applied sciences

MDPI

Article

F-ALBERT: A Distilled Model from a Two-Time Distillation System for Reduced Computational Complexity in ALBERT Model

Kyeong-Hwan Kim and Chang-Sung Jeong *

Department of Electrical Engineering, Korea University, Seoul 02841, Republic of Korea; kyunghwan@korea.ac.kr
* Correspondence: csjeong@korea.ac.kr

Abstract: Recently, language models based on the Transformer architecture have been predominantly used in AI natural language processing. These models, which have been proven to perform better with more parameters, have led to a significant increase in model size and computational load. ALBERT solves this problem by significantly reducing the number of parameters it retains by repeatedly reusing parameters. Although ALBERT significantly reduces the parameters it maintains, it requires a computational load similar to the original language model due to the reuse process. In this study, we develop a distillation system that decreases the number of times the ALBERT model reuses parameters and progressively reduces the parameters being reused. We propose a representation in this distillation system that can effectively distill the knowledge of the original model and develop a new architecture with reduced computation. Through this system, F-ALBERT, which had about half the computational load compared to the ALBERT model, restored about 98% of the performance of the original model on the GLUE benchmark.

Keywords: natural language processing; Knowledge Distillation; model compression; transformers

check for updates

Citation: Kim, K.-H.; Jeong, C.-S. F-ALBERT: A Distilled Model from a Two-Time Distillation System for Reduced Computational Complexity in ALBERT Model. *Appl. Sci.* **2023**, *13*, 9530. https://doi.org/10.3390/app13179530

Academic Editor: Christos Bouras

Received: 18 July 2023
Revised: 14 August 2023
Accepted: 21 August 2023
Published: 23 August 2023

1. Introduction

Recent natural language processing (NLP) research has focused on Transformer-based AI language models, which learn various contexts through unsupervised learning on extensive text data [1–3]. These models can be broadly categorized into classification and generation types, with BERT being a prominent classification model, known for its high performance in context-sensitive tasks through bidirectional encoding [4]. However, this bidirectional encoding increases complexity, leading to longer training and inference times. Additionally, models like BERT have shown improved performance with more parameters, leading to gradual increases in model size. Thus, research is being conducted to reduce both the number of parameters and the computational complexity of such models [5–8].

ALBERT (A Lite BERT) [9] is a transformer-based deep learning model used for natural language processing tasks. Compared to the traditional BERT model, ALBERT employs parameter-sharing and factorization methods to reduce the model size while maintaining performance, resulting in enhanced training speed and memory efficiency. However, despite significantly reducing the number of parameters, ALBERT maintains a similar level of computational complexity as the original BERT model. When the number of ALBERT reuse times matches the number of identical layer modules in the BERT structure, similar performance is observed on the GLUE benchmark. Consequently, the learning and inference speeds are the same as those of BERT. In light of this background, this research explores a novel approach to further enhance the efficiency and speed of ALBERT.

We propose the F-ALBERT Distillation System, designed to decrease the parameter reuse times in ALBERT and progressively reduce the number of parameters utilized as reuse times increase. This aims to enhance the efficiency of parameter utilization and minimize the model size by employing Knowledge Distillation [10], a technique focused on distilling a more complex teacher model into a more lightweight student model.

The F-ALBERT Distillation System we propose implements this by utilizing a two-step distillation process: Layer-Reduction ALBERT Distillation (LR-ALBERT Distillation) and Iterative Weight Pruning ALBERT Distillation (IP-ALBERT Distillation). The first step, LR-ALBERT Distillation, distills the teacher model into a student model with fewer layer parameter reuse times. Subsequently, the model undergoes further distillation into the newly proposed structure, IP-ALBERT, through IP-ALBERT Distillation. This IP-ALBERT model applies the Iterative Weight Pruning technique, which progressively decreases the parameters used in computation as parameter reuse iterates, reducing their number over time. F-ALBERT is the result of the system distilling ALBERT twice.

The research presented herein primarily encompasses two main categories of contribution. The first involves the development of a speed-optimization technique for models rooted in Knowledge Distillation. A knowledge representation method for distilling ALBERT is introduced, accompanied by a strategy for effective knowledge transfer through LR-Distillation and IP-Distillation. The second main area of contribution pertains to the proposition of an Iterative Weight Pruning technique. This technique strategically prunes parameters engaged in computation as parameter reuse progresses. A Partitioned Transition Module is proposed as part of this approach, effectively replacing the existing Transition Module within a layer. The Iterative Weight Pruning technique is implemented through the proposed IP-ALBERT. This optimized and distilled model, F-ALBERT, remarkably cuts down the computation load by about half compared to the existing ALBERT, yet it manages to restore approximately 98% of its performance.

