Cost-effective Distillation of Large Language Models

Sayantan Dasgupta, Trevor Cohn[*] and **Timothy Baldwin**
School of Computing & Information Systems
University of Melbourne
sayandg@umich.edu, {trevor.cohn, tbaldwin}@unimelb.edu.au

Abstract

Knowledge distillation (KD) involves training a small "student" model to replicate the strong performance of a high-capacity "teacher" model, enabling efficient deployment in resource-constrained settings. Top-performing methods tend to be task- or architecture-specific and lack generalizability. Several existing approaches require pretraining of the teacher on task-specific datasets, which can be costly for large and unstable for small datasets. Here we propose an approach for improving KD through a novel distillation loss agnostic to the task and model architecture. We successfully apply our method to the distillation of the BERT-base and achieve highly competitive results from the distilled student across a range of GLUE tasks, especially for tasks with smaller datasets.[1]

1 Introduction

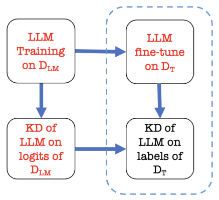
An unfortunate problem affecting large language models, such as BERT (Devlin et al., 2018) or GPT (Radford et al., 2019), is their high compute costs, as a consequence of their complex architectures and vast numbers of parameters. This is particularly apparent in initial (pre)training, but also impacts the cost of fine-tuning to specific tasks, and the practicality of their deployment on resource-constrained edge devices (Sun et al., 2020). *Knowledge distillation* (KD; Hinton et al. (2014)) attempts to mitigate these concerns through learning a small "student" model to replicate the behaviour of a larger, unwieldy "teacher". The idea is that much of the performance of the teacher can be captured by the student, despite it having many fewer parameters, and thereby better portability.

Several distillation methods have been proposed for large language models, including DistilBert (Sanh et al., 2019), which distills the 12-layer BERT transformer (Devlin et al., 2018) into a 6 layer student model with only a small loss in the performance on downstream tasks. Broadly, existing KD approaches are either architecture-specific or agnostic. The former group includes Jiao et al. (2020) and Sun et al. (2019a) which incorporate a loss term to encourage matching hidden between teacher and student, and thus requiring aligned teacher and student architectures. Approaches like Turc et al. (2019), on the other hand are architecture-agnostic, treating the teacher model as a black box using only the logits from language modelling heads for distillation it into a smaller LM. There are numerous advantages to the architecture-agnostic approach: (1) it is possible to distill a teacher model into a different student architecture, e.g. Tang et al. (2019) distills the BERT transformer into a simple single-layer Bi-LSTM; and (2) it frees the student to use different inference techniques, e.g., to better handle long sequences (Xiong et al., 2021; Vyas et al., 2020).

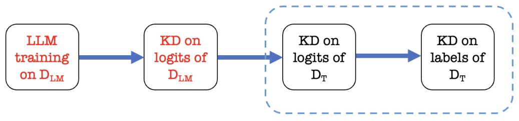
While the training of large language models incurs substantial compute resources – for instance the training cost of GPT3 (Brown et al., 2020) was estimated at $4.6 million using Nvidia Tesla V100 GPUs (Sharir et al., 2020). the cost of pretraining a given model is incurred only once. On the other hand, practitioners apply models to specific tasks, often involving fine-tuning of the LLM on their task-specific datasets, after which fine-tuned LLMs are then distilled into smaller LLMs for faster inference on real-time applications. This process incurs more modest compute costs, however, given the myriad of different applications, the process is repeated many times, meaning the aggregate cost can be significant, rivaling the cost of pre-training.[2] If we consider the per-instance training cost, fine-tuning is as costly as pre-training. Arguably this

[*]Now at Google DeepMind.
[1]Code available at `https://github.com/Sayan21/MAKD`

[2]Witness the explosion of BERT fine-tuning papers in the literature, and OpenAI's claim that GPT3 is being used in 300 applications: `https://openai.com/blog/gpt-3-apps`.



