Weight-Inherited Distillation for Task-Agnostic BERT Compression

Taiqiang Wu[1,2]*, **Cheng Hou**[2]*, **Shanshan Lao**[3],
Jiayi Li[3], **Ngai Wong**[1], **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.

Approach	Alignment Loss		Hard Loss	Task-Agnostic
	Logit	Feature		
DistilBERT	✓	✓	✓	✓
TinyBERT (GD)	✓	✓	✗	✓
PKD	✓	✓	✓	✗
MiniLM	✗	✓	✗	✓
MobileBERT	✓	✓	✓	✓
WID (ours)	✗	✗	✓	✓

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.

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 embedded 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*?

*Equal contributions. Work was done when Taiqiang and Cheng were interning at Tencent.

†Yujiu Yang and Zhe Zhao are the corresponding authors.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 13–28

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 ($\{\mathbf{x}_i\}_{i=1}^{|x|}$) 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 ($\{\mathbf{x}_i\}_{i=1}^{|x|}$) are packed to $\mathbf{H}^0 = [\mathbf{x}_1, \cdots, \mathbf{x}_{|x|}]$. After that, the L-layer transformer computes the encoding vectors following:

$$\mathbf{H}^l = \text{Transformer}_l(\mathbf{H}^{l-1}),\ l \in [1, L]. \quad (1)$$

The final output $\mathbf{H}^L = [h_1^L, \cdots, h_{|x|}^L] \in \mathbb{R}^{|x|\times d}$ is employed as the contextualized representation of $\{\mathbf{x}_i\}_{i=1}^{|x|}$. Each transformer layer consists of a multi-head self-attention (MHA) sub-layer and a feed-forward (FFN) sub-layer. In these two sub-layers, the residual connection (He et al., 2016) is employed, followed by Layer Normalization (LN) (Ba et al., 2016).

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

$$\begin{aligned}\mathbf{Q}_{l,a} &= \mathbf{H}^{l-1}\mathbf{W}_{l,a}^Q \\ \mathbf{K}_{l,a} &= \mathbf{H}^{l-1}\mathbf{W}_{l,a}^K \\ \mathbf{V}_{l,a} &= \mathbf{H}^{l-1}\mathbf{W}_{l,a}^V\end{aligned} \quad (2)$$

$$\mathbf{O}_{l,a} = \mathbf{A}_{l,a}\mathbf{V}_{l,a}, \mathbf{A}_{l,a} = \text{softmax}(\frac{\mathbf{Q}_{l,a}\mathbf{K}_{l,a}^T}{\sqrt{d_k}}) \quad (3)$$

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

$$\mathbf{O}_l = \text{LN}(\mathbf{H}^{l-1} + (||_{a=1}^A \mathbf{O}_{l,a})\mathbf{W}_l^O) \quad (4)$$

where $\mathbf{W}_l^O \in \mathbb{R}^{d\times d}$, LN is layer normalization and $||$ denotes the concatenation operation.

FFN The l-th FFN sub-layer consists of an up projection and a down projection, parameterized by $\mathbf{W}_l^U \in \mathbb{R}^{d\times d_f}$, $\mathbf{W}_l^D \in \mathbb{R}^{d_f\times d}$, and corresponding bias $\mathbf{b}_l^U \in \mathbb{R}^{d_f}$, $\mathbf{b}_l^D \in \mathbb{R}^d$:

$$\text{FFN}(\mathbf{O}_l) = \text{gelu}(\mathbf{O}_l\mathbf{W}_l^U + \mathbf{b}_l^u)\mathbf{W}_l^D + \mathbf{b}_l^d. \quad (5)$$

Typically, $d_f = 4d$. Finally, we obtain the output of layer l by:

$$\mathbf{H}^l = \text{LN}(\mathbf{O}_l + \text{FFN}(\mathbf{O}_l)). \quad (6)$$

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 $\mathbf{p}_s$ from the student and $\mathbf{p}_t$ from the teacher, which can be formalized as:

$$\mathcal{L}_{logit} = \mathcal{H}_1(\mathbf{p}_s/\tau, \mathbf{p}_t/\tau), \tag{7}$$

where τ 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:

$$\mathcal{L}_{feature} = \mathcal{H}_2(f^S(x), f^T(x)), \tag{8}$$

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:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{hard} + \alpha\mathcal{L}_{logit} + \beta\mathcal{L}_{feature}, \tag{9}$$

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

Meanwhile, selecting various loss functions is also laborious. In PKD, $\mathcal{L}_{feature}$ 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:

