Towards Automated Distillation: A Systematic Study of Knowledge Distillation in Natural Language Processing

Haoyu He[1] † Xingjian Shi[2] Jonas Mueller[2] Sheng Zha[2] Mu Li[2] George Karypis[2]

[1]Northeastern University [2]Amazon Web Services

Abstract Key factors underpinning the optimal Knowledge Distillation (KD) performance remain elusive as the effects of these factors are often confounded in sophisticated distillation algorithms. This poses a challenge for choosing the best distillation algorithm from the large design space for existing and new tasks alike and hinders automated distillation. In this work, we aim to identify how the distillation performance across different tasks is affected by the components in the KD pipeline, such as the data augmentation policy, the loss function, and the intermediate knowledge transfer between the teacher and the student. To isolate their effects, we propose *Distiller*, a meta-KD framework that systematically combines the key distillation techniques as components across different stages of the KD pipeline. Distiller enables us to quantify each component's contribution and conduct experimental studies to derive insights about distillation performance: 1) the approach used to distill the intermediate representations is the most important factor in KD performance, 2) the best-performed distillation algorithms are quite different across various tasks, and 3) data augmentation provides a large boost for small training datasets or small student networks. Based on these insights, we propose a simple *AutoDistiller* algorithm that can recommend a close-to-optimal KD pipeline for a new dataset/task. This is the first step toward automated KD that can save engineering costs and democratize practical KD applications.

1 Introduction

To reduce the inference cost while preserving most of the accuracy of prevalent large pretrained models used in natural language processing (NLP), *task-aware* KD is a popular and particularly promising approach for supervised learning tasks. The idea is to first fine-tune an accurate and large *teacher* model on the labeled data, and then train a separate *student* model that has much fewer parameters to mimic the predictions of the teacher. Innovations in KD for NLP generally improve the following aspects: 1) the loss function for gauging the discrepancy between student and teacher predictions, 2) the method for transferring intermediate network representations between teacher and student, 3) the use of data augmentation during student training, and 4) multiple stages of distillation. Many studies have simultaneously introduced new variations of more than one of these components, which confounds the impact of each component on the final performance of the distillation algorithm. In addition, it is often unclear whether the proposed KD algorithms retain their advantageous performance across different tasks besides those they are evaluated on. As a result, selecting the solution from the large combined design space of KD algorithms for a new application is increasingly challenging. This is a major obstacle for *automated KD*, of which the goal is to recommend a good KD pipeline for a new dataset.

To understand the importance of different components in KD and lay the cornerstone for automated KD, we conduct a systematic study of the design space of KD algorithms in NLP. We compose several key configurable components in KD into a meta-distillation pipeline, called *Distiller*. All candidate algorithms in the search space of *Distiller* work for two main types of NLP tasks: text classification and sentence tagging. We run extensive hyper-parameter tuning algorithms to search

[†] Work done while being an intern at Amazon Web Services.

for the best *Distiller*-configuration choices over GLUE [22] and SQuAD [15]. The hyper-parameter search helps us understand what impact different KD components have on student performance and motivates us to propose *AutoDistiller* as the very first step towards automated KD that can save engineering cost and democratize machine learning. *AutoDistiller* is designed to predict distillation performance based on KD pipeline choices and characteristics of tasks. We expect our proposed AutoDistiller, as a baseline of automated KD, to bring fresh insights to the rapidly growing automated machine learning (AutoML) community. Experimental analysis of *AutoDistiller* shows that it is potential to reliably prioritize high-performing KD configurations, and can suggest good distillation pipelines on two new datasets. Main contributions of this work are:

- A systematic study on the meta-KD pipeline *Distiller* to assess the impact of different components in KD, including the: 1) data augmentation policy, 2) loss function for transferring intermediate representations, 3) layer mapping strategies for intermediate representations, 4) loss function for transferring outputs, as well as what role the task and dataset type play.
- Using experimental results collected from our *Distiller* study and features extracted from downstream datasets, we fit a model that automatically predicts the best distillation strategy for new datasets. On hold-out datasets "BoolQ" [21] and "cloth", predicted strategies achieve 0.99 and 0.14 distillation ratios (fraction of the student's and teacher's performance enhancement gained from distillation) on average, outperforming randomly selected strategies with mean of -0.98 and -0.24. To the best of our knowledge, this is the first attempt toward automated KD in NLP, providing a baseline solution to future automated KD study.

2 Related Work

Knowledge Distillation. In the domain of NLP, recent KD literature discussed how to transfer knowledge from pretrained Transformer-based models efficiently. [17] proposed BERT-PKD that transfers the knowledge from both the final layer and intermediate layers of the teacher network. [9] proposed TinyBERT that first distills the general knowledge of the teacher by minimizing the Masked Language Model (MLM) objective [2], followed by task-specific distillation. [11] proposed a many-to-many layer mapping function leveraging the Earth Mover's Distance to transfer intermediate knowledge. [7] proposed DynaBERT, a multistage distillation pipeline to distill the teacher Transformer to student Transformer with different widths and depths. Focusing on AutoML settings with tabular data, [4] proposed a general KD algorithm for different classical ML models and ensembles thereof. [10] compared Kullback-Leibler divergence and mean squared error objectives in KD for image classification models, finding that the mean squared error performs better. [24] introduced an open-source KD toolkit that supports various KD techniques. Our work provides a systematic analysis of the different components in single-stage task-aware KD algorithms in NLP and propose the first automated KD algorithm in this area.

3 Methodology

Our study is structured around a configurable meta-distillation pipeline called *Distiller*[*] (details in Appendix A). It contains four configurable components: the data augmentation policy $a(\cdot,\cdot)$, the layer mapping configuration of intermediate distillation $\{m_{i,j}\}$, the intermediate distillation objective $l^{\text{inter}}(\cdot,\cdot)$, and the prediction layer distillation objective $l^{\text{pred}}(\cdot,\cdot)$. Assume the teacher network f^T has M layers and the student network f^S has N layers, for a given data/label pair (x, y) sampled from the dataset $\mathcal{D}$, the student learns from the teacher by minimizing loss:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{\hat{x},\hat{y}\sim a(x,y);x,y\sim\mathcal{D}} \sum_{i=1}^{M}\sum_{j=1}^{N} m_{i,j} l_{i,j}^{\text{inter}}(H_i^{\mathrm{T}}, H_j^{\mathrm{S}}) + \beta_1 l^{\text{pred}}(f^T(x), f^S(x)) + \beta_2 l^{\text{pred}}(f^T(\hat{x}), f^S(\hat{x})) \\ + \gamma_1 l^{\text{pred}}(y, f^S(x)) + \gamma_2 l^{\text{pred}}(\hat{y}, f^S(\hat{x})).\end{aligned} \tag{1}$$

[*] The complete training and testing code are available at https://github.com/Cli212/Distiller.

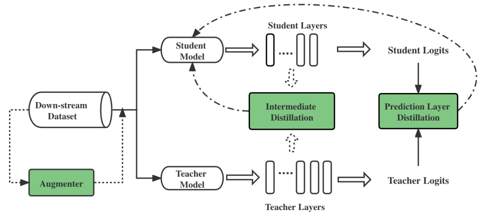


Figure 1: Overview of the *Distiller* pipeline. All configurable components are colored.

Here, $m_{i,j} \in [0,1]$ represents the layer mapping weight between the i-th teacher layer and j-th student layer, $H_i^{\mathrm{T}}, H_j^{\mathrm{S}}$ are the i-th and the j-th layer of hidden representations of the teacher and the student, β_1, β_2 control the strength of distilling from class probabilities produced by the teacher, and γ_1, γ_2 control the strength of learning from ground truth data (x, y) and synthesized data $(\hat{x}, \hat{y})$. In Appendix C.2, we illustrate how previous model distillation algorithms [9, 11, 12] can be encompassed in the Distiller framework.