(a) Standard KD Approach



(b) Our Approach

Figure 1: Comparison of our KD approach against standard KD. The red font signifies a computationally intensive step. D_{LM} represents the large generic corpora such as Wiki or BookCorpus, whereas D_T represents the smaller task-specific corpora. The steps in the dotted box are performed by practitioners, whereas the rest are on-off performed by the authors of LLMs.

is less of an issue for small datasets, as the fine-tuning costs will be also be small, however in this setting fine-tuning can be unstable (Zhang et al., 2020) because there are not enough data points to reliably tune the parameters.

In this paper, we propose an architecture-agnostic approach for LLM distillation to eliminate the fine-tuning step. The standard KD for an LLM is shown in Figure 1(a), whereas our approach corresponds to Figure 1(b). The boxes with red-font stand for computationally expensive steps. The boxes in the dotted line are replicated by the practitioners and contribute to the major cost, whereas the boxes outside represent a one-off cost and can be ignored as such. We show the derivation of our approach along with its convergence properties, and then we describe our training strategy. We finally demonstrate the effectiveness of our approach based on distilling BERT models evaluated against the GLUE benchmark (Wang et al., 2018).

2 Methodology

We follow the empirical risk management framework for deriving our KD approach. For simplicity, we assume *temperature* $\tau = 1$ from the original definition in (Hinton et al., 2014). Let us assume that for a problem over a domain X, Y, the Bayesian optimal probabilities are $p_0(x) = \mathbf{E}[Y|X = x]$. Then the ideal KD loss is a loss between the student probabilities $f(X)$ and $p_0(X)$ is $l(f, p_0)$, and the optimal student is

$$f_0 = \arg\min_{f \in \mathcal{F}} \mathbf{E}_X [l(f(X), p_0(X))] \,, \tag{1}$$

note we use f, p and $f(x), p(x)$ interchangeably.

Given that we do not know p_0, the best we can do is to train a teacher from a function class $\mathcal{F}$ over some loss to find an estimate $\hat{p}$. We replace $p_0(X)$ in the loss by the empirical distribution, $\hat{p}(X)$, to arrive at the KD loss $\mathbf{E}_X[l(f(X), \hat{p}(X))]$. This is the KD loss defined over the entire population of X, Y. Given a training set $\mathcal{D}$ of n data points $\{x_i, y_i\}_{i=1}^n$, we can estimate it as

$$\mathbf{E}_{\mathcal{D}}[l(f(X), \hat{p}(X))] = \frac{1}{n} \sum_{i=1}^{N} l(f(x_i), \hat{p}(x_i)) \,.$$

This is the typical KD loss used in Hinton et al. (2014), also known as Vanilla KD. The loss $l(f, \hat{p})$ is usually Kullbach-Liebler (KL) divergence $D_{KL}(\hat{p} \| f)$ for $\tau = 1$, or the squared difference of the logits. If the student obtained from optimizing the KD loss is $\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} l(f(x_i), \hat{p}(x_i))$, then with high probability (Dao et al., 2020) it satisfies

$$\|\hat{f} - f_0\|_n^2 = O\left(\frac{1}{n} + \|\hat{p} - p_0\|_n^2 + \delta_n(\mathcal{F}, p_0)^2\right) \,, \tag{2}$$

where $\| \cdot \|$ stands for the L_2 norm of the difference between the parameters of the two classification functions. $\delta_n(\mathcal{F}, p_0)^2$ is the local Rademacher radius of the class of function $\mathcal{F}$, and is usually convex when $\mathcal{F}$ is the family of neural network or kernel functions (Dao et al., 2020). It is specific to the classification function class of the teacher and is a constant when the teacher is fixed. The student error $\|f - f_0\|_n$ thus depends on the second order teacher error $\|\hat{p} - p_0\|_n^2 = \frac{1}{n} \sum_{i=1}^{n} \|\hat{p}(x_i) - p_0(x_i)\|_2^2$.