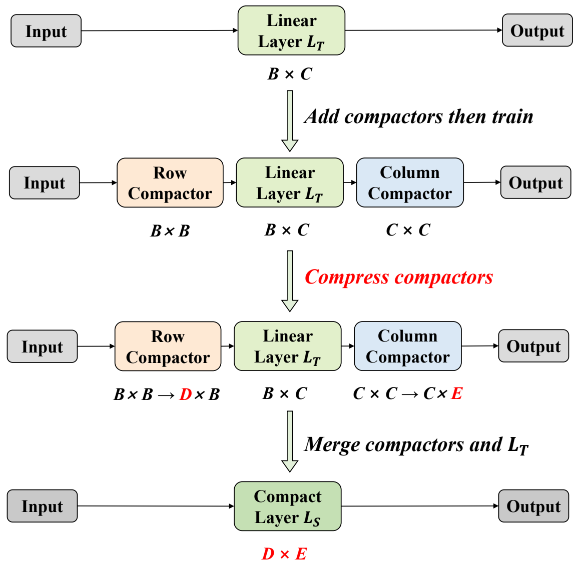

Figure 1: Overview of compressing linear layer L_T with weight $\mathbf{W}^{L_T} \in \mathbb{R}^{B\times C}$ to compact linear layer L_S with weight $\mathbf{W}^{L_S} \in \mathbb{R}^{D\times E}$ via WID. Both row compactor and column compactor are initialized as **identity matrices**. After training, we compress the compactors and merge them with the original layer. All the linear layers in the teacher model are compressed **simultaneously**.

- column mapping only, such as the token embedding matrix $W_T \in \mathbb{R}^{|V|\times d}$,
- row mapping only, such as the weight of output layer for MLM task with size $\mathbb{R}^{d\times |V|}$,
- column and row mapping, such as up projection $\mathbf{W}_{l,u} \in \mathbb{R}^{d\times d_f}$ in FFN.

In WID, we adopt the re-parameterization trick and design the row compactor for row mapping and column compactor for column mapping, respectively.

Figure 1 gives an example showing the process of compressing the original weight $\mathbf{W}^{L_T} \in \mathbb{R}^{B\times C}$ to a compact weight $\mathbf{W}^{L_S} \in \mathbb{R}^{D\times E}$ adopting both row compactor and column compactor. First, we insert the row compactor with weight $\mathbf{W}^{rc} \in \mathbb{R}^{B\times B}$ and the column compactor with weight $\mathbf{W}^{cc} \in \mathbb{R}^{C\times C}$ before and after the linear layer L_T from the teacher model. All compactors are linear layers without bias and their weights are initialized as identity matrices. For an arbitrary input X, the re-parameterized teacher model produces identical outputs as the original, since

$$X\mathbf{W}^{L_T} = X\mathbf{W}^{rc}\mathbf{W}^{L_T}\mathbf{W}^{cc}. \tag{10}$$

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

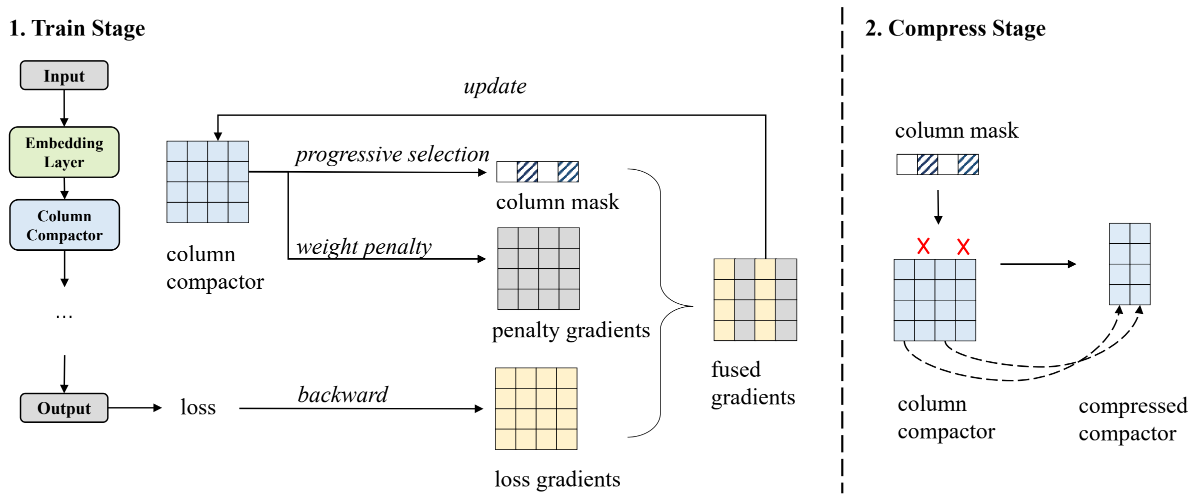

Figure 2: Training and compression for column compactor. During the training process, we add weight penalty gradients by columns and progressively select the mask to fuse the penalty gradients and original loss gradients. After training, we compress the column compactor following the column mask.

