Weight-Inherited Distillation for Task-Agnostic BERT Compression

Taiqiang Wu$^{1,2,*}$, Cheng Hou$^2$, Shanshan Lao$^3$, Jiayi Li$^3$, Ngai Wong$^3$, Zhe Zhao$^{2,†}$ and Yujiu Yang$^3$†

$^1$The University of Hong Kong, $^2$Tencent AI Lab, $^3$Shenzhen International Graduate School, Tsinghua University
takiwu@connect.hku.hk

Abstract

Knowledge Distillation (KD) is a predominant approach for BERT compression. Previous KD-based methods focus on designing extra alignment losses for the student model to mimic the behavior of the teacher model. These methods transfer the knowledge in an indirect way. In this paper, we propose a novel Weight-Inherited Distillation (WID), which directly transfers knowledge from the teacher. WID does not require any additional alignment loss and trains a compact student by inheriting the weights, showing a new perspective of knowledge distillation. Specifically, we design the row compactors and column compactors as mappings and then compress the weights via structural re-parameterization. Experimental results on the GLUE and SQuAD benchmarks show that WID outperforms previous state-of-the-art KD-based baselines. Further analysis indicates that WID can also learn the attention patterns from the teacher model without any alignment loss on attention distributions. The code is available at GitHub.

1 Introduction

Transformer-based Pre-trained Language Models (PLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLNET (Yang et al., 2019), have achieved great success in many Natural Language Process (NLP) tasks. These models are pre-trained on massive corpus via self-supervised tasks to learn contextualized text representations. However, PLMs have high costs in terms of storage, memory, and computation time, which brings challenges to online services in real-life applications. Therefore, it is crucial and feasible to compress PLMs while maintaining their performance.

Knowledge Distillation (KD), which trains a compact student model by mimicking the behavior of a teacher model, is a predominant method for PLM compression. There are two settings for KD in BERT compression: 1) task-specific, which first fine-tunes the teacher PLMs on specific tasks and then performs distillation, and 2) task-agnostic, which distills PLMs in the pre-training stage. For task-agnostic distillation, the student model can be directly and generically fine-tuned on various downstream tasks (Wang et al., 2020; Sun et al., 2020). Hence, we evaluate the proposed weight-inherited distillation (WID) under a task-agnostic setting.

Previous KD-based methods mainly focus on designing alignment losses to minimize the distance between the teacher model and the student model. We can further categorize these alignment losses into 1) logit-based, which measures the distance of logit distributions, and 2) feature-based, which aims to align the intermediate features including token embeddings, hidden states, and self-attention distributions. However, selecting various loss functions and balancing the weights of each loss are laborious (Sun et al., 2019; Jiao et al., 2020). Meanwhile, the knowledge is embeded in the weights. This gives rise to an intuitive thought: can we distill the knowledge by directly inheriting the weights, rather than aligning the logit distributions or intermediate features?

<table>
<thead>
<tr>
<th>Approach</th>
<th>Alignment Loss</th>
<th>Hard Loss</th>
<th>Task-Agnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logit Feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DistilBERT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TinyBERT (GD)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PKD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MinLM</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MobileBERT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WID (ours)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Comparison with previous state-of-the-art distillation methods. Logit and Feature denote whether logit-based loss and feature-based loss are used for distillation. To the best of our knowledge, WID is the first distillation method without any alignment loss and directly transfers the knowledge by weight inheritance.
In this work, we propose Weight-Inherited Distillation (WID), which does not require any additional alignment loss and trains the student by directly inheriting the weights from the teacher. In WID, we factorize the KD process into the compression of each weight matrix. Inspired by structural re-parameterization in CNN compression (Ding et al., 2021), we design row compactors and column compactors, and then view them as mappings to compress the weights by row and column, respectively. For the matrices to compress the row only, such as the output layer for MLM task (the column is always the size of vocabulary), we employ the row compactors exclusively to compress them. Moreover, during training, we design a novel alignment strategy to align the compactors due to the residual connection in Transformer (Vaswani et al., 2017). As shown in Table 1, WID is the only method for task-agnostic distillation without any alignment loss.

We conduct extensive experiments on downstream NLP tasks, including the GLUE and SQuAD benchmarks. Experimental results demonstrate that WID outperforms traditional KD-based baselines. Further analysis shows that WID can also learn high-level semantic knowledge such as self-attention patterns via inheriting weights.

Our contributions can be summarized as follows:

• We propose Weight-Inherited Distillation (WID), revealing a new pathway to KD by directly inheriting the weights via structural re-parameterization.

• We design the compactor alignment strategy and conduct WID for task-agnostic BERT compression. Experiments on the GLUE and SQuAD benchmark datasets demonstrate the effectiveness of WID for model compression.

• We perform further analyses on how to get better performance in BERT compression. Even more, we find that WID is able to learn attention patterns from the teacher.

2 Preliminaries

2.1 Embedding Layer

In BERT (Devlin et al., 2019), the input texts are tokenized to tokens by WordPiece (Wu et al., 2016). The representations \( \{x_i\}_{i=1}^{|x|} \) of the input sequence are constructed by summing the corresponding token embedding, segment embedding, and position embedding. For the token embedding layer in BERT, the weight is \( W_T \in \mathbb{R}^{|V| \times d} \), where \(|V|\) and \(d\) denote the sizes of the vocabulary and the hidden state vector, respectively.