2.1 Taylor Series Expansion of the Loss

Let us first start with a scalar $p \in [0, 1]$. If $\mathcal{L}(p)$ is a convex loss on p, then the following inequality holds (Böhning and Lindsay, 1988),

$$\mathcal{L}(p) \leq \mathcal{L}(\hat{p}) + (p - \hat{p}) \frac{d\mathcal{L}(p)}{dp}\bigg|_{p=\hat{p}} + \frac{1}{2}(p - \hat{p})^2 C \tag{3}$$

where $C = \arg\max_p \frac{d^2\mathcal{L}(p)}{dp^2}$ is the maximum curvature of the loss w.r.t. the entire domain of p. For example, for a binary cross entropy loss

Method	**Pre-training (D_{LM})**	**Task-specific (D_T)**	**Architecture-agnostic**
DistilBERT (Sanh et al., 2019)	BERT-base (truncated) + KD	Fine-tuning	No
Patient-KD (Sun et al., 2019a)	BERT-base (truncated)	Patient-KD	No
StudentBERT (Turc et al., 2019)	LM pretraining	Vanilla KD	Yes
TinyBERT$_4$ (Jiao et al., 2020)	KD with loss between attention matrices & hidden layers	KD with data augmentation w.r.t fine-tuned BERT-base	No
MobileBERT (Sun et al., 2020)	KD with layer transfer loss	Fine-tuning	No
Enhanced KD (ours)	LM pretraining	KD with Taylor series	Yes

Table 1: Detail of the two stages performed during KD under different approaches

$$\mathcal{L}(p) = -y\log(p) - (1-y)\log(1-p),$$

$$C = \arg\max_p \left(\frac{y}{p^2} + \frac{1-y}{(1-p)^2}\right). \qquad (4)$$

Observe that $C \to \infty$ as $p \to 0$ or $p \to 1$.

Now, when $p \in [0,1]^K$ is a vector of probabilities for K classes, we can extend the result to

$$\mathcal{L}(p) \leq \mathcal{L}(\hat{p}) + \left\langle p - \hat{p}, \left.\frac{d\mathcal{L}(p)}{dp}\right|_{p=\hat{p}} \right\rangle + \frac{1}{2}\|p-\hat{p}\|_2^2 C$$

with C now being the maximum value of the determinant of the Hessian, which is equivalent to the curvature of the loss. This is also similar to the inequalities for a β-smooth convex function (Bubeck et al., 2015, §3.2). However, the constant β is not really informative, unlike our case where we can connect C to the curvature of the loss,

$$C = \arg\max_p \det\left|\frac{d^2\mathcal{L}(p)}{dp^2}\right|. \qquad (5)$$

Coming back to KD, if we assume the teacher probabilities are $p \in [0,1]^K$ and the student probabilities are $f \in [0,1]^K$, then the vanilla KD loss is defined as $l(f,p)$. As long as $l(f,p)$ is convex w.r.t. to p, the following inequality holds,

$$l(f,p_0) \leq l(f,\hat{p}) + \langle p_0 - \hat{p}, \nabla_{\hat{p}} l(f,\hat{p})\rangle + \frac{1}{2}\|p_0 - \hat{p}\|_2^2 C(f)$$

Now we replace the derivatives with the partial derivatives as $\nabla_{\hat{p}} l(f,\hat{p}) = \left.\frac{\partial l(p)}{\partial p}\right|_{p=\hat{p}}$. The maximum curvature will be a function of the student probabilities f,

$$C(f) = \arg\max_p \det\left|\frac{\partial^2 l(f,p)}{\partial p^2}\right|. \qquad (6)$$

Recall that $l(f,p_0)$ is the ideal KD loss, as defined in Equation (1). Although we cannot estimate it, we can now obtain an upper bound on it and minimize this upper bound in our algorithm.

The most common KD loss used in the literature is the KL divergence between the student and the teacher probabilities $D_{KL}(\hat{p}\|f)$, when we keep $\tau = 1$. For KL divergence $l(f,\hat{p}) = \sum \hat{p}\log(\hat{p}/f)$ the first order derivative is,

$$\nabla_{\hat{p}} l(f,\hat{p}) = 1 + \log\hat{p} - \log f$$

and $C(f) = \arg\max_p \nabla_{\hat{p}}^2 l(f,\hat{p})$ will not contain any term involving f. This means we can exclude this term from KD. Removing the constant terms, the loss function becomes,

$$l(f,p_0) \leq l(f,\hat{p}) + \langle p_0 - \hat{p}, -\log(f)\rangle \qquad (7)$$