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:

$$\begin{aligned} \mathbf{W}^{rc} \in \mathbb{R}^{B\times B} \rightarrow \mathbf{W}^{rc'} \in \mathbb{R}^{D\times B} \\ \mathbf{W}^{cc} \in \mathbb{R}^{C\times C} \rightarrow \mathbf{W}^{cc'} \in \mathbb{R}^{C\times E}. \end{aligned} \tag{11}$$

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

$$\mathbf{W}^{L_S} = \mathbf{W}^{rc'}\mathbf{W}^{L_T}\mathbf{W}^{cc'} \in \mathbb{R}^{D\times E} \tag{12}$$

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 $\mathbf{W}^{cc} \in \mathbb{R}^{C\times C}$ 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:

$$g^{cc}_{pen} = \frac{\mathbf{W}^{cc}}{||\mathbf{W}^{cc}||_2} \tag{13}$$

where $||\mathbf{W}^{cc}||_2$ 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 $||\mathbf{W}^{cc}||_2$ 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:

$$g^{cc}_{fused}[:, i] = \begin{cases} g^{cc}_{pen}[:, i], & \text{if } M[i] = 1 \\ g^{cc}_{ori}[:, i], & \text{if } M[i] = 0 \end{cases} \tag{14}$$

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

$$\mathbf{W}^{cc'} = \mathbf{W}^{cc}[:, i], \text{ where } M[i] = 0. \tag{15}$$

Moreover, the process is similar for row compactors. We calculate $||\mathbf{W}^{rc}||_2$ 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.

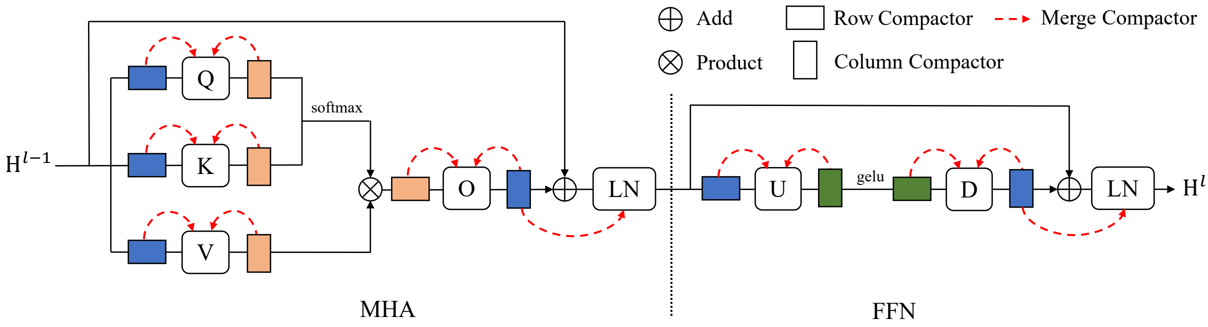

Figure 3: Compactor merging process for a Transformer block. For the bias terms, we merge them with corresponding column compactors. For beta and gamma in Layer Norm (LN), we adopt the previous column compactors to update them. During training, the compactors in the **same color** are aligned. For each group of the aligned compactors, we learn one of them and duplicate (or, flip) it for the rest compactors.

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 $\mathbf{H}^{l-1}$ and the first compactor after the $\mathbf{H}^{l}$ 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 $\text{BERT}_{\text{base}}$ as our teacher model [1] and implement WID based on TencentPretrain framework(Zhao et al., 2023). $\text{BERT}_{\text{base}}$ (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

[1] From https://huggingface.co/bert-base-uncased

Method	FLOPs	Params	SST-2	CoLA	MRPC	QNLI	QQP	RTE	STS-B	MNLI	SQuAD	AVG
$\text{BERT}_{\text{base}}$	22.7B	110.1M	92.7	59.1	90.4	91.7	91.4	70.8	90.1	84.5	89.6/82.6	84.3
DistilBERT	11.9B	67.5M	91.3	51.3	87.5	89.2	88.5	59.9	86.9	82.2	86.2/78.1	80.1
MiniLM	11.9B	67.5M	92.0	49.2	88.4	91.0	91.0	71.5	-	84.0	-/-	-
MiniLM v2	11.9B	67.5M	92.4	52.5	88.9	90.8	91.1	72.1	-	84.2	-/-	-
TinyBERT (GD)†	11.9B	67.5M	92.9	44.1	89.5	90.7	91.0	73.7	89.6	83.8	84.0/74.2	81.3
TinyBERT (GD)‡	10.4B	54.9M	92.3	47.0	87.3	90.8	90.9	69.7	89.0	83.3	85.4/76.2	81.2
WID_{55} (ours)	10.4B	54.9M	92.4	61.7	88.2	90.1	91.0	70.4	87.9	82.9	88.5/80.8	**83.4**
TinyBERT (GD)‡	1.6B	11.3M	88.4	30.3	80.4	87.5	89.1	65.3	84.0	79.4	80.5/70.7	75.6
WID_{11} (ours)	1.6B	11.3M	88.8	44.2	81.9	85.4	89.5	60.3	84.5	78.4	81.2/72.4	**76.7**