3.1 AutoDistiller

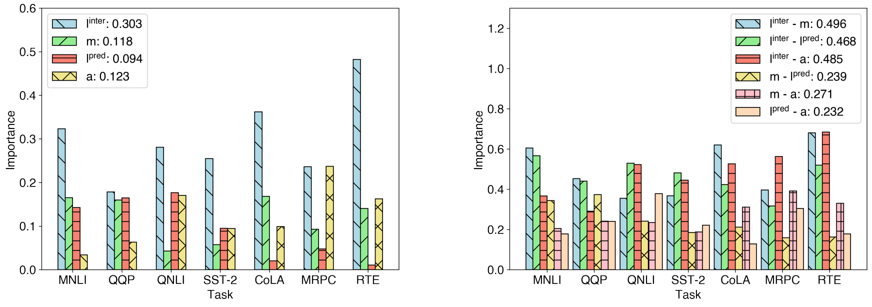
To enable automated distillation, we fit a prediction model that recommends a good KD pipeline based on the features extracted from the dataset and Distiller search space. To encode semantic information of the dataset into features, the context and dataset descriptions are further represented by aggregated GloVe [13] embeddings of words weighted by TF-IDF. We also extract numerical features such as the finetuned baseline score and the finetuned teacher score to measure how tough the task is in need of a complex network, and the number of samples in the dataset. Details about dataset feature extraction are described in Table 2 in Appendix. To effectively measure the performance enhancement gained by distillation over finetuning, *AutoDistiller* is trained to predict the **distillation ratio**:

$$r = \frac{S_{\text{distill}} - S_{\text{finetune}}}{T_{\text{finetune}} - S_{\text{finetune}}}, \tag{2}$$

where S_{distill} is the evaluation score of the distilled student, $T_{\text{finetune}}, S_{\text{finetune}}$ are evaluation scores of finetuned student and teacher. This allows us to clearly represent how much performance improvement the student gains from distillation rather than finetuning from scratch. Besides, this ratio can be used across different tasks and different teacher/student architectures so we can continuously update *AutoDistiller* by feeding more distillation results in the future. Once trained on features extracted from datasets as well as features of each candidate distillation configuration, *AutoDistiller* recommends distillation pipelines that maximize the predicted distillation ratio given any downstream dataset/task. To the best of our knowledge, *AutoDistiller* is the first attempt toward automated KD in NLP.

4 Experiments

Under the previously described experimental setup, we conduct experiments and collect more than 1300 sets of data points that include *Distiller* configuration, the dataset/task, and distillation results. Analyzing the data reveals three major findings: 1) design of the intermediate distillation module is the most important among all factors studied, 2) DA provides a large boost when the dataset or the student model is small, and 3) the best distillation policy varies among datasets. Detailed experimental results revealed these three findings can be found in Appendix C.5. Inspired by these findings, we train a meta-learning model *AutoDistiller* that is able to recommend a good distillation policy on a new dataset based on which configurations tended to work well for alike datasets applied in our study.



(a)

(b)

Figure 2: As assessed via fANOVA, we report the individual importance of the four Distiller components in (a) and importance of interactions between any two of the four components in (b). Four components are: l^{inter} for intermediate distillation objective, l^{pred} for prediction layer distillation objective, a for data augmentation and m for layer mapping strategy. Average importance for each component (across tasks) is listed in the legend.

4.1 Importance of Components

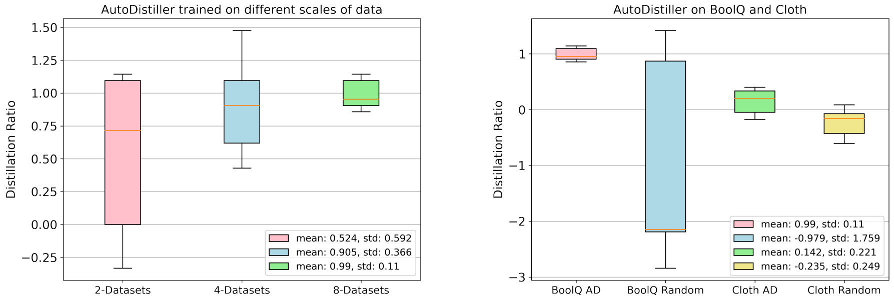
To study the importance of each component described in the previous section, we randomly sample *Distiller* configurations in the designed search space while controlling the optimizer and other unrelated hyper-parameters. We apply each sampled distillation configuration on a diverse set of NLP tasks and different teacher/student architectures. To analyze the importance of different components in *Distiller*, we adopt fANOVA [8], an algorithm for quantifying the importance of individual hyper-parameters as well as their interactions in determining downstream performance. We use fANOVA to evaluate the importance of the four components in *Distiller* as well as their pairwise combinations: data augmentation, intermediate distillation objective, layer mapping strategy, and prediction layer distillation objective. We report the results in Figure 2, which illustrates that the objective function for intermediate distillation l^{pred} has the highest individual importance, and the combination of the intermediate distillation objective and layer mapping strategy has the highest joint importance. One hypothetical explanation is that the teacher can provide token-level supervision to the student via intermediate distillation, which can better guide the learning process of the student. Therefore, one should most critically focus on these two components when selecting or designing a particular KD pipeline.

4.2 Performance of AutoDistiller

Recall that in Section 3.1, we construct the performance prediction model *AutoDistiller* on the extracted dataset features and previously collected experimental results (over 1300 pieces). Here we split all experimental results in a (80/20) train/test ratio then train and evaluate *AutoDistiller* via AutoGluon-Tabular [3], a simple AutoML tool for supervised learning.

As we construct dataset features from scratch, it is essential to evaluate how much these features contribute to the final performance of *AutoDistiller*. Therefore, we compute permutation feature importance [1], which is defined as the drop of prediction accuracy after the values of a particular feature are shuffled in the test data. From the results in Figure 4 in Appendix, we observe that all features pertain positive importance, among which the task and context embeddings are the two features with the highest feature importance. This shows that the dataset domain and problem type are important factors to consider when constructing AutoDistiller features.

Given that the objective of *AutoDistiller* is to recommend near optimal distillation configurations for new datasets, we applied *AutoDistiller* on two datasets "BoolQ" [21] and "cloth" [16] that are not considered in our previous experiments to evaluate its adaptability. In Section 3.1, we discussed that



(a) AD trained on results of 2, 4, 8 tasks. The scaling of AD training data avoid producing trivial strategies where distillation is worse than finetuning.

(b) Top-5 AD strategies vs. 5 randomly selected strategies. AD significantly outperforms random search on both tasks.

Figure 3: Distillation ratio of *AutoDistiller* (AD) recommended strategies given teacher $\text{BERT}_{\text{BASE}}$ and student TinyBERT_4 under two settings. Higher ratio indicates better distillation performance. Mean and standard deviation of the groups of ratios are listed in the legend.

AutoDistiller can be continuously updated with more distillation results due to the carefully designed scheme. Here we conducted an ablation experiment to verify the necessity of having more datasets for training *AutoDistiller*. We gradually increase the number of datasets for training *AutoDistiller* and compare the performance of the students trained under the recommended distillation configurations. Results in Figure 3a demonstrate that scaling of AD training data produces more stable distillation strategies. In addition, we evaluate the effectiveness of *AutoDistiller* by comparing the distillation ratios obtained by the top-N strategies suggested by *AutoDistiller* with the distillation ratios from N randomly sampled strategies. Results are shown in Figure 3b. We observe that both groups of randomly selected strategies experience high variance in performance and may even be worse than a finetuned student from scratch (distillation ratio < 0). Randomly combining techniques in different components to form a KD strategy performs as a lottery ticket, which is inefficient in practical AutoML cases. We see great potential that a tool as AutoDistiller can solve this issue as a result of the stable and effective KD strategies it recommends.

