 16 \rightarrow 12$, we first use the entire model as the teacher and the sub-model composed of the first 20 layers as the student and so on.

We report the results when using a distillation schedule of $24 \rightarrow 20 \rightarrow 16 \rightarrow 12$ on BERT-large at table 4. When we directly train the model to predict from layer 12 and to match the prediction of the entire model (Direct Self-Teaching), the performance on the development set drops by a very large margin for the above two tasks (61.3 $\rightarrow$ 31.0 for CoLA and 72.6 $\rightarrow$ 53.4 for RTE). However, if we gradually distill from the entire model to layer 12, the gap between the distilled model and the whole model closes up.

4.5 Ablation on Distillation Schedule

We show the results of the MRPC task with different distillation schedules in Figure 4. We found that pruning one layer at a time during the end of the schedule preserves the model better. All of our gradual distillation schedules outperform simple self-teaching from layer 24 to layer 12.

We found that gradually decreasing the number of pruned layers (the orange line) performs slightly better than other schedules given a fixed model compression ratio: in this case, cutting the number of layers in half (24 $\rightarrow$ 12).

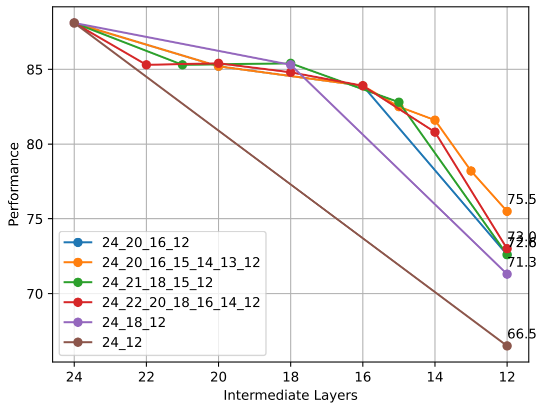


Figure 4: Results with Different Distillation Schedules on MRPC

4.6 Self-Teaching on Encoder-Decoder Models

Table 5 reports the results of distilling from T5 large using self-teaching. We only use the middle chunk (the last 12 layers of the encoder + the first 12 layers of the decoder) during inference after distillation. We were able to observe that even without a schedule of self-teaching, we can retain over 90% of the performance of the full model across two summarization tasks with different numbers of training examples. However, extracting the middle layers of a model can be expensive since it requires additional forward passes through the model, unlike self-teaching with the bottom few layers that only need one forward pass.

	CNN/DM (90,000)	Wikilingua-English (10,000)
Full 48 layers (1B)	42.5/20.7/39.8 (100%)	31.3/14.8/26.7 (100%)
12 Encoder, 12 Decoder (517M)	38.8/15.2/34.4 (91.3%)	28.5/9.6/26.2 (91.1%)

Table 5: Results of Self-Teaching on T5-Large on two conditional generation tasks. The number of training examples is shown next to the task. Cutting the number of layers in half still retains over 90% of the performance with self-teaching.

4.7 Task Agnostic Distillation

5 Related Works

Advanced Distillation Techniques. Knowledge Distillation was originally proposed for training a single model to match the performance of an ensemble of models [6]. Follow-up works apply KD to train smaller models to achieve faster inference [25, 21]. Another line of work called **self-distillation** uses KD as a performance-boosting technique under the scenario when the model architecture of the teacher and student are identical but with different initializations [5]. Follow-up work regularizes the output of the model when being fed a perturbed input from deviating too much from the output when being fed an original input, which shows large improvement in both vision [39, 40] and text [1]. Recent studies leverage dropout as a way to perturb the inputs and perform self-distillation [35, 9, 37]

The closest to our work is Zhang et al. [41], which penalizes the prediction of the final layer deviating too much from predictions from the intermediate layers in a convolutional neural network. Our work differs in that Zhang et al. [41] applies KD to boost the performance of the teacher model, while the aim of our work is to train a smaller model with minimal resource requirements.

Transformer Compression. Traditional KD matches the output distribution between the teacher model and the student model, which requires the logits to be probability distributions. Recent works also proposed to match the probability distribution of attention weights [30] and also minimizing the mean squared error between intermediate layers [24, 10] for more fine-grained distillation.