As we do not have knowledge of p_0, we cannot compute the loss directly. But we can take an unbiased estimate of p_0 as y from the training data $\mathcal{D}$, enabling the computation of the Taylor series term. As such, our KD loss is,

$$\begin{aligned}\mathcal{L}_{KD} &= \mathbf{E}_{x,y\sim\mathcal{D}}[l(f,\hat{p})] + \langle y - \hat{p}, -\log(f)\rangle \\ &\geq \mathbf{E}_{x,y\sim\mathcal{D}}[l(f,p_0)]\end{aligned} \qquad (8)$$

Following Mackey et al. (2018), an $O(n^{-1/(2k+2)})$ estimate of the teacher p with k Neyman orthogonal factors gives us an $O(1/\sqrt{n})$ estimation of the student f. For Vanilla KD (i.e. $k=0$), we see in Equation (2) that an $O(1/\sqrt{n})$ estimation of the student must have a $O(1/\sqrt{n})$ estimation of the teacher p, which is a more conservative requirement. The Taylor series term satisfies the condition of the first-order orthogonal term ($k=1$). That means now a $O(1/n^{1/4})$ estimate of teacher error $\|p - p_0\|_n$ is enough to give us an $O(1/\sqrt{n})$ bound of the student error $\|f - f_0\|_n$. $O(1/n^{1/4})$ is a weaker convergence guarantee than $O(1/\sqrt{n})$. This simply means now we can train a good student even from a weaker estimate of the teacher.

Finally, combining this with the explicit classification loss $\mathcal{L}_{class}$ for the student, the overall loss function for some $\lambda \in [0, 1]$ is

$$\mathcal{L} = \lambda\mathcal{L}_{class} + (1 - \lambda)\mathcal{L}_{KD} \qquad (9)$$

3 Training Strategy

Existing methods generally rely on a two-stage approach: (1) pre-train the student model on the entire or a truncated part of the same dataset as the teacher (D_{LM}), and (2) perform fine-tuning or KD on a task-specific dataset (D_T). This avoids the costly fine-tuning of BERT on task-specific datasets. For example, Turc et al. (2019) and Sun et al. (2019a) perform simple pretraining of the student model on D_{LM}, while Sanh et al. (2019) and Jiao et al. (2020) perform KD on D_{LM}. While Sanh et al. (2019) and Sun et al. (2020) only perform output layer fine-tuning on the task-specific dataset, others perform KD on D_T. The details of the different stages of training are summarized in Table 1.

To test our method, we choose to perform KD on BERT language models (D_{LM}) from Huggingface (Wolf et al., 2020) and perform KD using only the task-specific dataset D_T. We do not use a fine-tuned teacher on the task-specific dataset. Fine-tuning of BERT is not only expensive but may be unstable for small datasets (Zhang et al., 2020). While the teachers without fine-tuning will be weak, as described in Section 2.1, our proposed approach is designed to be robust to this.

4 Experiments

We use datasets from GLUE (Wang et al., 2018) for our experiments, specifically: SST-2 (Socher et al., 2013) for sentiment classification; MRPC (Dolan and Brockett, 2005), QQP, and STS-B for paraphrase similarity matching (Conneau and Kiela, 2018); and MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE (Wang et al., 2018) for natural language inference. We use KL divergence loss and the first Taylor series term (see Equation 8). For datasets with real-valued outputs, we can use Platt scaling (Platt et al., 1999) with a sigmoid function centered at the mean to convert it to a probability. For example, for STSB the output is a real number between 0 and 5, which we convert the target t into a probability via Platt Scaling $p = 1/(1 + \exp(-(t - 2.5)))$.

The teacher model is BERT-base (Devlin et al., 2018), with 109 million parameters across 12 layers, and 768d hidden states. We conduct experiments for three student models as listed in Table 2. We take our baseline results for Vanilla KD from the corresponding student model in Turc et al. (2019). We present results for our method based on: (a) a 4-layer student model, which we compare with the 4-layer TinyBERT model (Jiao et al., 2020) and MobileBERT (Sun et al., 2020);[3] and (b) a 6-layer student model, which we similarly compare against 6-layer TinyBert and DistilBERT (Sanh et al., 2019) models. We constrain all experiments to run on a single RTX-3090 GPU with 24GB RAM. The benchmark TinyBERT, MobileBERT, and Distilbert models were downloaded from the Huggingface repository (Wolf et al., 2020) and used without further modification. We present the results of 6-layer TinyBERT from Zhou et al. (2022).