5 Discussion

Large amount of KD algorithms are under a tendency to optimize separate components in the entire KD pipeline, complicating the search space of distillation techniques that are applied on different benchmarks, which fails to match industry demand to automatically select the best distillation strategy for heterogeneous downstream tasks. Our work makes efforts to approach this by proposing an automated distillation system *AutoDistiller* and experimental results reveal the promising vision of a stable and evolvable automated KD system.

Since automated distillation is fairly a new concept, this work aims to put forth a *proof-of-concept* and seeks to answer research questions instead of making a '*sotaeesque*' comparison. We leave extensive comparisons on other setups and baselines to future work. Because *AutoDistiller* serves as a baseline solution to automated KD, it has several limitations: 1) *AutoDistiller* lacks of training data because the 1300+ data points we collect from our experimental results are not a formally constructed dataset but products of meta-learning, and 2) *AutoDistiller* is compared only to randomly selected strategies, which is a low bar given the complex KD pipeline. We hope that as the *AutoDistiller* shows its potential, automated KD will become a research area that bridges academic research and industry demands.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[3] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[4] Rasool Fakoor, Jonas W Mueller, Nick Erickson, Pratik Chaudhari, and Alexander J Smola. Fast, accurate, and simple models for tabular data via augmented distillation. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[5] Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

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

[7] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. DynaBERT: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*, 2020.

[8] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *ICML*, pages 754–762. PMLR, 2014.

[9] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *EMNLP*, 2020.

[10] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. In *IJCAI*, 2021.

[11] Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. BERT-EMD: Many-to-many layer mapping for bert compression with earth mover's distance. In *EMNLP*, 2020.

[12] Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. MixKD: Towards efficient distillation of large-scale language models. In *ICLR*, 2021.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[14] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *ICML*, 2019.

[15] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[16] Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alexander J Smola. Benchmarking multimodal automl for tabular data with text fields. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[17] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *EMNLP*, 2019.

[18] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *ICLR*, 2020.

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

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[21] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

[22] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[23] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP*, 2019.

[24] Ziqing Yang, Yiming Cui, Zhipeng Chen, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–16. Association for Computational Linguistics, 2020.

Appendix. – Towards Automated Distillation: A Systematic Study of Knowledge Distillation in Natural Language Processing

A Distiller

A.1 Data Augmentation Policy

When the amount of labeled data is small, data scarcity becomes a key challenge for training students. This can be mitigated via Data Augmentation (DA) by generating additional data samples. Unlike in supervised learning, where labels for synthetic augmented data may be unclear unless the augmentation is limited to truly benign perturbations, the soft labels for augmented data in KD are simply provided by the teacher which allows for more aggressive augmentation [4]. Denote the set of training samples of the downstream task as $\mathcal{D}_{\text{train}}$, any augmenter $a(\cdot,\cdot)$ stretches the distribution from $E_{x,y\sim\mathcal{D}_{\text{train}}}$ to $E_{\hat{x},\hat{y}\sim a(x,y),x,y\sim\mathcal{D}_{\text{train}}}$. We consider various elementary DA operations including: 1) MLM-based contextual augmentation (CA) , 2) random augmentation (RA), 3) backtranslation (BT) and 4) mixup. The search space of possible augmentations in *Distiller* is constructed by stacking these four elementary operations in an arbitrary order, as detailed in Algorithm 1.

Mixup constructs a synthetic training example via the weighted average of two samples (including the labels) drawn at random from the training data. To use it in NLP, [5, 12] applied mixup on the word embeddings at each sentence position $x_{i,t}$ with $\lambda \in [0,1]$ as the mixing-ratio for a particular pair of examples x_i, x_j:

$$\hat{x}_{i,t} = \lambda x_{i,t} + (1-\lambda)x_{j,t},\ \hat{y}_i = \lambda y_i + (1-\lambda)y_j, \tag{3}$$

Here λ is typically randomly drawn from a *Uniform* or *Beta* distribution for each pair, y_i, y_j are labels in one-hot vector format, and $(\hat{x}_i, \hat{y}_i)$ denotes the new augmented sample. To further extend mixup for sentence tagging tasks, in which each token has its own label, we propose calculating the weighted combination of the ground-truth target at each location t as the new target:

$$\hat{x}_{i,t} = \lambda x_{i,t} + (1-\lambda)x_{j,t},\ \hat{y}_{i,t} = \lambda y_{i,t} + (1-\lambda)y_{j,t}, \tag{4}$$

Fundamental settings of the other three DA operations can be found in Appendix.

A.2 Prediction Layer Distillation

In traditional KD, the student network learns from the output logits of the teacher network, adopting these as soft labels for the student's training data [6]. Here we penalize the discrepancy between the outputs of student vs. teacher via:

$$\mathcal{L}_{\text{pred}} = l^{\text{pred}}(f^T(x), f^S(x)), \tag{5}$$

where $l^{\text{pred}}(\cdot,\cdot)$ is the KD loss component whose search space in this work includes either: softmax Cross-Entropy (CE) or Mean Squared Error (MSE).

Algorithm 1: Data Augmentation Policy

```
Params: A sequence of elementary data augmentation operations $\mathcal{G}$, $\forall \mathcal{G}_j \in \{CA, RA, BT, Mixup\}$.
Input: Training Dataset $\mathcal{D}_{\text{train}}$
Output: Augmented dataset $\mathcal{D}_{\text{synthesize}}$

Initialize $\mathcal{D}_{\text{synthesize}} \leftarrow \{\}$
foreach $\{x_i, y_i\} \in \mathcal{D}_{train}$ do
    for $j \leftarrow$ 1 to len($\mathcal{G}$) do
        $\hat{x}_i, \hat{y}_i = \mathcal{G}_j(x_i, y_i)$;
        $x_i, y_i \leftarrow \hat{x}_i, \hat{y}_i$;
    end
    $\mathcal{D}_{\text{synthesize}} \leftarrow \mathcal{D}_{\text{synthesize}} \cup \{x_i, y_i\}$
end
```

A.3 Intermediate Representation Distillation

To ensure the knowledge is sufficiently transferred, we can allow the student to learn from the intermediate layers of the teacher rather than only the latter's output predictions by minimizing the discrepancies between selected layers from the teacher and the student. These high-dimensional intermediate layer representations constitute a much richer information-dense signal than is available in the low-dimensional predictions from the output layer. [17] shows that this intermediate distillation scheme enables the student to *patiently* learn the rich information in the teacher's hidden layers. As teacher and student usually have different number of layers and hidden-state dimensionalities, it is not clear how to map teacher layers to student layers and how to measure the discrepancy between their hidden states. Previous works proposed various discrepancy measures for intermediate distillation, including: Cross-Entropy (CE), Mean Squared Error (MSE), L2 distance, Cosine Similarity (CS), and Patient Knowledge Distillation (PKD) [17]. For these objectives, we establish the following result. Proof is in Appendix B.

Theorem 1. *Minimizing MSE, L2, or PKD loss, and maximizing CS between two random variables X, Y are equivalent to maximizing particular lower bounds of the mutual information $I(X;Y)$.*

In our KD setting, X and Y correspond to the hidden state representations of the student and teacher model (for random training examples), respectively. Inspired by this theorem, it is reasonable to use the bounds of MI as intermediate objective functions in KD. Particularly, we consider the multisample MI lower bound of [14], which estimates $I(X;Y)$ given the sample x, y from $p(x, y)$ and another K additional IID samples $z_{1:K}$ that are drawn from a distribution independent from X, Y:

$$I(X;Y) \geq E_{p(x,z_{1:K})p(y|x)}\left[\log \frac{e^{f(x,y)}}{\alpha m(y;x,z_{1:K}) + (1-\alpha)q(y)}\right] - E_{p(x,z_{1:K})p(y)}\left[\log \frac{e^{f(x,y)}}{\alpha m(y;x,z_{1:K}) + (1-\alpha)q(y)}\right] + 1 \triangleq I_\alpha. \tag{6}$$

In I_α, $f(\cdot,\cdot)$ and $q(\cdot)$ are critic functions for approximating unknown densities and $m(\cdot,\cdot)$ is a Monte-Carlo estimate of the partition function that appears in MI calculations. Typically, the space z and the sample x, y are from the same mini-batch while training, namely the mini-batch size is $K + 1$. $I_\alpha \in [0, 1]$ flexibly trade off bias and variance, since increasing α reduces the variance of the estimator while increasing its bias. We propose to use I_α as an objective for intermediate distillation and call it MI-α. Our implementation leverages a Transformer encoder [20] to learn $f(\cdot,\cdot)$ and $q(\cdot)$. To our knowledge, this is the first attempt to utilize complex neural network architectures for critic functions in MI estimation; typically only shallow multi-layer perceptrons (MLPs) are used [18]. Our results in Table 6 reveal that Transformer produces a better critic function than MLP.