Table 2: Comparison between our WID and various task-agnostic distillation methods. We compare the task-agnostic distilled models without both data augmentation and task-specific distillation. † means that we fine-tune the official weights. ‡ means that we reproduce the methods following the official code. Other results are taken from corresponding papers. For MiniLM and MiniLM v2, the average reported scores are 81.0 and 81.7, and both are lower than the 82.3 of WID.

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 $\text{IB-BERT}_{\text{large}}$ (much higher accuracy than $\text{BERT}_{\text{base}}$) 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 $\text{BERT}_{\text{base}}$ as the teacher model. As shown in Table 2, WID retains 98.9% and 90.9% performance of $\text{BERT}_{\text{base}}$ using only 49.2% and 10.2% parameters, respectively. In particular, in the CoLA task, WID_{55} gets a higher score than $\text{BERT}_{\text{base}}$. Compared to the baselines with 67.5M parameters, WID_{55} 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_{55} 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.

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

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

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

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}.

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 $\text{BERT}_{\text{base}}$, 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) $\text{BERT}_{\text{base}}$, 2) WID_{55}^{head}, which is compressed by $\text{BERT}_{\text{base}}$ adopting the dropping head strategy, 3) BERT_{55}, which shares the same architecture as WID_{55}^{head}. Both $\text{BERT}_{\text{base}}$ and BERT_{55} are downloaded from the official repository [2]. 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 $\text{BERT}_{\text{base}}$ 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 $\text{BERT}_{\text{base}}$. A similar conclusion is also observed in Zhang et al. (2023).

(2) An educated teacher teaches better. The WID_{55}^{head} is compressed by $\text{BERT}_{\text{base}}$ 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 $\text{BERT}_{\text{base}}$ 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

[2]https://github.com/google-research/bert

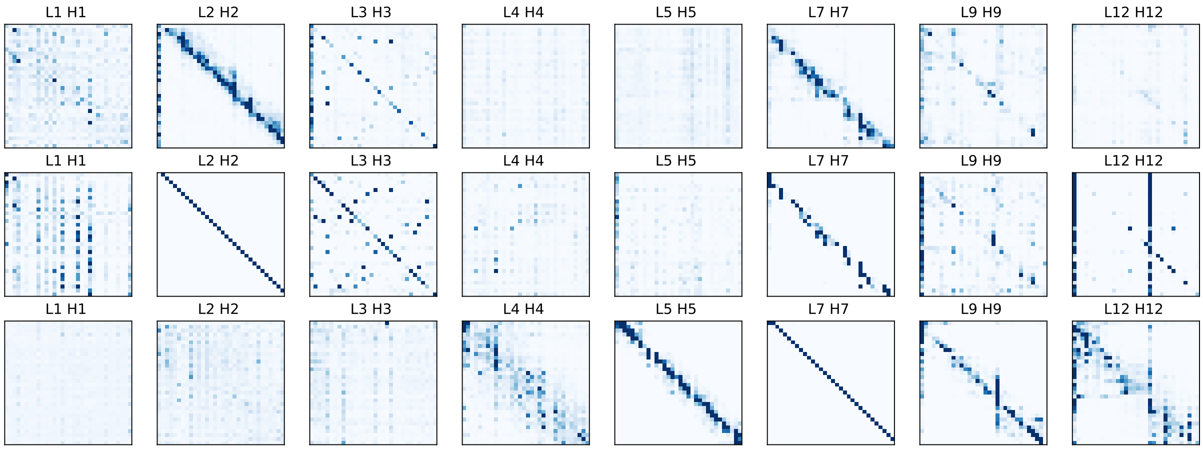

Figure 4: Attention distributions under same input tokens for $\text{BERT}_{\text{base}}$ (upper), WID_{11}^{dim} (middle), and BERT_{11} (bottom). Our WID can learn the knowledge about attention distributions from the teacher without any alignment loss.

4, WID can learn the attention patterns in various layers of the teacher model $\text{BERT}_{\text{base}}$, while BERT_{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 distills the PLM in pre-training stage, and task-specific, 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.

Acknowledge

This work was supported by the Shenzhen 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_{55}^{dim} 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

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14. Association for Computational Linguistics.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. 2021. Resrep: Lossless CNN pruning via decoupling remembering and forgetting. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 4490–4500. IEEE.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*. Asian Federation of Natural Language Processing.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2021. Compressing large-scale transformer-based models: A case study on BERT. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic BERT with adaptive width and depth. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M. Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10619–10629. Association for Computational Linguistics.