2.2 Transformer Layer

Transformer layers are adapted to encode the contextual information of input texts. The input vector \( \{x_i\}_{i=1}^{|x|} \) are tokenized to tokens by WordPiece (Wu et al., 2016). The representations \( \{x_i\}_{i=1}^{|x|} \) are employed, followed by Layer Normalization (LN) (Ba et al., 2016).

MHA For the \( l \)-th transformer layer with \( A \) attention heads, the output \( O_{l,a} \) of the attention head \( a \in [1, A] \) is calculated as:

\[
O_{l,a} = A_{l,a} V_{l,a} W_{l,a}^Q \sqrt{d_k}
\]

where linear projection \( W_{l,a}^Q, W_{l,a}^K, W_{l,a}^V \in \mathbb{R}^{d \times d_k} \) and \( d_k = \frac{d}{A} \) is the dimension of each head. The final output of MHA sub-layer is as follows:

\[
O_l = \text{LN}(H^{l-1} + (\|O_{l,a}\|_{a=1} A_{l,a} V_{l,a} W_{l,a}^Q))
\]
2.3 Knowledge Distillation

Knowledge Distillation (KD) trains a compact student model \( S \) by mimicking the behaviors of the teacher model \( T \). The losses can be categorized into logit-based and feature-based.

For logit-based loss, the target is to minimize the distance between logit distribution \( p_s \) from the student and \( p_t \) from the teacher, which can be formalized as:

\[
\mathcal{L}_{\text{logit}} = \mathcal{H}_1(p_s/\tau, p_t/\tau),
\]

where \( \tau \) is the temperature and \( \mathcal{H}_1 \) is the cross-entropy loss or KL-divergence.

Feature-based loss aims to align the intermediate features between the teacher and the student by:

\[
\mathcal{L}_{\text{feature}} = \mathcal{H}_2(f^{S}(x), f^{T}(x)),
\]

where \( \mathcal{H}_2 \) is the loss function such as Mean Square Error (MSE) and \( f(x) \) denotes for the intermediate output including hidden state vector \( \mathbf{H} \) and attention distribution \( \mathbf{A} \).

As shown in Table 1, logit-based and feature-based loss can be jointly employed for better distillation. However, balancing the weights of each loss is laborious. For example, the overall loss of PKD (Sun et al., 2019) is:

\[
\mathcal{L} = (1 - \alpha)\mathcal{L}_{\text{hard}} + \alpha\mathcal{L}_{\text{logit}} + \beta\mathcal{L}_{\text{feature}},
\]

where \( \mathcal{L}_{\text{hard}} \) is the loss on target tasks and \( \alpha \) and \( \beta \) are the hyper-parameters. PKD performs grid search over \( \alpha \) and \( \tau \), where \( \alpha \in \{0.2, 0.5, 0.7\} \) and \( \tau \in \{5, 10, 20\} \). After that, the best \( \alpha \) and \( \tau \) are fixed, followed by a search of \( \beta \in \{10, 100, 500, 1000\} \).

Meanwhile, selecting various loss functions is also laborious. In PKD, \( \mathcal{L}_{\text{feature}} \) is defined as the mean square loss between the normalized hidden states for each layer. DistilBERT (Sanh et al., 2019) adopts the cosine embedding loss for hidden states. TinyBERT (Jiao et al., 2020) employs the mean square loss for self-attention distributions, embedding layer outputs, and hidden states.

3 Weight-Inherited Distillation

3.1 Structural Re-parameterization

As mentioned in Section 2, the PLMs (e.g., BERT) consist of embedding layers and transformer layers. To compress the BERT, we have to learn a mapping from the larger weight in the teacher model to the compact one. In terms of matrices, these mappings can be categorized as:

\[
XW^{LT} = XW^{rc}W^{LT}W^{cc}.
\]

Second, we train the re-parameterized teacher model on the pre-training task. After training,
1. Train Stage

2. Compress Stage

Figure 2: Training and compression for column compactor. During the training process, we add weight penalty gradients by columns and progressively select the mask to fuse the penalty gradients and original loss gradients. After training, we compress the column compactor following the column mask.

the row compactor is compressed by reducing the $B - D$ rows, and the column compactor is compressed by reducing $C - E$ columns. The objects are as follows:

$$W^{rc}_{cc} \in \mathbb{R}^{B \times B} \rightarrow W^{rc'}_{cc} \in \mathbb{R}^{D \times B}$$

$$W^{cc} \in \mathbb{R}^{C \times C} \rightarrow W^{cc'} \in \mathbb{R}^{C \times E}.$$  \hspace{1cm} (11)

More details can be found in Section 3.2. Finally, we merge the compressed compactors $W^{rc'}$, $W^{cc'}$ and the original teacher layer $W^{L_T}$ to obtain the compact layer for the student following:

$$W^{L_S} = W^{rc'} W^{L_T} W^{cc'} \in \mathbb{R}^{D \times E}$$ \hspace{1cm} (12)

For the weights to compress the rows only, we adopt the row compactor exclusively. Similarly, we employ the column compactor exclusively for the weights to compress the columns only.