Note that for intermediate distillation, objectives like MSE attempt to ensure the teacher and student representations take matching values, whereas objectives like MI (and tighter bounds thereof) merely attempt to ensure the information in the teacher representation is also captured in the student representation. The latter aim is conceptually better suited for KD, particularly in settings where the student's architecture differs from the teacher, in which case forcing intermediate student representations to take the same values as teacher representations may even be harmful for tiny student networks that lack the capacity to learn the same function composition used by the teacher. Besides, MSE in theory measures two variables with matched size, while MI-α is more flexible in variable size due to the use of neural critic functions. This feature of MI-α makes it particularly suitable for KD, where the intermediate representations of the student and the teacher may differ in dimensions. We emphasize that a high MI between student and teacher representations suffices for the teacher's prediction to be approximately recovered from the student's intermediate representation (assuming the teacher uses deterministic output layers as is standard in today's NLP models). Given that high MI suffices for the student to match the teacher, we expect tighter MI bounds like MI-α can outperform looser bounds like MSE that impose additional requirements on the student's intermediate representations beyond just their information content.

A.3.1 **Layer Mapping Strategy**. We investigate three intermediate layer mapping strategies: 1) Skip: the student learns from every $\lfloor M/N \rfloor$ layer of the teacher, i.e., $m_{i,j} = 1$ when $j = i \times \lfloor M/N \rfloor$; 2) Last: the student learns from the last k layers of the teacher, i.e., $m_{i,j} = 1$ when $j = i + M - N$; and 3) EMD: a many-to-many learned layer mapping strategy [11] based on Earth Mover's Distance. The intermediate loss with EMD mapping can be denoted as:

$$\mathcal{L}_{\text{EMD}}(H^S_{1:N}, H^T_{1:M}) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} w^H_{i,j} d^H_{i,j}}{\sum_{i=1}^{M}\sum_{j=1}^{N} w^H_{i,j}}, \tag{7}$$

where $D^H = [d^H_{i,j}]$ is a distance matrix representing the cost of transferring the hidden states knowledge from H^T to H^S. And $W^H = [w^H_{i,j}]$ is the mapping flow matrix which is learned by minimizing the cumulative cost required to transfer knowledge from H^T to H^S. In *Distiller*, the distance matrix is calculated via intermediate objective function: $d^H_{i,j} = l^{\text{inter}}(H^S_i, H^T_j)$.

B Proof of Theorem 1

Denote the Mutual Information (MI) between two random variables X and Y as $I(X;Y)$. Based on the results on variational bounds of MI [14], we derive a theorem that optimizing common knowledge distillation objectives, including Mean Squared Error (MSE), L2 distance, and cosine similarity between X and Y, can be viewed as maximizing certain lower bounds of $I(X;Y)$. To prove the theroem, we leverage this lemma:

Lemma 1 (I_{TUBA}). *Assume that $f(x,y)$ is an arbitrary neural network that takes x and y as inputs and outputs a scalar and $a(y) > 0$. The lower bound of x and y can be estimated by:*

$$I(X;Y) \geq E_{p(x,y)}[\log \frac{e^{f(x,y)}}{a(y)}] - E_{p(x)p(y)}[\frac{e^{f(x,y)}}{a(y)}] \triangleq I_{TUBA}.$$

Proof. Based on the definition of MI, we have:

$$I(X;Y) = E_{p(x,y)}[\log \frac{p(x|y)}{p(x)}] = E_{p(x,y)}[\log \frac{p(y|x)}{p(y)}].$$

Replacing the intractable conditional distribution $p(x|y)$ with a tractable variational distribution $q(x|y)$ yields a lower bound on MI due to the non-negativity of the KL divergence:

$$\begin{aligned} I(X;Y) &= E_{p(x,y)}[\log \frac{q(x|y)}{p(x)}] + E_{p(x,y)}[\log \frac{p(x|y)}{q(x|y)}] \\ &= E_{p(x,y)}[\log \frac{q(x|y)}{p(x)}] + E_{p(y)}[KL(p(x|y)||q(x|y))] \\ &\geq E_{p(x,y)}[\log q(x|y)] + H(X), \end{aligned}$$

where $H(X)$ is the entropy of X. Then we choose an energy-based variational family that uses a critic function $f(x,y)$ and is scaled by the data density $p(x)$ to represent $q(x|y)$:

$$q(x|y) = \frac{p(x)}{Z(y)} e^{f(x,y)}, Z(y) = E_{p(x)}[e^{f(x,y)}].$$

Substituting this distribution into (B) gives a lower bound on MI:

$$I(X;Y) \geq E_{p(x,y)}[f(x,y)] - E_{p(y)}[\log Z(y)].$$

However, this objective is still intractable. To form a tractable bound, we can upper bound the log partition function by this inequality: $\log(x) \leq \frac{x}{a} + \log(a) - 1$ for all $x, a > 0$. Apply this inequality to get:

$$\begin{aligned}
I(X;Y) &\geq E_{p(x,y)}[f(x,y)] - E_{p(y)}[\log Z(y)] \\
&\geq E_{p(x,y)}[f(x,y)] - E_{p(y)}[\frac{E_{p(x)}[e^{f(x,y)}]}{a(y)} + \log a(y) - 1] \\
&= E_{p(x,y)}[f(x,y)] - E_{p(x,y)}[\log a(y)] \\
&- E_{p(x)p(y)}[\frac{e^{f(x,y)}}{a(y)}] + 1 \\
&\geq E_{p(x,y)}[\log \frac{e^{f(x,y)}}{a(y)}] - E_{p(x)p(y)}[\frac{e^{f(x,y)}}{a(y)}].
\end{aligned}$$

This bound holds for any $a(y) > 0$.

Theorem 1. *Minimizing MSE, L2, or PKD loss, and maximizing the cosine similarity between two random variables X, Y are equivalent to maximizing particular lower bounds of the mutual information $I(X;Y)$. In knowledge distillation, samples $x \in X$ and $y \in Y$ are hidden states generated by the student model and teacher model.*

Proof. We prove this theorem by constructing $f(x,y)$ and $a(y)$ in Lemma 1 for each loss function.

MSE $\mathcal{L}_{\text{MSE}}(x,y) = ||x-y||_2^2$, let $f(x,y) = -||x-y||_2^2$, $a(y) = 1$, we have:

$$\begin{aligned}
I(X;Y) &\geq E_{p(x,y)}[\log e^{-||x-y||_2^2}] - E_{p(x)p(y)}[e^{-||x-y||_2^2}] \\
&\geq E_{p(x,y)}[\log e^{-||x-y||_2^2}] - E_{p(x)p(y)}[e^{0}] \\
&= E_{p(x,y)}[\log e^{-||x-y||_2^2}] - 1 \\
&= E_{p(x,y)}[-||x-y||_2^2] - 1.
\end{aligned}$$

Therefore, minimizing the MSE loss between x and y can be viewed as maximizing the lower bound of $I(X;Y)$.