The only hyper-parameter we optimize with our method is λ, in the range $[0, 1]$ at a step-size of 0.1, with a fixed temperature of $\tau = 1$ and learning rate of $\eta = 5 \times 10^{-5}$ (for the Adam optimizer).

In the results in Table 2, we register improvements in the GLUE metrics using the modified loss for all our student architectures against the baseline of Vanilla KD (Turc et al., 2019). Relative to the other KD methods, we get consistently better results for smaller datasets like MRPC, RTE, and STSB, but are slightly below the best KD models for the larger datasets, noting that these are all architecture-specific and rely on additional fine-tuning or data augmentation. The effect of dataset size follows from the theory in Equation (2), which shows that the teacher error typically follows the sample complexity $\|p_0 - \hat{p}\|_n \in O(n^{-1/(2k+2)})$, with $k = 0$ being the best case (Mackey et al., 2018). The difference between p_0 and $\hat{p}$ is large for smaller n, and this teacher error in turn reflects in the student error in Vanilla KD. I.e., our technique for expanding the loss makes a large difference for smaller n.

TinyBERT is overall the strongest performer for larger datasets ($>$ 10K samples) but achieves this using expensive task-specific fine-tuning and data augmentation. Data augmentation helps single-sentence tasks more than paired tasks because it is difficult to align the extra data in a pair according to

[3]MobileBERT uses a 6-layer architecture, but has similar #parameters as our 4-layer model.

Task # of Training Samples (in K)	# of P(M)	QQP 363.8	MNLI (m/mm) 392.7	SST-2 67.3	QNLI 104.7	MRPC 3.7	RTE 2.5	STSB 5.7
BERT base	109	87.9	84.6/84.9	93.0	91.2	90.4	71.4	89.8
Vanilla KD (2 x 128)	4	62.2	70.2/70.3	83.2	81.5	71.1	57.2	73.6
Our method (2 x 128)	4	**64.4**	**71.7/70.5**	**83.4**	**81.6**	**72.1**	**62.1**	**76.2**
Vanilla KD (4 x 312)	15	66.5	75.4/74.9	87.6	84.8	83.2	62.6	77.1
MobileBERT$_{\text{TINY}}$	15	68.9	81.5/81.6	91.7	**89.5**	87.9	65.1	80.1
TinyBERT$_4^\dagger$ (4 x 312)	15	**71.3**	**82.5/81.8**	**91.9**	87.7	86.4	66.6	80.4
Our method (4 x 312)	15	68.8	80.6/80.1	89.9	86.5	**88.1**	**66.7**	**82.2**
Vanilla KD (6 x 768)	66	70.7	82.8/82.2	91.0	88.9	86.8	65.3	81.0
DistilBERT (6 x 768)	66	70.1	82.6/81.3	92.5	88.9	86.9	58.4	81.3
TinyBERT$_6^\dagger$ (6 x 368)	66	**71.6**	**84.6/83.2**	**93.1**	**90.4**	87.3	66.8	83.7
Our method (6 x 768)	66	71.4	82.8/82.5	91.6	89.3	**89.0**	**67.5**	**84.0**

Table 2: Results for different student models on the GLUE test dataset, with result blocks grouping models of the same architecture and parameter base. The Vanilla KD results are of the corresponding student model from (Turc et al., 2019). The numbers in parenthesis are the number of layers and hidden states, respectively. The second column indicates the number of parameters (millions). The scores mentioned are F1 score for QQP & MRPC, Pearson's correlation for STSB, and accuracy for the rest of the datasets. † TinyBERT uses additional unlabelled data compared to the other methods, conferring an advantage.

the task. This is why TinyBert performs better than even BERT-base for SST2. We achieved the best results over tasks with small datasets, which is where task-specific KD is more difficult. The simplicity of our approach also makes it compatible with KD for more complex tasks like machine translation (Wang et al., 2021). A fairer comparison would be against the results of TinyBert without data augmentation, but those results were not reported in their publication.