Yann LeCun, John S. Denker, and Sara A. Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann.

Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. 2020. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3187–3199. Association for Computational Linguistics.

Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. 2021. Weight distillation: Transferring the knowledge in neural network parameters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2076–2088. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of BERT through hybrid model compression. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3225–3234. International Committee on Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. 2021. Training with quantization noise for extreme model compression. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4821–4836. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jiahao Wang, Songyang Zhang, Yong Liu, Taiqiang Wu, Yujiu Yang, Xihui Liu, Kai Chen, Ping Luo, and Dahua Lin. 2023. Riformer: Keep your vision backbone effective but removing token mixer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. Minilmv2: Multi-head self-attention relation distillation for compressing pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2140–2151. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Taiqiang Wu, Zhe Zhao, Jiahao Wang, Xingyu Bai, Lei Wang, Ngai Wong, and Yujiu Yang. 2023. Edge-free but structure-aware: Prototype-guided knowledge distillation from gnns to mlps. *arXiv preprint arXiv:2303.13763*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1513–1528. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7859–7869. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Chen Zhang, Yang Yang, Jiahao Liu, Jingang Wang, Yunsen Xian, Benyou Wang, and Dawei Song. 2023. Lifting the curse of capacity gap in distilling language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4535–4553. Association for Computational Linguistics.

Zhe Zhao, Yudong Li, Cheng Hou, Jing Zhao, Rong Tian, Weijie Liu, Yiren Chen, Ningyuan Sun, Haoyan Liu, Weiquan Mao, et al. 2023. Tencentpretrain: A scalable and flexible toolkit for pre-training models of different modalities. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 217–225.

Wangchunshu Zhou, Canwen Xu, and Julian J. McAuley. 2022. BERT learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7037–7049. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A GLUE and SQuAD

A.1 Data Statistics

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

Task	#Train	#Dev	Metric
SST-2	67k	872	Accuracy
QNLI	105k	5.5k	Accuracy
MNLI	393k	20k	Accuracy
QQP	364k	40k	Accuracy
CoLA	8.5k	1k	Matthews corr.
RTE	2.5k	276	Accuracy
STS-B	7k	1.5k	Spearman corr.
MRPC	3.7k	408	Accuracy
SQuAD	87.6k	34.7k	F1 & EM

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.

Algorithm 1 Weight-Inherited Distillation

Input: teacher model $\mathcal{T}$ with width d_t
Params: k: number of rows/columns to compress, N: steps to increase k, d: increment for k each time
Output: student model $\mathcal{S}$ with width d_s

Add compactors for $\mathcal{T}$ to construct the re-parameterized teacher model $\hat{\mathcal{T}}$. Initialize the weights for compactors as identity matrices.
$k \leftarrow 0$; $M \leftarrow [\,]$
for $i = 0$ to max training steps **do**
  Forward a batch through $\hat{\mathcal{T}}$, derive the gradients g_{ori} for compactors to update
  if $i\%N == 0$ & $k < d_t - d_s$ **then**
    Calculate p-norm values
    Select the top-k row/column with the lower norm to get M
    Get penalty gradients g_{pen} following Eq. 13
    $g_{fused} \leftarrow f(g_{ori}, g_{pen}, M)$ following Eq. 14
    $k \leftarrow k + d$
  end if
  Update the compactors with corresponding g_{fused} and original layers with g_{ori}
  Apply the compactor aligning strategy
end for
Compress the compactors following Eq. 15
Merge the compactors and original layers following Eq. 12 to get compact layers for $\mathcal{S}$
return $\mathcal{S}$

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 $\{\cdot\}$ 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.

Method	Type	Params	SST-2	CoLA	MRPC	QNLI	QQP	RTE	STS-B	MNLI	AVG
$BERT_{base}$	Teacher	110.1M	92.7	59.1	90.4	91.7	91.4	70.8	90.1	84.5	83.8
DynaBERT	TS-KD	67.5M	92.7	54.6	85.0	90.6	91.1	66.1	88.6	83.7	81.6
MetaDistill	TS-KD	67.5M	92.3	58.6	86.8	90.4	91.0	69.4	89.1	83.8	82.7
TinyBERT*	TS-KD	67.5M	91.9	52.4	86.5	89.8	90.6	67.7	88.7	83.8	81.4
BlockPruning	Pruning	77.0M	89.3	52.6	88.3	88.2	90.7	63.9	84.6	82.9	80.1
WID_{55} (ours)	TA-KD	54.9M	92.4	61.7	88.2	90.1	91.0	70.4	87.9	82.9	83.4
CoFi	Pruning	28.4M	90.6	35.6	82.6	86.1	90.1	64.7	83.1	80.6	76.6
WID_{11} (ours)	TA-KD	11.3M	88.8	44.2	81.9	85.4	89.5	60.3	84.5	78.4	76.6

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.