PKD Loss $\mathcal{L}_{\text{PKD}}(x,y) = ||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2$, let $f(x,y) = -||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2$, we have:

$$\begin{aligned}
I(X;Y) &\geq E_{p(x,y)}[\log e^{-||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2}] \\
&- E_{p(x)p(y)}[e^{-||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2}] \\
&\geq E_{p(x,y)}[\log e^{-||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2}] - E_{p(x)p(y)}[e^{0}] \\
&= E_{p(x,y)}[\log e^{-||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2}] - 1 \\
&= E_{p(x,y)}[-||\frac{x}{||x||_2} - \frac{y}{||y||_2}||_2^2] - 1.
\end{aligned}$$

Consequently, minimizing the PKD loss between x and y can be viewed as maximizing the lower bound of $I(X;Y)$.

L2 Loss $\mathcal{L}_{L_2}(x,y) = ||x-y||_2$, let $f(x,y) = -||x-y||_2$, a(y)=1, we have:

$$\begin{aligned}
I(X;Y) &\geq E_{p(x,y)}[\log e^{-||x-y||_2}] - E_{p(x)p(y)}[e^{-||x-y||_2}] \\
&\geq E_{p(x,y)}[\log e^{-||x-y||_2}] - E_{p(x)p(y)}[e^{0}] \\
&= E_{p(x,y)}[\log e^{-||x-y||_2}] - 1 \\
&= E_{p(x,y)}[-||x-y||_2] - 1.
\end{aligned}$$

Consequently, minimizing the L2 loss between x and y is equivalent to maximizing the lower bound of $I(X, Y)$.

Cosine Similarity The cosine similarity between two hidden states x and y is calculated as $\frac{x \cdot y}{||x|| \times ||y||}$, let $f(x, y) = \frac{x \cdot y}{||x|| \times ||y||} - 1, a(y) = 1$

$$\begin{aligned} I(X, Y) &\geq E_{p(x,y)}[\log e^{\frac{x \cdot y}{||x|| \times ||y||} - 1}] - E_{p(x)p(y)}[e^{\frac{x \cdot y}{||x|| \times ||y||} - 1}] \\ &\geq E_{p(x,y)}[\log e^{\frac{x \cdot y}{||x|| \times ||y||} - 1}] - E_{p(x)p(y)}[e^{0}] \\ &= \frac{1}{e} E_{p(x,y)}[\log e^{\frac{x \cdot y}{||x|| \times ||y||}}] - 1 \\ &= \frac{1}{e} E_{p(x,y)}[\frac{x \cdot y}{||x|| \times ||y||}] - 1. \end{aligned}$$

Consequently, maximizing the cosine similarity between x and y can be viewed as maximizing mutual information between x and y.

C Additional Experiment/Computation Details

C.1 Distiller Search Space

Here we recap the full search space considered for each stage of the KD pipeline in Distiller: l^{inter} $\in$ {MSE, L2, CS, PKD, MI-α (α = 0.1, 0.5, or 0.9)}, l^{pred} $\in$ {MSE, CE}, $\{m_{i,j}\}$ $\in$ {Skip, Last, EMD}, augmentation policy a is one or combinations of elementary augmentation operations in {CA, RA, BT, Mixup} .

C.2 Relationship to Other Distillation Methods

Distiller is a generic meta-framework that encompasses various KD pipelines used in previous work. For example, Distiller with the following configurations corresponds to the KD pipeline used in each of the cited works: l^{pred} = CE, l^{inter} = MSE, $m_{i,j}$ = Skip, a = CA [9]; l^{pred} = CE, l^{inter} = MSE, $m_{i,j}$ = EMD [11]; l^{pred} = CE, a = Mixup [12].

C.3 Architecture of Teacher and Student Networks

In Table 3, we use two baseline models BERT-PKD$_4$ and BERT-EMD$_4$. As described in the original paper [17], we initialize BERT-PKD$_4$ with the first 4 layers of parameters from pretrained BERT$_{\text{BASE}}$. BERT-EMD$_4$ is initialized from TinyBERT$_4$ so they have the same architecture. We list the detailed configurations of the teacher and student architectures investigated in Table 1.

C.4 Experimental Setup & Computing Details

All experiments are evaluated on GLUE [22] and SQuAD v1.1 [15] that contain classification, regression, and tagging tasks. We view SQuAD as a span part-of-speech tagging task by finding the correct answer span from the given context. The same metrics for these tasks as in the original papers [22, 15] are adopted.

[19] finds initializing students with pretrained weights is better for distillation, therefore, we initialize student models with either weights obtained from task-agnostic distillation [9] or pretrained from scratch [19]. Task-specific fine-tuned models BERT$_{\text{BASE}}$, RoBERTa$_{\text{LARGE}}$, and ELECTRA$_{\text{LARGE}}$ are considered as teacher models in our experiments. Models with comparably fewer parameters such as TinyBERT$_4$, ELECTRA$_{\text{SMALL}}$ and others detailed in Table 1 in Appendix are selected as students.

For data augmentation, We consider various elementary DA operations including: 1) MLM-based contextual augmentation (CA) , 2) random augmentation (RA), 3) backtranslation (BT) and

Table 1: Network architectures of the teacher and student models used in the paper.

Model	#Params	h_{units}	n_{layers}	h_{mid}	n_{heads}
Teacher architectures					
RoBERTa$_{\text{LARGE}}$	335M	1024	24	4096	16
ELECTRA$_{\text{LARGE}}$	335M	1024	24	4096	16
BERT$_{\text{BASE}}$	110M	768	12	3072	12
Student architectures					
TinyBERT$_6$	67M	768	6	3072	12
BERT-PKD$_4$	52M	768	4	3072	12
BERT$_{\text{MEDIUM}}$	41M	512	8	2048	8
BERT$_{\text{SMALL}}$	29M	512	4	2048	8
TinyBERT$_4$	14M	312	4	1200	12
ELECTRA$_{\text{SMALL}}$	14M	256	12	1024	4
BERT$_{\text{MINI}}$	11M	256	4	1024	4
BERT$_{\text{TINY}}$	4M	128	2	512	2

4) mixup. For contextual augmentation, we use the pretrained BERT model to do word level replacement by filling in randomly masked tokens. As in EDA [23], our random augmentation randomly swaps words in the sentence or replaces words with their synonyms. For backtranslation, we translate the sentence from one language (in this paper, English) to another language (in this paper, German) and then translate it back. Unlike existing implementations of DA in NLP that separately generate an augmented dataset first and then recall it over training, in Distiller, we apply a dynamic DA strategy where we generate a new augmented dataset every U $(= 5)$ epochs during training. We find this strategy to be more time-consuming and flexible as a DA pipeline. In addition, we use the teacher network to compute the soft label $\hat{y}$ assigned to any augmented sample $\hat{x}$.

For hyper-parameters in Equation 1, our preliminary experiments suggest that setting β_1, β_2 to 1 produces the best overall performance so we fix their values to 1 in subsequent results. γ_1 and γ_2 are set to 0.5 when DA is applied (otherwise β_2, γ_1 and γ_2 are set to 0). For controlled experiments, unless specified explicitly, we fix l^{inter} as MI-α $(\alpha = 0.9)$ and the layer mapping as "Skip". Critic functions in MI-α are powered by a shallow Transformer.

All experiments are performed on a single machine with 4 NVIDIA T4 GPUs. To reduce the hyper-parameter search space, we fix the batch size as 16 and the learning rate as 5e-5 for all experiments. We used automated mix precision FP16 for training. Maximum sequence length is set to 128 for sentence-pair tasks in GLUE, 64 for single sentence tasks, and 320 for SQuAD. Most of the experiments are trained for 15 epochs except 30 epochs for the challenging task CoLA and 30 epochs for SQuAD. As for the critic functions in MI-α, two-layer Transformer($h_{\text{mid}} = 256$, $n_{\text{heads}} = 8$) or 4-layer MLP is the choice in our implementation.