3.2 Compactor Compression

The goal is to maintain the performance of the teacher model as much as possible and compress the compactors simultaneously.

Figure 2 presents the training and compression process for the column compactor. To compress the compactors, we add extra penalty loss to minimize the norms of some columns. Given the column compactor $W^{cc} \in \mathbb{R}^{C \times C}$ and original gradients $g^{cc}_{ori} \in \mathbb{R}^{C \times C}$ from training tasks, the penalty gradients $g^{cc}_{pen} \in \mathbb{R}^{C \times C}$ are calculated as follows:

$$g^{cc}_{pen} = \frac{W^{cc}}{|W^{cc}|_2}$$ \hspace{1cm} (13)

where $|W^{cc}|_2$ denotes the Euclidean norm across each column.

However, applying the $g^{cc}_{ori}$ and penalty gradients $g^{cc}_{pen}$ to the same row/column leads to the gradient competition (Ding et al., 2021). Therefore, we choose some columns to reduce and apply the penalty gradients $g^{cc}_{pen}$, while the rest columns are adopted to keep performance and updated with $g^{cc}_{ori}$.

Specifically, we pick top-$k$ columns with lower norm value based on the $|W^{cc}|_2$ and set the corresponding value in our column mask $M = \{0, 1\}^C$ to be 1. Later, the original gradients $g^{cc}_{ori}$ and the penalty gradients $g^{cc}_{pen}$ are fused as follows:

$$g^{cc}_{fused}[;i] = \begin{cases} 
    g^{cc}_{pen}[;i], & \text{if } M[i] = 1 \\
    g^{cc}_{ori}[;i], & \text{if } M[i] = 0
\end{cases}$$ \hspace{1cm} (14)

where $0 \leq i \leq C$. We employ the fused gradients $g^{cc}_{fused}$ to update the corresponding column compactor. After training, we compress the column compactor by column mask:

$$W^{cc'} = W^{cc}[;i], \text{ where } M[i] = 0.$$ \hspace{1cm} (15)

Moreover, the process is similar for row compactors. We calculate $|W^{rc}|_2$ for each row and select the top-$k$ rows with the lower norm value.

For stability and better performance, we choose the rows/columns of the compactors progressively. Concretely, we increase $k$ by $d$ for $N$ steps until reaching the desired size during the training stage. Moreover, we also try the dynamic selection (Ding et al., 2021) for mask and it makes no effect.
3.3 Compactor Alignment Strategy

To apply WID for BERT compression, we design a novel compactor alignment strategy. Since each dimension in a hidden representation $h_1$ is connected to the same dimension in another hidden representation $h_2$ through a residual connection, the compactors before and after the $h_1$ and $h_2$ need to be aligned. As shown in Figure 3, the compactors in a transformer block are divided into three groups (same color, same group). The first compactor before the $H^{l-1}$ and the first compactor after the $H^l$ are also aligned with groups in blue. Therefore, the column compactor for the embedding layer, the row compactor for the output layer, and compactors in blue from each layer are all aligned. Meanwhile, the groups in orange/green can be different across layers since they are not adjacent. For each group of the aligned compactors, we learn one of them and duplicate (or, flip) it for the rest. Please refer to Appendix B.2 for more details.

4 Experiments

4.1 Task-Agnostic Distillation

We employ the uncased version of BERT$_{\text{base}}$ as our teacher model and implement WID based on TencentPretrain framework (Zhao et al., 2023). BERT$_{\text{base}}$ (Devlin et al., 2019) is a 12-layer transformer model ($d=768$, $A=12$, $L=12$), which contains 110M parameters. For student models, we compress the teacher model to various model sizes for comparison, including WID$_{55}$ ($d=516$, $A=12$, $L=12$) with 55M parameters and WID$_{11}$ ($d=192$, $A=12$, $L=12$) with 11M parameters. We use the documents of English Wikipedia and BookCorpus (Zhu et al., 2015) for pre-training following Devlin et al. (2019). We use AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.99$. The compactors are trained with peak learning rate 5e-5 and the original linear layers with peak learning rate 1e-6. For WID, we adopt the 2-norm and set $N=500$, $d=\lfloor (d_t - d_s)/16 \rfloor$. It costs about 64 hours to train for 400,000 steps with a batch size of 960 on 8 A100 GPUs.

4.2 Downstream Tasks

Following previous PLM-based KD methods (Sanh et al., 2019; Wang et al., 2020), we evaluate our WID on the SQuAD v1.1 (Rajpurkar et al., 2016) and GLUE benchmark (Wang et al., 2019). The GLUE benchmark consists of CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QQP (Chen et al., 2018), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016) and RTE (Bentivogli et al., 2009). After task-agnostic distillation, we fine-tune our compressed BERT WID$_{55}$ and WID$_{11}$ on these benchmarks adopting the grid search and report the results on the development sets. The result of MNLI is the score of MNLI-m. More details about these datasets including dataset sizes and metrics and the hyperparameters for fine-tuning can be found in the Appendix A.