C.2 Comparison with Pruning

We try to compare WID with pruning methods for BERT compression, including task-specific CoFi (Coarse- and Fine-grained Pruning,(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_{55}^{dim} trained for 400k steps. We re-implement TinyBERT and train 3 epochs following the setting in Jiao et al. (2020). We reduce the training steps for WID_{55}^{dim} to 50k and 100k. All experiments are carried out with 8 A100 GPUs. As shown in Table 7, WID_{55}^{dim} trained with 100k steps can still outperform TinyBERT and save more than 50% training time.

Model	Steps	Time	Score
TinyBERT (GD)	450k	33h	81.27
WID_{55}^{dim}	50k	8h	80.78
WID_{55}^{dim}	100k	16h	81.65
WID_{55}^{dim}	400k	64h	83.08

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

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.

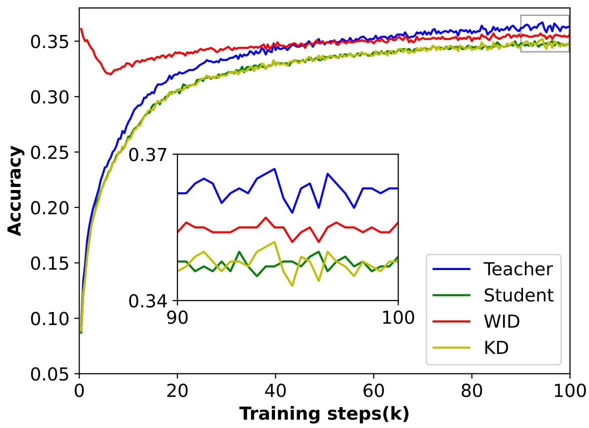

Figure 5: The training process for teacher GPT, vanilla student GPT, and students via KD and WID.

C.5 Attention Distributions

We visualize the attention distributions for the teacher $BERT_{base}$, pre-trained $BERT_{55}$ and the student WID_{11}^{head} 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.