C.5 Additional Experimental Results

MI-α for Intermediate Distillation. We submitted our MI-α model predictions to the official GLUE leaderboard to obtain test set results and report the average scores over all tasks (the "AVG" column) as summarized in Table 3. The results show that the student model distilled via the MI-α objective function outperforms previous student models distilled via MSE or PKD loss. Results in Table 5 in Appendix indicate that MI-α also works well for tagging tasks. MI-α can be interpreted as learning to maximize the lower bound by updating parameters in the neural network-powered critic functions, resulting in a tighter bound than other objectives/bounds without the minimax learning.

Benefits of Data Augmentation. Data augmentation in KD provides the student additional opportunities to learn from the teacher, especially for datasets of limited size. Thus, we investigate

Table 2: We extract features from downstream datasets so that every task can be represented as a fixed-dimension embedding. The extracted embedding can be fed into *AutoDistiller* as dense features. In this table, we describe how the embedding is acquired.

Feature	Description
Context Embedding	Every document can be represented as a weighted average of the GloVe vectors, where the weights are defined by the TF-IDF scheme. Each downstream dataset is viewed as a “document” in the TF-IDF scheme. Precisely, the embedding of a dataset s is $v_s = \frac{1}{\lvert s \rvert} \sum_{w \in s} \mathrm{IDF}_w \overrightarrow{v_w}$, where $\lvert s \rvert$ denotes the number of words in the dataset, and $\mathrm{IDF}_w := \log \frac{1+N}{1+N_w}$ is the inverse document frequency of word w. N is the total number of datasets, and N_w denotes the number of datasets containing w. Intuitively, this feature represents the content of datasets.
Task Embedding	For the dataset, we collect their literal descriptions, usually one or two sentences. Then aggregate GloVe vectors of every word in these sentences and get a description embedding. This feature represents the semantic objective of the task and how the data is formatted.
Baseline Score	We use a lite Bi-LSTM model as the baseline model and fintune it on the downstream dataset. This feature aims to measure the difficulty of each task by measuring how well a simple architecture can perform on the specific dataset.
Teacher Score	The fine-tuned teacher score on the dataset. Comparing the teacher score to aforementioned baseline score tells how much boost can a complex model has on this dataset.
Number of Examples	Number of training samples in the dataset.

Table 3: Comparison of evaluation results on GLUE test set. $\text{BERT}_{\text{BASE}}$ (G) and $\text{BERT}_{\text{BASE}}$ (T) indicate the fine-tuned $\text{BERT}_{\text{BASE}}$ from [2] and the teacher model trained by ourselves, respectively. BERT-Skip-MSE_4, BERT-EMD_4, and MI-α are both initialized from TinyBERT_4, the difference is that BERT-Skip-MSE_4 is trained with “Skip” as intermediate layer mapping strategy and MSE as intermediate loss, BERT-EMD_4 is trained with “EMD” as intermediate layer mapping strategy and MSE as intermediate loss, our MI-α model is trained with “Skip” as intermediate layer mapping strategy and MI-α as intermediate loss.

Model	#Params	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	MRPC	RTE	STS-B	AVG
		(393k)	(393k)	(364k)	(108k)	(67k)	(8.5k)	(3.5k)	(2.5k)	(5.7k)	
$\text{BERT}_{\text{BASE}}$ (G)	110M	84.6	83.4	71.2	90.5	93.5	52.1	88.9	66.4	85.8	79.6
$\text{BERT}_{\text{BASE}}$ (T)	110M	84.5	83.6	71.7	90.9	93.4	49.3	87.0	67.3	84.7	79.2
BERT-PKD_4 [17]	52M	79.9	79.3	**70.2**	85.1	89.4	24.8	82.6	62.3	82.3	72.9
BERT-Skip-MSE_4	14M	81.3	80.3	69.1	86.1	90.0	25.3	85.6	63.2	80.3	73.5
BERT-EMD_4 [11]	14M	**82.1**	**80.6**	69.3	87.2	91.0	25.6	**87.6**	66.2	82.3	74.7
MI-α (α = 0.9, ours)	14M	81.9	**80.6**	69.8	**87.4**	**91.5**	**25.9**	87.0	**67.4**	**84.0**	**75.1**

the effect of DA on four data-limited tasks: CoLA, MRPC, RTE and STS-B. We also study whether various student model architectures/sizes benefit differently from DA. Table 4 demonstrates that DA generally provides a boost to student performance and is especially effective upon small models ($\text{BERT}_{\text{MINI}}$ and $\text{BERT}_{\text{TINY}}$), which is consistent with previous papers [9].

Experiments on SQuAD. We conduct an ablation study on SQuAD to verify the effectiveness of our MI-α objective and also the proposed mixup strategy for sentence tagging tasks in Section A.1. Table 5 shows that both MI-α and mixup boost comparable student performance.

MI-α constructed with different critic functions. We implement MI-α using various neural-network architectures as critic functions. Here we compare the performance of MI-α powered by two kinds of critic functions Transformer and MLP in Table 6. The result shows that a small Transformer architecture performs as a better critic function than an MLP in MI-α especially when the task is a token-level task (SQuAD v1.1).

How much do larger/better teachers help. Table 7 shows the performance of different students distilled from teachers of different sizes and pretraining schemes. From the results, we observe that although the teacher $\text{ELECTRA}_{\text{LARGE}}$ has the best performance on average score, most of the students of $\text{ELECTRA}_{\text{LARGE}}$ performs worse than students of $\text{RoBERTa}_{\text{LARGE}}$. $\text{ELECTRA}_{\text{SMALL}}$ is the only student that performs the best with $\text{ELECTRA}_{\text{LARGE}}$ as teacher, that may be attributed to

Table 4: Student performance with (out) augmentation (augmenter initialized as CA+RA+mixup). We report the relative improvement for rows starting with "+ aug".

Model	#Params	CoLA	MRPC	RTE	STS-B	AVG
		mcc	f1/acc	acc	spearman/pearson	
BERT$_{\text{BASE}}$ (T)	110M	55.0	89.6/85.0	65.0	88.4/88.6	78.6
TinyBERT$_6$	67M	51.3	**92.5/89.7**	**75.5**	89.6/89.8	**81.4**
+ aug		**+0.1**	-1.1/-1.8	-3.3	**+0.2/+0.2**	-1.0
BERT$_{\text{MEDIUM}}$	41M	44.1	**89.3/84.8**	65.3	88.3/88.6	76.7
+ aug		**+5.3**	-0.4/-0.7	**+4.4**	**+0.6/+0.5**	**+1.6**
BERT$_{\text{SMALL}}$	29M	37.4	86.8/80.6	64.6	87.7/88.0	74.2
+ aug		**+5.0**	**+0.1/+0.8**	**+0.4**	**+0.3/+0.2**	**+1.1**
TinyBERT$_4$	14M	23.6	88.9/**83.8**	67.1	88.0/88.1	73.3
+ aug		**+7.9**	**+0.3/+0.0**	**+2.2**	**+0.7/+0.7**	**+2.0**
ELECTRA$_{\text{SMALL}}$	14M	42.8	88.3/83.8	66.4	87.4/87.5	76.0
+ aug		**+16.2**	**+3.4/+3.7**	**+1.8**	**+1.0/+1.0**	**+4.5**
BERT$_{\text{MINI}}$	11M	11.2	**86.1/80.1**	62.8	87.1/**87.2**	69.1
+ aug		**+23.2**	**+0.0**/-0.1	**+3.3**	**+0.2/+0.0**	**+4.4**
BERT$_{\text{TINY}}$	4M	6.0	83.2/73.3	60.0	84.0/83.6	65.0
+ aug		**+6.6**	**+1.7/+3.7**	**+4.3**	**+0.1/+0.7**	**+2.9**

Table 5: Ablation study of distillation performance on SQuAD v1.1 dev set. The first line is the statistics of the BERT$_{\text{BASE}}$ teacher. ELECTRA$_{\text{SMALL}}$ and TinyBERT$_6$ are two student networks. ELECTRA$_{\text{SMALL}}$ (FT) means to fine-tune ELECTRA$_{\text{SMALL}}$ without KD. TinyBERT$_6$† represents results obtained from [9]. Models that end with "(MSE)" are distilled with the MSE loss while "+ MI-α" stands for MI-α (α=0.9) as the intermediate loss function. "+ mixup" means to further apply the mixup augmentation.