4.3 Baselines

For a fair comparison, we compare our WID with the task-agnostic distillation baselines. These baselines include: 1) DistilBERT (Sanh et al., 2019), which distills the student by the combination of the original MLM loss, the cosine distance for features, and the KL divergence for output logits. 2) TinyBERT (GD) (Jiao et al., 2020), which aligns the attention distributions and hidden states
for general distillation. 3) MiniLM (Wang et al., 2020) and MiniLM v2 (Wang et al., 2021), which align the attention matrix and values-values scaled dot-product. We also reproduce the TinyBERT in the same architecture as WID, following the official code. For fair comparison, we employ the same corpus and follow the official hyperparameters. We do not compare with MobileBERT (Sun et al., 2020) since its teacher is IB-BERT (large) (much higher accuracy than BERT\textsubscript{base}) and its computations (4096 batch size, 740,000 steps) is much higher. Moreover, we also compare WID with task-specific methods in Appendix C.1.

### 4.4 Main Results

We compare WID with other task-agnostic distillation methods in various model sizes. All the methods utilize the BERT\textsubscript{base} as the teacher model. As shown in Table 2, WID retains 98.9% and 90.9% performance of BERT\textsubscript{base} using only 49.2% and 10.2% parameters, respectively. In particular, in the CoLA task, WID\textsubscript{SS} gets a higher score than BERT\textsubscript{base}. Compared to the baselines with 67.5M parameters, WID\textsubscript{SS} gets comparable performance with MiniLM and higher performance than DistilBERT with fewer parameters. Meanwhile, WID outperforms the TinyBERT under the same architecture on GLUE benchmarks and SQuAD, showing its superiority over the traditional KD methods with logit-based loss and feature-based loss. Without CoLA, WID\textsubscript{SS} gets an average score of 85.8 and still outperforms the TinyBERT (GD) with an average score of 85.0.

Meanwhile, we apply WID for generative PLM. Please refer to C.4 for more details.

**Larger Performance Gap** Since the performance gap between teacher and student has always been a crucial point and difficulty in KD, we conduct experiments for smaller student models (11.3M parameters). We reproduce the task-agnostic TinyBERT under the General Distillation (GD) as the baseline. As shown in Table 2, we find that WID (average score: 76.7) still outperforms TinyBERT (average score: 75.6) when the student model is about 10x smaller.

### 5 Analysis and Discussion

#### 5.1 WID vs Pruning

Pruning (LeCun et al., 1989) aims to remove redundant weights from a neural network to achieve parameter-efficiency while preserving model performance, including unstructured pruning which sets weights to 0, and structured pruning which removes components such as attention heads. Unstructured pruning methods do not reduce the model size. However, WID is very likely to be confused with structured pruning methods.

Structured pruning methods aim to remove the redundant units and then usually get sub-networks without a pre-defined structure. However, WID does not remove any parts of the original weights from the teacher models but learns a student model with a pre-defined structure. Meanwhile, the goal of KD is to transfer the knowledge from teacher models to student models. In WID, we design the compactors as mappings to inherit knowledge from teacher models, rather than to find sub-networks. Hence, we consider WID as a KD method though the compression process of compactors is similar to pruning. More comparison between WID and pruning methods can be found in C.2.

---

### Table 2: Comparison between our WID and various task-agnostic distillation methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
<th>Params</th>
<th>SST-2</th>
<th>CoLA</th>
<th>MRPC</th>
<th>QNLI</th>
<th>QQP</th>
<th>RTE</th>
<th>STS-B</th>
<th>MNLI</th>
<th>SQuAD</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{base}</td>
<td>22.7B</td>
<td>110.1M</td>
<td>92.7</td>
<td>59.1</td>
<td>90.4</td>
<td>91.7</td>
<td>91.4</td>
<td>70.8</td>
<td>90.1</td>
<td>84.5</td>
<td>89.6/82.6</td>
<td>84.3</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>11.9B</td>
<td>67.5M</td>
<td>91.3</td>
<td>51.3</td>
<td>87.5</td>
<td>89.2</td>
<td>88.5</td>
<td>59.9</td>
<td>86.9</td>
<td>82.2</td>
<td>86.2/78.1</td>
<td>80.1</td>
</tr>
<tr>
<td>MiniLM</td>
<td>11.9B</td>
<td>67.5M</td>
<td>92.0</td>
<td>49.2</td>
<td>88.4</td>
<td>91.0</td>
<td>91.0</td>
<td>71.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MiniLM v2</td>
<td>11.9B</td>
<td>67.5M</td>
<td>92.4</td>
<td>52.5</td>
<td>88.9</td>
<td>90.8</td>
<td>91.1</td>
<td>72.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TinyBERT (GD)(†)</td>
<td>11.9B</td>
<td>67.5M</td>
<td>92.9</td>
<td>44.1</td>
<td>89.5</td>
<td>90.7</td>
<td>91.0</td>
<td>73.7</td>
<td>89.6</td>
<td>83.8</td>
<td>84.0/74.2</td>
<td>81.3</td>
</tr>
<tr>
<td>TinyBERT (GD)(‡)</td>
<td>10.4B</td>
<td>54.9M</td>
<td>92.3</td>
<td>47.0</td>
<td>87.3</td>
<td>90.8</td>
<td>90.9</td>
<td>69.7</td>
<td>89.0</td>
<td>83.3</td>
<td>85.4/76.2</td>
<td>81.2</td>
</tr>
<tr>
<td>WID\textsubscript{SS} (ours)</td>
<td>10.4B</td>
<td>54.9M</td>
<td>92.4</td>
<td>61.7</td>
<td>88.2</td>
<td>90.1</td>
<td>91.0</td>
<td>70.4</td>
<td>87.9</td>
<td>82.9</td>
<td>88.5/80.8</td>
<td>83.4</td>
</tr>
<tr>
<td>TinyBERT (GD)(†)</td>
<td>1.6B</td>
<td>11.3M</td>
<td>88.4</td>
<td>30.3</td>
<td>80.4</td>
<td>87.5</td>
<td>89.1</td>
<td>65.3</td>
<td>84.0</td>
<td>79.4</td>
<td>80.5/70.7</td>
<td>75.6</td>
</tr>
<tr>
<td>WID\textsubscript{SS} (ours)</td>
<td>1.6B</td>
<td>11.3M</td>
<td>88.8</td>
<td>44.2</td>
<td>81.9</td>
<td>85.4</td>
<td>89.5</td>
<td>60.3</td>
<td>84.5</td>
<td>78.4</td>
<td>81.2/72.4</td>
<td>76.7</td>
</tr>
</tbody>
</table>
### 5.2 MHA: Dropping Heads or Reducing Dimension