Combining Pruning and Distillation. Model pruning removes redundant parameters to learn a smaller but compact model. Recent studies combine model pruning and distillation to improve the performance of the student model [7, 8, 36, 34, 10]. The two main paradigms of combining are to prune and distill sequentially [7, 8] or simultaneously [36, 34, 10]. However, these methods require iterative estimation of parameter importance and additional memory to store the student model. Both requirements can be expensive when the teacher model is scaled to a larger size. In contrast, our method leverages the fact that the deeper layers of transformers are generally less important and distills within the same model to save time and memory.

6 Conclusion

The burdensome distillation process that either requires additional memory or iterative re-training hinders the development of smaller models that achieve similar performance to their large counterparts. In this work, we propose **self-teaching**: distilling within a model itself for more efficient distillation of large Transformer models [27].

References

[1] Armen Aghajanyan et al. "Better Fine-Tuning by Reducing Representational Collapse". In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=OQ08SN70M1V`.

[2] Jacob Devlin et al. "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`. URL: `https://aclanthology.org/N19-1423`.

[3] Claire Cardie Faisal Ladhak Esin Durmus and Kathleen McKeown. "WikiLingua: A New Benchmark Dataset for Multilingual Abstractive Summarization". In: *Findings of EMNLP, 2020*. 2020.

[4] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=rJl-b3RcF7`.

[5] Tommaso Furlanello et al. "Born Again Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1607–1616. URL: `https://proceedings.mlr.press/v80/furlanello18a.html`.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: `1503.02531 [stat.ML]`.

[7] Lu Hou et al. "DynaBERT: Dynamic BERT with Adaptive Width and Depth". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 9782–9793. URL: `https://proceedings.neurips.cc/paper_files/paper/2020/file/6f5216f8d89b086c18298e043bfe48ed-Paper.pdf`.

[8] François Lagunas et al. "Block Pruning For Faster Transformers". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10619–10629. DOI: `10.18653/v1/2021.emnlp-main.829`. URL: `https://aclanthology.org/2021.emnlp-main.829`.

[9] Hyoje Lee et al. *Self-Knowledge Distillation via Dropout*. 2022. arXiv: `2208.05642 [cs.CV]`.

[10] Chen Liang et al. "HomoDistil: Homotopic Task-Agnostic Distillation of Pre-trained Transformers". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: `https://openreview.net/forum?id=D7srTrGhAs`.

[11] Chen Liang et al. "No Parameters Left Behind: Sensitivity Guided Adaptive Learning Rate for Training Large Transformer Models". In: *International Conference on Learning Representations*. 2022. URL: `https://openreview.net/forum?id=cuvga_CiVND`.

[12] Chen Liang et al. "Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization". 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 6524–6538. DOI: 10.18653/v1/2021.acl-long.510. URL: https://aclanthology.org/2021.acl-long.510.

[13] Zhuang Liu et al. "Rethinking the Value of Network Pruning". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rJlnB3C5Ym.

[14] Ekdeep Singh Lubana and Robert Dick. "A Gradient Flow Framework For Analyzing Network Pruning". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=rumv7QmLUue.

[15] Paul Michel, Omer Levy, and Graham Neubig. "Are Sixteen Heads Really Better than One?" In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

[16] Pavlo Molchanov et al. "Importance Estimation for Neural Network Pruning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11256–11264. DOI: 10.1109/CVPR.2019.01152.

[17] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJGCiw5gl.

[18] Ramesh Nallapati et al. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. DOI: 10.18653/v1/K16-1028. URL: https://aclanthology.org/K16-1028.

[19] Myle Ott et al. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 48–53. DOI: 10.18653/v1/N19-4009. URL: https://aclanthology.org/N19-4009.

[20] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *J. Mach. Learn. Res.* 21.1 (Jan. 2020). ISSN: 1532-4435.

[21] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019*. 2019. arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[22] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *CoRR* abs/1910.01108 (2019). URL: http://arxiv.org/abs/1910.01108.

[23] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1073–1083. DOI: 10.18653/v1/P17-1099. URL: https://www.aclweb.org/anthology/P17-1099.

[24] Siqi Sun et al. "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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4323–4332. DOI: 10.18653/v1/D19-1441. URL: https://aclanthology.org/D19-1441.

[25] Raphael Tang et al. "Distilling Task-Specific Knowledge From BERT Into Simple Neural Networks". In: *ArXiv* abs/1903.12136 (2019). URL: http://arxiv.org/abs/1903.12136.