Model	**SQuAD v1.1**	
	EM	F1
BERT$_{\text{BASE}}$ (T)	80.9	88.2
ELECTRA$_{\text{SMALL}}$ (FT)	75.3	83.5
ELECTRA$_{\text{SMALL}}$ (MSE)	79.2	86.8
+ MI-α	79.0	86.8
+ mixup, MSE	80.1	87.4
+ mixup, MI-α	**80.2**	**87.6**
TinyBERT$_6$† [9]	79.7	87.5
TinyBERT$_6$ (MSE)	77.8	85.5
+ MI-α	80.0	87.8
+ mixup, MSE	78.6	86.2
+ mixup, MI-α	**81.1**	**88.6**

ELECTRA$_{\text{SMALL}}$ and ELECTRA$_{\text{LARGE}}$ are pretrained on the same pretraining task, so they have a similar knowledge representation scheme. And also, for datasets MNLI, QQP, QNLI, and SST-2, which have abundant amount of training data, students of BERT$_{\text{BASE}}$ perform better.

Benefits of Intermediate Distillation. fANOVA results in Figure 2 indicate that intermediate distillation achieves the highest importance among the components in distillation pipeline. To further validate this observation, we compare distillation results between settings where intermediate distillation is used or not. Table 8 shows that large datasets (>5k training sample) clearly benefit from intermediate distillation, which can be interpreted as efficient data provides more iterations for students to query the teacher and learn the intermediate function composition. This interpretation shares the similar inner thought with our findings about data augmentation in Section C.5.

Table 6: Comparison of evaluation results on GLUE test set. We compare the distillation performance when using MLP and Transformer as critic functions in MI-α respectively. BERT$_{\text{BASE}}$ (T) indicates the teacher model trained by ourselves. Both of the students are initialized with TinyBERT$_4$ [9] and distilled with "Skip" as intermediate layer mapping strategy and MI-α as intermediate objective functions. The difference is, TinyBERT$_4$ (MLP) is trained with a 4-layer MLP with hidden state of 512 as critic function while TinyBERT$_4$ (Transformer) uses a 2-layer Transformer with feed-forward hidden size 256.

Model	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	MRPC	RTE	STS-B	SQuAD v1.1	AVG
	(393k)	(393k)	(364k)	(108k)	(67k)	(8.5k)	(3.5k)	(2.5k)	(5.7k)	(108k)	
BERT$_{\text{BASE}}$ (T)	84.5	83.6	71.7	90.9	93.4	49.3	87.0	67.3	84.7	88.2	80.0
TinyBERT$_4$ (MLP)	81.7	**80.6**	69.7	**87.6**	**91.6**	24.9	**87.0**	**67.4**	81.9	70.1	74.3
TinyBERT$_4$ (Transformer)	**81.9**	**80.6**	**69.8**	87.4	91.5	**25.9**	**87.0**	**67.4**	**84.0**	**71.7**	**74.7**

Table 7: Performance comparison with different teacher and student models. We abbreviate three teacher models BERT$_{\text{BASE}}$, RoBERTa$_{\text{LARGE}}$ and ELECTRA$_{\text{LARGE}}$ as B, R, E. Results are evaluated on GLUE dev set and best results are in-bold.

Model	#Params	Teacher	MNLI-m	MNLI-mm	QQP		QNLI	SST-2	CoLA	MRPC		RTE	STS-B		AVG
			acc	acc	f1	acc	acc	acc	mcc	f1	acc	acc	spearman	pearson	
BERT$_{\text{BASE}}$ (T)	110M		84.1	84.7	88.0	91.1	91.7	93.0	55.0	89.6	85.0	65.0	88.4	88.6	83.7
RoBERTa$_{\text{LARGE}}$ (T)	335M		90.2	90.1	89.6	92.1	94.7	96.3	64.6	91.3	88.0	78.7	**91.7**	**91.8**	88.3
ELECTRA$_{\text{LARGE}}$ (T)	335M		**90.5**	**90.4**	**90.3**	**92.8**	**95.1**	**96.6**	**67.4**	**91.7**	**88.5**	**84.5**	88.7	88.9	**88.8**
BERT$_{\text{BASE}}$	110M	R	**84.5**	**84.6**	88.6	91.5	**91.7**	**93.2**	59.3	91.6	88.0	66.4	89.0	89.4	84.8
		E	84.4	**84.6**	**88.8**	**91.7**	91.6	92.8	**59.5**	**91.9**	**88.7**	**69.3**	**89.1**	**89.6**	**85.2**
TinyBERT$_6$	67M	B	**83.9**	**84.0**	**88.1**	**91.2**	**91.3**	91.6	**50.5**	90.3	86.5	75.5	89.4	89.4	**84.3**
		R	83.5	83.5	88.0	**91.2**	90.8	**92.2**	48.0	**91.9**	**88.7**	72.6	**89.9**	**90.0**	84.2
		E	83.0	83.0	87.8	91.0	90.6	91.3	48.6	91.6	88.5	**76.2**	89.1	89.3	84.2
BERT$_{\text{MEDIUM}}$	41M	B	**82.6**	**83.0**	**87.9**	**91.0**	**90.0**	90.8	48.3	88.9	84.1	**65.0**	**88.2**	88.4	**82.4**
		R	80.9	81.4	87.6	90.8	89.0	**91.4**	50.5	88.9	84.6	64.3	**88.2**	**88.6**	82.2
		E	81.0	81.3	87.5	90.7	89.0	90.9	**51.0**	**89.5**	**85.3**	64.3	88.0	88.2	82.2
BERT$_{\text{SMALL}}$	29M	B	**81.0**	**81.0**	**87.4**	**90.6**	**87.3**	**90.5**	**43.1**	87.8	82.4	63.5	87.0	87.2	**80.7**
		R	78.7	78.6	87.0	90.4	87.0	88.6	41.2	**89.1**	**84.1**	**64.3**	**87.1**	**87.3**	80.3
		E	78.6	78.8	87.2	90.5	87.0	89.3	43.0	88.7	**84.1**	63.9	86.8	87.1	80.4
TinyBERT$_4$	14M	B	**81.1**	**81.6**	**87.2**	**90.4**	**87.4**	**90.6**	12.3	89.4	85.0	66.4	87.7	87.8	78.9
		R	80.0	80.7	86.5	90.0	86.0	89.4	**24.6**	90.4	86.5	67.9	**88.0**	**88.1**	**79.8**
		E	80.0	80.2	86.2	89.6	85.9	88.9	21.8	**90.9**	**86.8**	**68.6**	87.6	87.6	79.5
ELECTRA$_{\text{SMALL}}$	14M	B	**82.7**	**83.8**	87.8	90.9	**89.7**	91.3	**60.6**	91.3	87.7	60.6	87.4	87.5	83.5
		R	82.3	83.2	88.1	91.2	89.5	90.6	58.6	91.3	87.5	67.5	**87.6**	**87.8**	83.8
		E	82.0	82.7	**88.5**	**91.5**	89.3	**91.4**	**60.6**	**92.3**	**89.0**	**69.7**	86.6	86.7	**84.2**
BERT$_{\text{MINI}}$	11M	B	**78.5**	**79.7**	**86.6**	**90.0**	**84.9**	**87.8**	20.1	**87.0**	**81.6**	61.0	**86.2**	86.1	77.5
		R	76.6	77.3	86.0	**90.0**	84.6	85.9	32.2	86.6	81.1	**65.3**	**86.2**	**86.3**	**78.2**
		E	76.3	77.1	85.9	89.5	84.2	85.9	**33.8**	**87.0**	**81.6**	65.0	85.7	85.5	78.1
BERT$_{\text{TINY}}$	4M	B	**72.8**	**73.4**	**83.5**	**87.2**	**81.3**	**83.9**	0.0	84.5	75.7	58.5	81.4	**79.9**	71.8
		R	71.5	72.0	82.9	86.7	80.2	83.1	6.2	84.9	76.2	60.3	**81.9**	**79.9**	**72.1**
		E	71.2	71.8	82.9	87.0	80.0	82.8	**6.7**	**85.2**	**77.0**	**62.8**	78.4	77.3	71.9

Table 8: Comparison of evaluation results on GLUE dev set. We compare the distillation performance with (out) intermediate distillation. A BERT$_{\text{BASE}}$ model is used as the teacher and TinyBERT$_4$ is the student. TinyBERT$_4$ (KD) represents using a vanilla knowledge distillation (student only learns from the outputs of teacher) and "+intermediate distillation" represents using vanilla KD and intermediate distillation.