Multi-Head Attention (MHA) allows the model to jointly attend to the information from different representation subspaces (Vaswani et al., 2017). When compressing the weights in MHA, there are two options, including 1) dropping heads, which reduces the number of heads $A$, and 2) reducing dimension, which reduces the size of each head $d_k$. For TinyBERT (Jiao et al., 2020) and MiniLM (Wang et al., 2020), they keep $A=12$ and reduce $d_k$ due to the constraint of attention-based loss. Our proposed WID is more flexible since we do not employ any alignment loss. Moreover, we can easily achieve these two strategies by constraining the column mask in MHA. For WID$_{55}$ and WID$_{11}$ reported in Table 2, we reduce the size of each attention head following TinyBERT for a fair comparison.

To further explore these two strategies, we conduct WID under these two settings and report the scores on downstream tasks. In BERT$_{base}$, we have $A=12$ and $d_k=64$. The student models are selected as: WID$_{55}^{dim}$ ($A=12$, $d_k=43$), WID$_{55}^{head}$ ($A=8$, $d_k=64$), WID$_{11}^{dim}$ ($A=12$, $d_k=16$), and WID$_{11}^{head}$ ($A=3$, $d_k=64$). As shown in Table 3, the dropping head strategy performs slightly worse under 55M parameters and much better under 11M parameters. For attention heads in WID$_{55}$, both 43 and 64 are large enough to encode the textual information in the representation subspace. Thus, the WID$_{55}^{dim}$ with more attention heads gets slightly better results. Similarly, the attention heads with size 16 perform worse due to the limited representation subspace, leading to the poor performance of WID$_{11}^{dim}$.

### 5.3 Impact of Teacher Models

To study the impact of teacher models, we compare the results of three teachers, including 1) BERT$_{base}$, 2) WID$_{55}^{head}$, which is compressed by BERT$_{base}$ adopting the dropping head strategy, 3) BERT$_{55}$, which shares the same architecture as WID$_{55}^{head}$. Both BERT$_{base}$ and BERT$_{55}$ are downloaded from the official repository. We compress these three teachers to WID$_{11}^{head}$ employing the dropping head strategy. Table 4 shows the results of three teachers. Some findings are summarized as follows:

1. A smaller teacher can also teach a smart student. Both BERT$_{base}$ and BERT$_{55}$ are pre-trained on the MLM tasks. But the student from BERT$_{55}$ gets an average score of 77.5, which is comparable to 77.6 from the student of BERT$_{base}$. A similar conclusion is also observed in Zhang et al. (2023).

2. An educated teacher teaches better. The WID$_{55}^{head}$ is compressed by BERT$_{base}$ adopting the dropping head strategy. Compared to BERT$_{55}$ under the same architecture, WID$_{55}^{head}$ can teach a better student on both GLUE benchmarks and the SQuAD task.

### 5.4 Looking into WID

We visualize the attention distributions between the teacher BERT$_{base}$ and the student WID$_{11}^{dim}$ with the same input tokens. For more comparison, we also pre-train BERT$_{11}$ from scratch which shares the same architecture as WID$_{11}^{dim}$. As shown in Figure