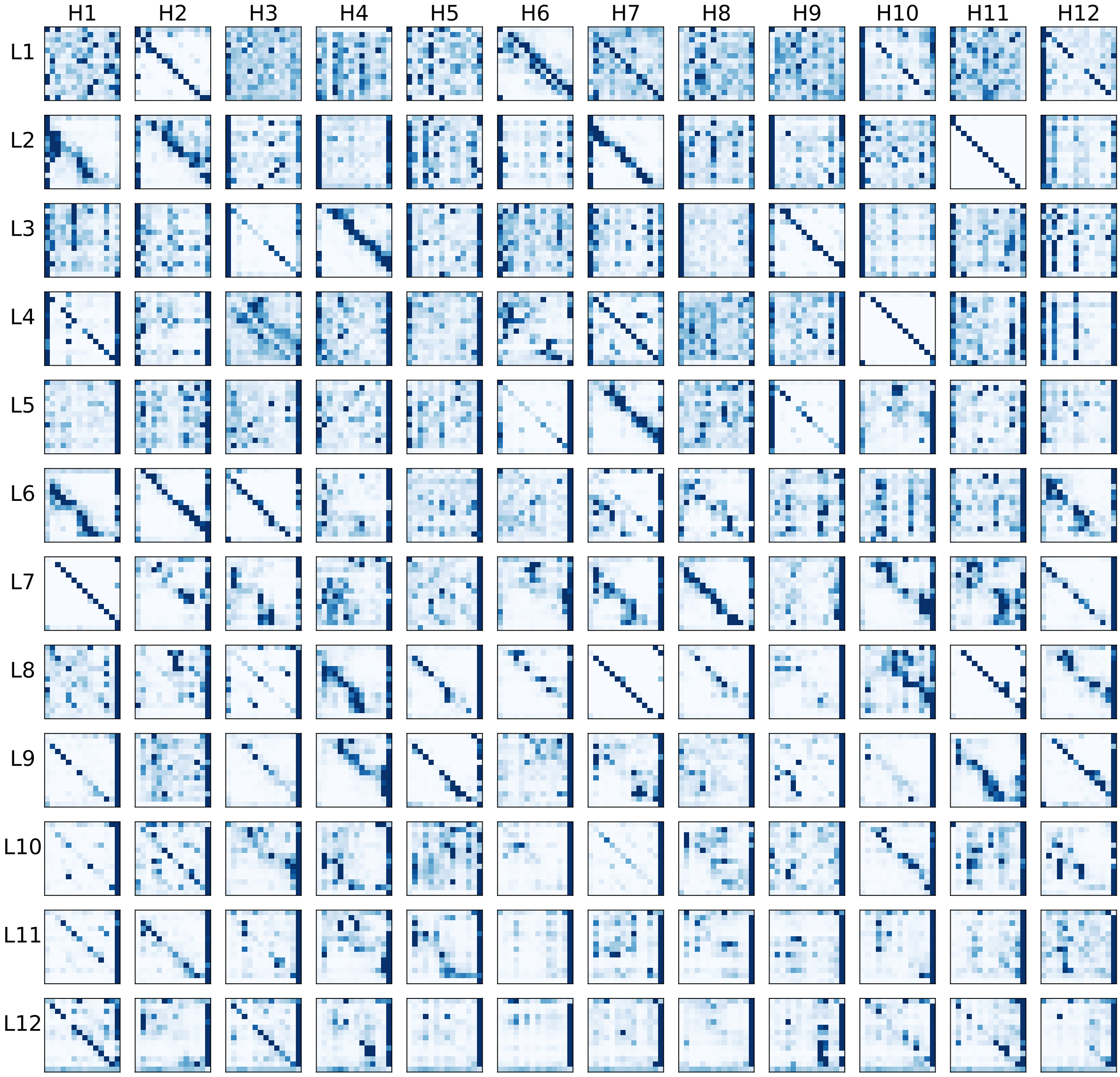

Figure 6: The self-attention distributions for teacher model $\text{BERT}_{\text{base}}$.