[26] L. Theis et al. "Faster gaze prediction with dense networks and Fisher pruning". arXiv:1801.05787. 2018. URL: https://arxiv.org/abs/1801.05787.

[27] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[28] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://aclanthology.org/W18-5446.

[29] Alex Wang et al. "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems". In: *arXiv preprint 1905.00537* (2019).

[30] Wenhui Wang et al. "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 5776–5788. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[31] Jason Wei et al. "Emergent Abilities of Large Language Models". In: *Transactions on Machine Learning Research* (2022). Survey Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=yzkSU5zdwD.

[32] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. URL: https://aclanthology.org/N18-1101.

[33] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[34] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. "Structured Pruning Learns Compact and Accurate Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1513–1528. DOI: 10.18653/v1/2022.acl-long.107. URL: https://aclanthology.org/2022.acl-long.107.

[35] xiaobo liang xiaobo et al. "R-Drop: Regularized Dropout for Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 10890–10905. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/5a66b9200f29ac3fa0ae244cc2a51b39-Paper.pdf.

[36] Dongkuan Xu et al. "Rethinking Network Pruning – under the Pre-train and Fine-tune Paradigm". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 2376–2382. DOI: 10.18653/v1/2021.naacl-main.188. URL: https://aclanthology.org/2021.naacl-main.188.

[37] Haoran Xu, Philipp Koehn, and Kenton Murray. "The Importance of Being Parameters: An Intra-Distillation Method for Serious Gains". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 170–183. URL: https://aclanthology.org/2022.emnlp-main.13.

[38] Runxin Xu et al. "S^4-Tuning: A Simple Cross-lingual Sub-network Tuning Method". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 530–537. DOI: 10.18653/v1/2022.acl-short.58. URL: https://aclanthology.org/2022.acl-short.58.

[39] Ting-Bing Xu and Cheng-Lin Liu. "Data-Distortion Guided Self-Distillation for Deep Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 5565–5572. DOI: 10.1609/aaai.v33i01.33015565. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4498.

[40] Sukmin Yun et al. "Regularizing Class-Wise Predictions via Self-Knowledge Distillation". In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[41] Linfeng Zhang et al. “Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 3712–3721. DOI: `10.1109/ICCV.2019.00381`.

[42] Qingru Zhang et al. “PLATON: Pruning Large Transformer Models with Upper Confidence Bound of Weight Importance”. In: *arXiv preprint arXiv:2206.12562* (2022).

A Appendix

A.1 Hyper-parameters

	RTE	MRPC	CoLA	SST-2	QNLI	QQP
Learning Rate	3e-6, 1e-5	1e-5, 2e-5	3e-6, 3e-6	3e-6, 3e-6	5e-6, 3e-6	3e-6, 3e-6
Max Length	512	512	256	512	256	256
Epochs	10					
Batch Size	16					

Table 6: Detailed Hyper-parameter configuration for BERT-base and BERT-large experiments

A.2 Importance scores of a Trained model

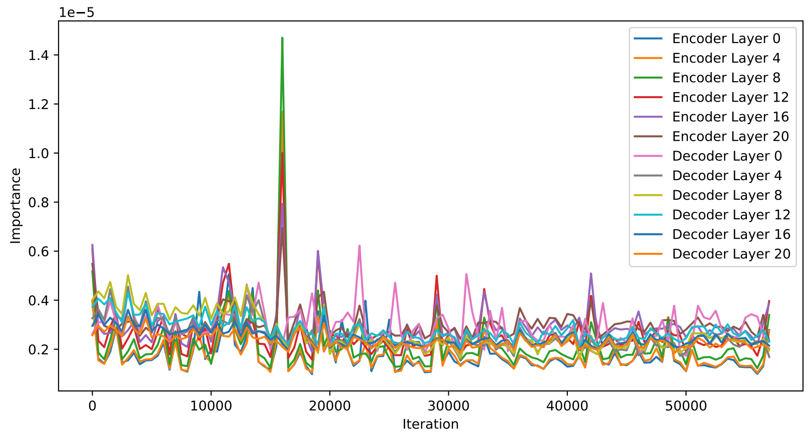


Figure 5: Importance of individual layers of T5-large (24 layer encoder + 24 layer decoder, 1.3B parameters)