---

Table 3: Comparison between dropping heads and reducing dimension of each head for WID$_{55}$ with 55M parameters and WID$_{11}$ with 11M parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>SST-2</th>
<th>CoLA</th>
<th>MRPC</th>
<th>QNLI</th>
<th>QQP</th>
<th>RTE</th>
<th>STS-B</th>
<th>MNLI</th>
<th>SQuAD</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>WID$_{55}^{dim}$</td>
<td>92.4</td>
<td>61.7</td>
<td>88.2</td>
<td>90.1</td>
<td>91.0</td>
<td>70.4</td>
<td>87.9</td>
<td>82.9</td>
<td>88.5/80.8</td>
<td>83.4</td>
</tr>
<tr>
<td>WID$_{55}^{head}$</td>
<td>92.0</td>
<td>61.6</td>
<td>88.2</td>
<td>89.4</td>
<td>91.0</td>
<td>70.8</td>
<td>87.6</td>
<td>82.6</td>
<td>87.3/79.4</td>
<td>83.0</td>
</tr>
<tr>
<td>WID$_{11}^{dim}$</td>
<td>88.8</td>
<td>44.2</td>
<td>81.9</td>
<td>85.4</td>
<td>89.5</td>
<td>60.3</td>
<td>84.5</td>
<td>78.4</td>
<td>81.2/72.4</td>
<td>76.7</td>
</tr>
<tr>
<td>WID$_{11}^{head}$</td>
<td>89.6</td>
<td>46.2</td>
<td>83.1</td>
<td>86.1</td>
<td>89.5</td>
<td>62.1</td>
<td>85.3</td>
<td>79.0</td>
<td>81.7/72.9</td>
<td>77.6</td>
</tr>
</tbody>
</table>

Table 4: Comparison between different teacher models after they are compressed to WID$_{11}^{head}$. BERT$_{55}$ means the BERT model with same architecture as WID$_{55}^{head}$.

<table>
<thead>
<tr>
<th>Teacher</th>
<th>Params</th>
<th>SST-2</th>
<th>CoLA</th>
<th>MRPC</th>
<th>QNLI</th>
<th>QQP</th>
<th>RTE</th>
<th>STS-B</th>
<th>MNLI</th>
<th>SQuAD</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT$_{base}$</td>
<td>110.1M</td>
<td>89.6</td>
<td>46.2</td>
<td>83.1</td>
<td>86.1</td>
<td>89.5</td>
<td>62.1</td>
<td>85.3</td>
<td>79.0</td>
<td>81.7/72.9</td>
<td>77.6</td>
</tr>
<tr>
<td>BERT$_{55}$</td>
<td>54.2M</td>
<td>89.5</td>
<td>43.2</td>
<td>84.6</td>
<td>86.3</td>
<td>89.7</td>
<td>63.2</td>
<td>85.7</td>
<td>79.4</td>
<td>81.2/72.5</td>
<td>77.5</td>
</tr>
<tr>
<td>WID$_{55}^{head}$</td>
<td>54.2M</td>
<td>89.9</td>
<td>46.2</td>
<td>84.8</td>
<td>86.5</td>
<td>89.5</td>
<td>64.6</td>
<td>84.7</td>
<td>78.8</td>
<td>82.1/73.5</td>
<td>78.1</td>
</tr>
</tbody>
</table>

---

Footnote: 2https://github.com/google-research/bert
4. WID can learn the attention patterns in various layers of the teacher model BERT\textsubscript{base}, while BERT\textsubscript{11} can not. The results of more attention heads can be found in Appendix C.5.

In WID, we do not use any alignment loss between the teacher and the student. However, the compressed student model can still learn attention patterns. This indicates that inheriting the weights can also inherit high-level semantic knowledge.

6 Related Work

6.1 BERT Compression
Transformer-based Pre-trained Language Models (PLMs) can be compressed via Quantization (Stock et al., 2021; Tao et al., 2022), Matrix Decomposition (Mao et al., 2020), Pruning (Xia et al., 2022; Lagunas et al., 2021), and Knowledge Distillation (Jiao et al., 2020; Wang et al., 2020). We refer the readers to Ganesh et al. (2021) for a comprehensive survey. In this paper, we focus on KD for BERT compression.

6.2 Knowledge Distillation
KD aims to transfer the knowledge from the teacher model to the student model (Hinton et al., 2015; Wang et al., 2023; Wu et al., 2023). The distillation methods can be directly divided into three main categories: offline distillation, online distillation, and self-distillation (Gou et al., 2021). For PLMs, the majority of methods follow the offline distillation pattern where the teacher model is pre-trained before distillation. Meanwhile, distillation methods for PLMs can be divided into task-agnostic, which fine-tunes the teacher model on specific tasks and then distills.

In this work, we focus on the task-agnostic distillation. Previous methods mainly focus on designing extra matching losses for the student model to mimic the teacher model. These losses mainly include feature-based loss for features in intermediate layers and logit-based loss for output logits. DistilBERT (Sanh et al., 2019) adopts the output logit and embedding outputs of the teacher to train the student. TinyBERT (Jiao et al., 2020) and MobileBERT (Sun et al., 2020) further employ the self-attention distributions and hidden states for alignment loss. Such layer-to-layer distillation restricts the number of student layers or requires an extra mapping function. To address this issue, MiniLM (Wang et al., 2020) proposes a new loss based on the attention matrix and values-values scaled dot-product. WD (Lin et al., 2021) employs a similar idea to inherit the knowledge in parameters. However, WD initializes the weights of student models randomly and still requires alignment losses.

Different from existing methods, WID does not require additional alignment losses, thus avoiding laborious selection for both loss functions and loss weights.

7 Conclusion
This work proposes a novel Weight-Inherited Distillation (WID) method for task-agnostic BERT compression. In WID, we factorize the compression process as weight mappings, and then design the row compactors and column compactors for row mappings and column mappings, respectively. Empirical results on various student model sizes
demonstrate the effectiveness of WID. Further analysis indicates that inheriting the weights can also inherit high-level semantic knowledge such as attention patterns. In future work, we would consider reducing the extra memory cost by compactor layers, such as compactor sharing. Moreover, employing WID on the large language model (LLM) would be another interesting topic.