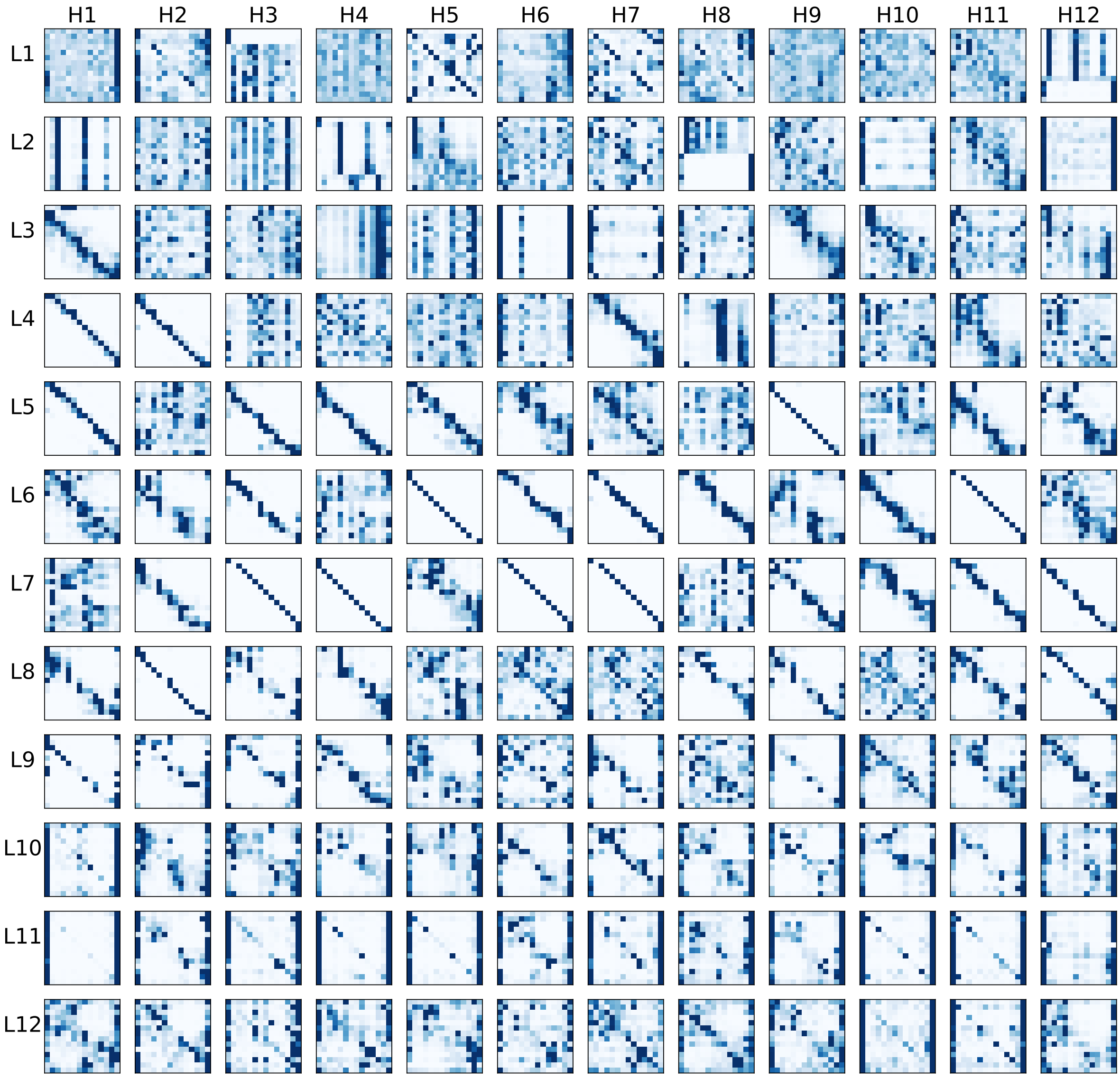

Figure 7: The self-attention distributions for BERT_{11}.

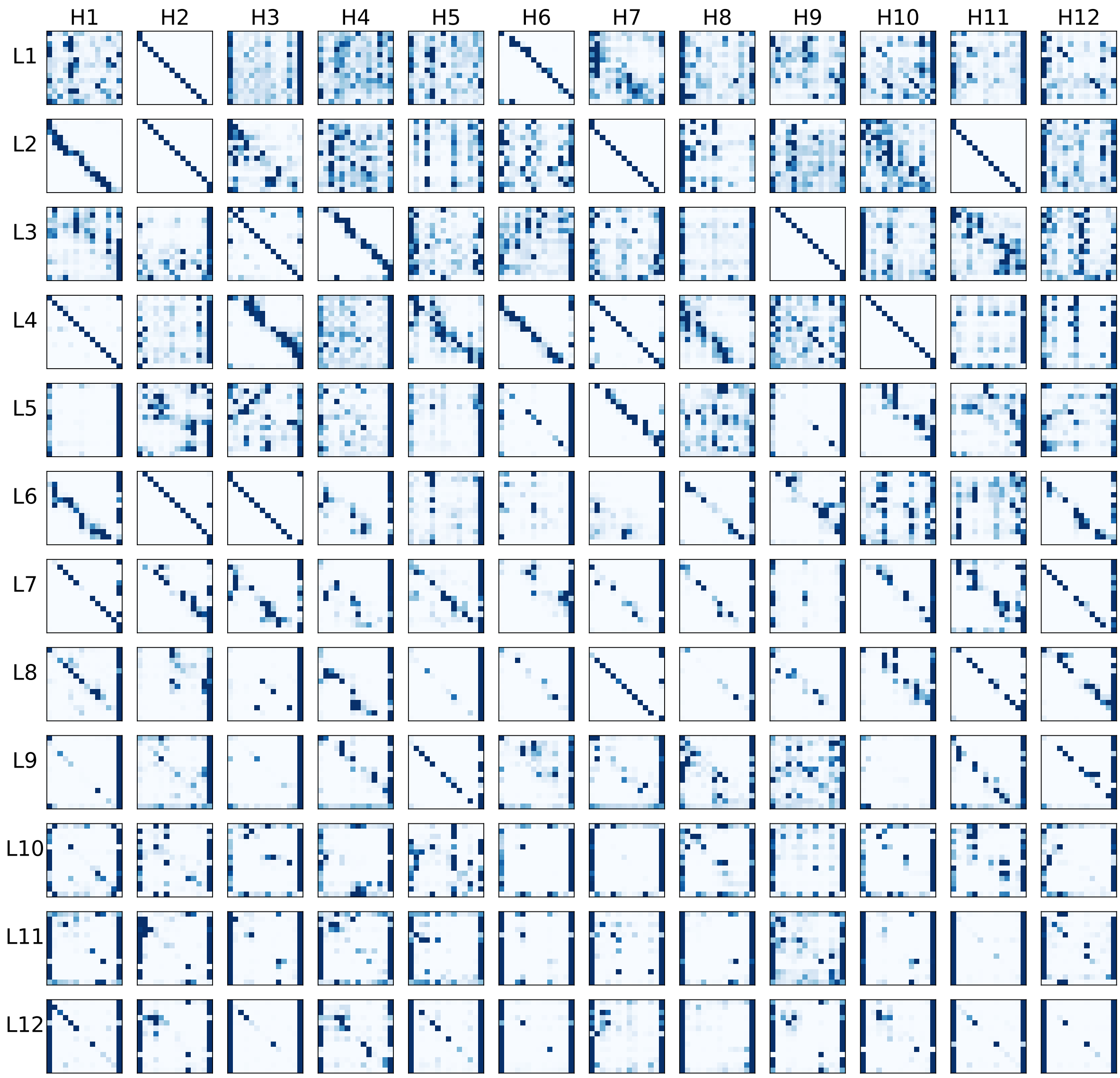

Figure 8: The self-attention distributions for our proposed WID_{11}^{dim}.