5 Conclusion

We have proposed a general approach to improve KD on language models. We constrain the experiments on BERT mainly due lack of benchmarks on other LLMs as well as resource limitations. But any LLM distillation will show a similar trend. Existing KD methods are highly customized to the specifics of the teacher model, and require additional pre-training, fine-tuning, or data augmentation. Our approach is much simpler and agnostic to both architecture and task. We ran our experiments on an RTX3090 GPU with 24GB RAM which cost only $0.11 an hour, which is considerably cheap compared to other approaches that include teacher fine-tuning. We showed that our method is particularly effective on small datasets, and competitive with other KD methods which are much more computationally intensive and tailored to the teacher. A possible reason could be since the fine-tuning of BERT on small datasets like MRPC, STSB, or RTE can be unstable (Zhang et al., 2020), eliminating it makes the KD more robust and improves the results. All other methods such as TinyBert (Jiao et al., 2020) or PatientKD (Sun et al., 2019b) use fine-tuned teachers. DistilBert (Sanh et al., 2019) does not use a fine-tuned teacher, but it is only limited to students with a hidden state of 784 due to the cosine loss it uses and lacks generalization across architectures.

6 Ethical Issues

As we distill the knowledge from an existing model (here BERT-base), our approach does not introduce any extra ethical concerns during knowledge distillation. However, if a bias is already present in the teacher model, it might get transferred to the student model (Hooker et al., 2020). This is not specific to our algorithm but is a common risk for all types of knowledge distillation.

7 Limitations

A key limitation of our experiments is that we only consider English corpora. The exclusive use of English datasets is unlikely to have a substantive effect on distillation performance, and we would expect the results to transfer to other languages and datasets, however, languages with rich morphology may present modeling challenges arising from tokenization, that is, with many small word-pieces, language modeling (and its distillation) is likely to be a considerably harder task. As it stands, our work follows the standard evaluation protocols in peer benchmarks e.g., Jiao et al. (2020), Sanh et al. (2019), and Turc et al. (2019).

We only use BERT-base (Devlin et al., 2018) as our teacher model and benchmark against students that use it as a teacher model. For larger teacher models such as BERT-large or GPT2 (Radford et al., 2019), the inference time as well as memory requirement would be much higher, and would necessitate larger GPU clusters. This is a consequence of the cost of the forward pass with the teacher model, rather than our distillation algorithm, which has a much lighter footprint. We argue that the result from one transformer-based pre-trained language model should generalize well to other transformer-based pre-trained models. Thus our results are representative, despite our smaller-scale evaluation protocol.

Another shortcoming of transformer models, in general, is their scalability to long text. In this setting, model-agnostic knowledge distillation, like our technique, enjoys a distinctive advantage. We can incorporate techniques like Beltagy et al. (2020) or Xiong et al. (2021) to speed up attention in the student model enabling it to scale to long texts, even when paired with a different architecture for the teacher. Jiao et al. (2020) and Sanh et al. (2019) rely on specific model internals during distillation, and therefore the student model has to be similar to the teacher.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Dankmar Böhning and Bruce G Lindsay. 1988. Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck et al. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tri Dao, Govinda M Kamath, Vasilis Syrgkanis, and Lester Mackey. 2020. Knowledge distillation as semiparametric inference. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*.

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*.

Lester Mackey, Vasilis Syrgkanis, and Ilias Zadik. 2018. Orthogonal machine learning: Power and limitations. In *International Conference on Machine Learning*, pages 3375–3383. PMLR.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. 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, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019b. Patient knowledge distillation for bert model compression. arXiv arXiv:1908.09355.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. arXiv preprint arXiv:1908.08962.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Fusheng Wang, Jianhao Yan, Fandong Meng, and Jie Zhou. 2021. Selective knowledge distillation for neural machine translation. 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 6456–6466, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14138–14148.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. *arXiv preprint arXiv:2006.05987*.

Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. BERT learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7037–7049, Dublin, Ireland. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 6 (Limitations)

☑ A2. Did you discuss any potential risks of your work?
Section 7 (Ethics)

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3, 4

☑ B1. Did you cite the creators of artifacts you used?
Section 3, 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We only used open-source artifacts. We will open-source our own code and models too using MIT license.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Our distilled encoder-only models for general-purpose NLP do not violate the intended use of the artifacts.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. We only used standard anonymized datasets.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 3, 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Only the single-run result is provided because the experiments are too computationally intensive. It will waste energy and cause unnecessary CO2 emissions.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3, 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.