Acknowledgment

This work was supported by the Shenzen Science and Technology under Grant JSGG20220831110203007, and by the Theme-based Research Scheme (TRS) project T45-701/22-R, Hong Kong SAR.

Limitations

Our proposed WID inserts row/column compactors to learn the mappings from the teacher model to the student model. Thus, WID requires additional computational time and memory. However, WID still outperforms TinyBERT with fewer time costs. As shown in Table 7, WID trained with 100k steps achieves a higher score and saves more than 50% time costs compared to TinyBERT. However, we believe that such a trade-off is valuable since a faster and better compact student would save more time on downstream tasks.

References


Jiahao Wang, Songyang Zhang, Yong Liu, Taiqiang Wu, Yujiu Yang, Xihui Liu, Kai Chen, Ping Luo,


### A GLUE and SQuAD

#### A.1 Data Statistics

Table 5 shows the sizes of the train/development set and the metrics for downstream tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>#Train</th>
<th>#Dev</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>67k</td>
<td>872</td>
<td>Accuracy</td>
</tr>
<tr>
<td>QNLI</td>
<td>105k</td>
<td>5.5k</td>
<td>Accuracy</td>
</tr>
<tr>
<td>MNLI</td>
<td>393k</td>
<td>20k</td>
<td>Accuracy</td>
</tr>
<tr>
<td>QQP</td>
<td>364k</td>
<td>40k</td>
<td>Accuracy</td>
</tr>
<tr>
<td>CoLA</td>
<td>8.5k</td>
<td>1k</td>
<td>Matthews corr.</td>
</tr>
<tr>
<td>STS-B</td>
<td>7k</td>
<td>1.5k</td>
<td>Spearman corr.</td>
</tr>
<tr>
<td>MRPC</td>
<td>3.7k</td>
<td>408</td>
<td>Accuracy</td>
</tr>
<tr>
<td>SQuAD</td>
<td>87.6k</td>
<td>34.7k</td>
<td>F1 &amp; EM</td>
</tr>
</tbody>
</table>

Table 5: Data statistics of GLUE and SQuAD datasets.

#### A.2 Hyperparameters

We employ the grid search to fine-tune the GLUE benchmarks and SQuAD.

**GLUE** The learning rate is searched in \{1e-5, 2e-5, 3e-5\}. We set the search space for the training batch size based on the size of the training set. For large datasets including QNLI, MNLI, and QQP, the batch size is searched in \{32, 48\}. For small datasets including MRPC, RTE, CoLA, and STS-B, the batch size is searched in \{4, 6\}. For SST-2, the batch size is searched in \{8, 16\}. All tasks are trained for 10 epochs.

**SQuAD** The learning rate is searched in \{1e-5, 2e-5, 3e-5\} and batch size is searched in \{4, 6, 8\}. The training epochs are set to 3.

### B Method Details

#### B.1 Algorithm

More details about the proposed WID can be found in Algorithm 1.

#### B.2 Groups of Aligned Compactors

Specifically, we can divide all the compactors in BERT into the following aligned groups:

- One group in blue: \{CC for embedding layer, blue compactors in each Transformer layer, RC for output layer\},
- \(L\) groups in orange: \{orange compactors in layer 1\}; \{orange compactors in layer 2\}; ... \{orange compactors in layer \(L\)\},
- \(L\) groups in green: \{green compactors in layer 1\}; \{green compactors in layer 2\}; ... \{green compactors in layer \(L\)\},

Where RC/CC denotes the row/column compactor and \{\} denotes a group. For the only one group in blue, we calculate the column compactor for the embedding layer and duplicate (or, flip) it for the other compactors. For each group in orange, we calculate the column compactor for the Value projection and duplicate (or, flip) it for the rest three compactors. For each group in green, we calculate the column compactor for the Up-project and flip it for the other one.

### C Extensive Analysis

#### C.1 Comparison with Task-Specific Distillation

We also compare WID with task-specific distillation methods where the teacher model in task-specific distillation methods is fine-tuned for the task before distillation. For baselines, we select BERT-of-Theseus (Xu et al., 2020), DynaBERT (Hou et al., 2020) and MetaDistill(Zhou et al., 2022). As shown in Table 6, WID also outperforms these task-specific methods on the GLUE benchmarks.
### C.2 Comparison with Pruning

We try to compare WID with pruning methods for BERT compression, including task-specific CoFi (Xia et al., 2022) and BlockPruning (Li et al., 2020). As mentioned in Appendix C.1, the task-agnostic setting is more difficult than the task-specific setting. However, as shown in Table 6, WID still achieves comparable results with less than 50% parameters compared to CoFi, and achieves better performance than BlockPruning with 28.7% fewer parameters.

### C.3 Less Training Steps

In Table 2, we report the results of WID trained with less steps on GLUE benchmarks. We re-implement TinyBERT and train 3 epochs following the setting in Jiao et al. (2020). We reduce the training steps for WID to 50k and 100k. All experiments are carried out with 8 A100 GPUs. As shown in Table 7, WID trained with 100k steps can still outperform TinyBERT and save more than 50% training time.