Model	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	MRPC	RTE	STS-B	SQuAD v1.1	AVG
	(393k)	(393k)	(364k)	(108k)	(67k)	(8.5k)	(3.5k)	(2.5k)	(5.7k)	(108k)	
	acc	acc	f1/acc	acc	acc	mcc	f1/acc	acc	spearman/pearson	f1/em	
BERT$_{\text{BASE}}$ (T)	84.1	84.7	88.0/91.1	91.7	93.0	55.0	89.6/85.0	65.0	88.4/88.6	88.2/80.9	83.8
TinyBERT$_4$ (KD)	80.1	80.3	86.4/89.7	85.8	89.1	16.1	**89.6/85.3**	**66.8**	**88.4/88.5**	77.3/66.8	77.9
+intermediate distillation	**80.7**	**81.3**	**87.0/90.2**	**86.8**	**90.0**	**21.3**	89.3/84.8	65.3	88.2/88.4	**79.4/69.4**	**78.7**

More about AutoDistiller. We evaluate the performance of *AutoDistiller* on the 8 GLUE datasets via a leave-one-dataset-out cross-validation protocol. Figure 5 shows that *AutoDistiller* achieves positive Spearman's rank correlation coefficients for most datasets.

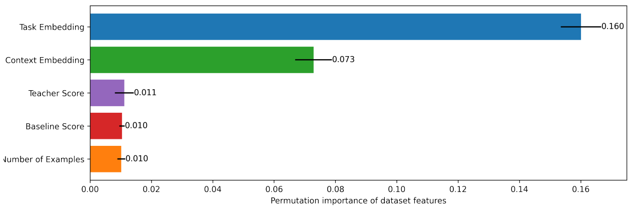


Figure 4: Permutation importance of the five dataset features.

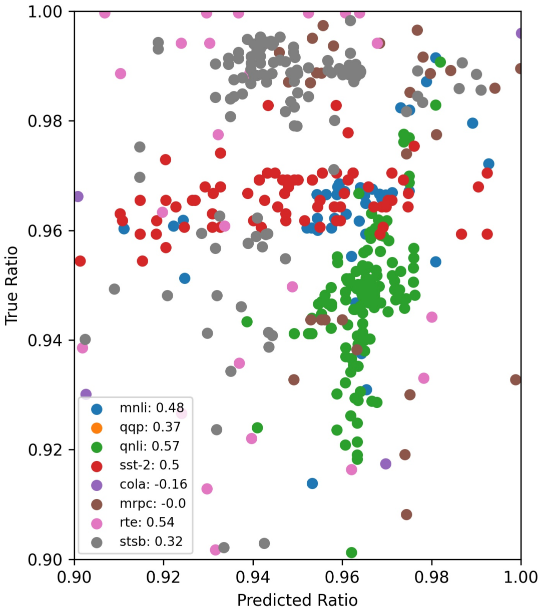


Figure 5: Evaluating held-out AutoDistiller predictions on GLUE via leave-one-out estimates. We use dataset-level cross-validation that holds out each GLUE dataset from AutoDistiller training. For each held-out dataset, the legend lists Spearman's rank correlation between the predicted vs. actual ratio (teacher's performance achieved by student's) across different KD pipelines. The average Spearman's rank correlation value across the 8 datasets is 0.33.

D Top Configurations in Distiller

Here we list the top 5 configurations from the Distiller search space that performed best on each dataset in Table 9.

Table 9: Top 5 configurations to distill a $\text{BERT}_{\text{BASE}}$ teacher to a TinyBERT_4 student on every dataset. To reduce the search space, we only compare configurations that don't use data augmentation. As Hyper-parameter α is only valid for MI-α, the value of α is set to N for other intermediate loss in the table.

Task	Intermediate Loss	α	Layer Mapping Strategy	KD Loss	#Example	Score
MNLI	MI-α	0.9	EMD	MSE	393000	81.7
MNLI	MI-α	0.1	EMD	MSE	393000	81.7
MNLI	MI-α	0.5	Skip	MSE	393000	81.7
MNLI	MSE	N	EMD	MSE	393000	81.6
MNLI	MSE	N	Skip	MSE	393000	81.6
QQP	MI-α	0.9	EMD	MSE	364000	90.2
QQP	MI-α	0.1	EMD	MSE	364000	90.2
QQP	CE	N	Skip	MSE	364000	90.2
QQP	MI-α	0.1	EMD	MSE	364000	90.1
QQP	MI-α	0.1	Skip	MSE	364000	90.1
QNLI	CE	N	Last	MSE	105000	87.4
QNLI	MI-α	0.5	Skip	MSE	105000	87.4
QNLI	MI-α	0.5	Last	CE	105000	87.3
QNLI	MI-α	0.9	Skip	CE	105000	87.2
QNLI	MI-α	0.1	Skip	CE	105000	87.1
SST-2	Cos	N	Last	MSE	67000	90.6
SST-2	MSE	N	Skip	CE	67000	90.5
SST-2	MI-α	0.1	Last	CE	67000	90.3
SST-2	CE	N	Skip	MSE	67000	90.3
SST-2	MI-α	0.9	Skip	CE	67000	90.3
CoLA	MI-α	0.1	EMD	MSE	8500	22.3
CoLA	MI-α	0.5	EMD	MSE	8500	21.6
CoLA	MI-α	0.5	EMD	CE	8500	21.1
CoLA	MI-α	0.1	Last	MSE	8500	21.1
CoLA	MI-α	0.5	Skip	MSE	8500	21.0
MRPC	MI-α	0.1	EMD	CE	3700	90.3
MRPC	MI-α	0.5	EMD	CE	3700	90.2
MRPC	CE	N	Skip	CE	3700	89.9
MRPC	MI-α	0.9	EMD	CE	3700	89.9
MRPC	CE	N	Last	MSE	3700	89.7
RTE	MI-α	0.1	Skip	CE	2500	70.8
RTE	MI-α	0.9	Skip	CE	2500	70.4
RTE	MI-α	0.5	Skip	CE	2500	70.0
RTE	MI-α	0.1	Last	CE	2500	70.0
RTE	MI-α	0.5	Last	CE	2500	69.3
STS-B	MI-α	0.9	Skip	MSE	7000	88.0
STS-B	MI-α	0.5	Skip	MSE	7000	88.0
STS-B	MI-α	0.9	EMD	MSE	7000	87.9
STS-B	MI-α	0.1	Last	MSE	7000	87.9
STS-B	PKD	N	Skip	MSE	7000	87.9
SQuAD v1.1	MSE	N	Skip	MSE	130000	72.6
SQuAD v1.1	CE	N	Skip	MSE	130000	72.4
SQuAD v1.1	MSE	N	EMD	MSE	130000	72.3
SQuAD v1.1	MI-α	0.9	Skip	CE	130000	71.9
SQuAD v1.1	MI-α	0.9	Skip	CE	130000	71.7