### C.4 WID for GPT Compression

To evaluate the performance of WID on the generative pre-trained language model, we train a GPT model and compress it via vanilla KD and WID. Due to the limited GPU memory, we train a GPT teacher (12 layers and hidden size as 768) for 100k steps. After that, we train a student model (12 layers and hidden size as 512) and compress the teacher model into such a setting via vanilla KD and WID. During distillation, we employ BookCorpus as training datasets and report the training accuracy. For hyperparameters, the batch size is 64 and the learning rate is 1e-4. Figure 5 shows the training process. We can conclude that WID still works for generative pre-trained language models, and can get better performance than vanilla KD.

### C.5 Attention Distributions

We visualize the attention distributions for the teacher BERT base, pre-trained BERT 55 and the student WID 11 under the same input tokens (input sentence: "if the world harassed me, it will harass you too.") in Figure 6, Figure 7 and Figure 8, respectively. WID can effectively learn the attention patterns from the teacher model while BERT 11 is much more different.

---

**Table 6: Comparison among WID, task-specific distillation methods, and pruning methods on GLUE benchmarks without data augmentation. TS-KD and TA-KD denote task-specific knowledge distillation and task-agnostic knowledge distillation, respectively. * means the results are taken from Zhou et al. (2022). Other results are taken from the corresponding papers.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Params</th>
<th>SST-2</th>
<th>CoLA</th>
<th>MRPC</th>
<th>QNLI</th>
<th>QQP</th>
<th>RTE</th>
<th>STS-B</th>
<th>MNLI</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT base</td>
<td>Teacher</td>
<td>110.1M</td>
<td>92.7</td>
<td>59.1</td>
<td>90.4</td>
<td>91.7</td>
<td>91.4</td>
<td>70.8</td>
<td>90.1</td>
<td>84.5</td>
<td>83.8</td>
</tr>
<tr>
<td>DynaBERT</td>
<td>TS-KD</td>
<td>67.5M</td>
<td>92.7</td>
<td>54.6</td>
<td>86.8</td>
<td>90.4</td>
<td>91.0</td>
<td>69.4</td>
<td>89.1</td>
<td>83.8</td>
<td>82.7</td>
</tr>
<tr>
<td>MetaDistill</td>
<td>TS-KD</td>
<td>67.5M</td>
<td>92.3</td>
<td>58.6</td>
<td>86.6</td>
<td>90.6</td>
<td>90.6</td>
<td>67.7</td>
<td>88.7</td>
<td>83.8</td>
<td>81.4</td>
</tr>
<tr>
<td>TinyBERT*</td>
<td>TS-KD</td>
<td>67.5M</td>
<td>91.9</td>
<td>52.4</td>
<td>88.3</td>
<td>90.7</td>
<td>90.0</td>
<td>70.4</td>
<td>87.9</td>
<td>82.9</td>
<td>80.1</td>
</tr>
<tr>
<td>BlockPruning</td>
<td>Pruning</td>
<td>77.0M</td>
<td>92.4</td>
<td>51.7</td>
<td>86.1</td>
<td>90.4</td>
<td>90.6</td>
<td>67.7</td>
<td>88.7</td>
<td>83.8</td>
<td>83.4</td>
</tr>
<tr>
<td>WID 55 (ours)</td>
<td>TA-KD</td>
<td>54.9M</td>
<td>92.4</td>
<td>51.7</td>
<td>86.1</td>
<td>90.4</td>
<td>90.6</td>
<td>67.7</td>
<td>88.7</td>
<td>83.8</td>
<td>83.4</td>
</tr>
<tr>
<td>CoFi</td>
<td>Pruning</td>
<td>28.4M</td>
<td>90.6</td>
<td>35.6</td>
<td>82.6</td>
<td>86.1</td>
<td>90.1</td>
<td>64.7</td>
<td>87.9</td>
<td>82.9</td>
<td>80.6</td>
</tr>
<tr>
<td>WID 11 (ours)</td>
<td>TA-KD</td>
<td>11.3M</td>
<td>88.8</td>
<td>44.2</td>
<td>81.9</td>
<td>85.4</td>
<td>89.5</td>
<td>60.3</td>
<td>84.5</td>
<td>78.4</td>
<td>76.6</td>
</tr>
</tbody>
</table>

**Table 7: Comparison between TinyBERT and WID trained with less steps on GLUE benchmarks.**

<table>
<thead>
<tr>
<th>Model</th>
<th>Steps</th>
<th>Time</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyBERT (GD)</td>
<td>450k</td>
<td>33h</td>
<td>81.27</td>
</tr>
<tr>
<td>WID 55</td>
<td>50k</td>
<td>8h</td>
<td>80.78</td>
</tr>
<tr>
<td>WID 55</td>
<td>100k</td>
<td>16h</td>
<td>81.65</td>
</tr>
<tr>
<td>WID 55</td>
<td>400k</td>
<td>64h</td>
<td>83.08</td>
</tr>
</tbody>
</table>

---

![Figure 5](image-url)
Figure 6: The self-attention distributions for teacher model BERT$_{\text{base}}$. 
Figure 7: The self-attention distributions for BERT. 

27
Figure 8: The self-attention distributions for our proposed WID_{11}^{dim}.