The composition of our manuscript is as follows. In Section 2, we elucidate the computational process and structure of the existing BERT model, explaining how ALBERT reuses this architecture to aid readers' understanding. Additionally, we present the context and direction of previous research, clearly defining our research position and contributions. In Section 3, we introduce our specific distillation techniques, focusing on the system and the system components that distill ALBERT into F-ALBERT. The system is organized into two main distillation procedures, and we detail the distillation models and techniques for each step. In Section 4, we conduct experiments by distilling the model under various configurations and comprehensively evaluating the distilled models. Section 5 concludes the paper by highlighting the achievements and outlining future research directions.

2. Related Works

2.1. Pretrained Language Models

The Transformer technology has brought about revolutionary developments in natural language processing by introducing a Self-Attention mechanism to capture relationships between words within a sentence [11]. This technology consists of encoders and decoders, demonstrating outstanding performance in various natural language processing tasks.

Models such as BERT [4] and GPT [12] utilize encoders and decoders independently for natural language understanding (NLU) and natural language generation tasks, respectively. While BERT offers bidirectional context, excelling in tasks like sentence classification, entity recognition, and question answering, GPT performs well in unidirectional context tasks like sentence completion, machine translation, and summarization.

With the advancement in computational capability and the growing demand for complex natural language processing performance, language models have progressively increased in size [13]. They leverage diverse datasets and numerous parameters to learn intricate patterns and knowledge. However, the expansion in the development costs and services has led to various studies focusing on reducing the number of parameters and computational load in the model. This study explores the possibility of lightweight language models through a lightweight and high-speed BERT model, aiming to minimize the decrease in the GLUE benchmark score while achieving a high level of lightweighting.

2.2. BERT Architecture

2.2.1. Embedding Layer Module

BERT begins by receiving a sequence of integer tokens corresponding to words as input. The Embedding Layer converts these sequences of integer tokens into vectors using various embedding dictionaries.

2.2.2. Transformer Layer Module

- Self-Attention Module (S): The Self-Attention mechanism allows the model to consider other words in the context when encoding a particular word. The mathematical details are as follows: Given an input sequence of tokens, we have embeddings Q(queries), K(keys), and V(values), which are obtained by multiplying the input embedding matrix by weight matrices W^Q, W^K, and W^V, respectively.

  $$S(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

  where d_k is the dimensionality of the key vectors.
  The Self-Attention values are a weighted sum of V, where the weights are determined by the softmax scores.
- Transition Module (T): The module transforms the word vector representations x received from the Self-Attention Module into non-linear high-dimensional vectors. The Transition Module consists of two linear transformations with a ReLU activation in between:

  $$T(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

  where W_1, W_2 are weight matrices, and b_1, b_2 are bias vectors.

BERT is a structure in which the modules presented above are composed in the following way.

$$O_l = T_l(S_l(O_{l-1})),\ l = 1,2,\ldots,L$$

$$O_0 = embedding\ vectors$$

2.3. ALBERT

ALBERT is a model that has successfully reduced the parameter usage of BERT. ALBERT (A Lite BERT) comprises a Transformer layer module with the same structure as BERT. However, ALBERT has two significant differences from BERT. The first is that it reduces the size of the embedding parameters through weight factorization, making the model smaller. The second is the parameter reuse technique, which shares weights between Transformer layers. Considering parameter reuse as the focus, ALBERT can be expressed as follows.

$$O_l = T_{shared}(S_{shared}(O_{l-1})),\ l = 1,2,\ldots,L$$

$$O_0 = embedding\ vectors$$

The term “shared” refers to the use of the same parameters across all layers. This is a crucial technique for reducing the total number of parameters ALBERT maintains. Sharing weights between layers is equivalent to reusing the Transformer layers repetitively.

2.4. Advances in Lightweight Transformer Models

Recently, as the parameters of Transformer models have become increasingly large [14], research has been conducted to make them more lightweight [15–17]. In this context, we briefly examine the latest research trends and our unique approaches to the subject:

1. **Optimization of Intermediate Layer Feature Transfer:**
   Traditional research has focused on the student model transmitting features from a limited number of Transformer layers, omitting intermediate layer features. This facilitates rapid learning through learning intermediate representations, but information

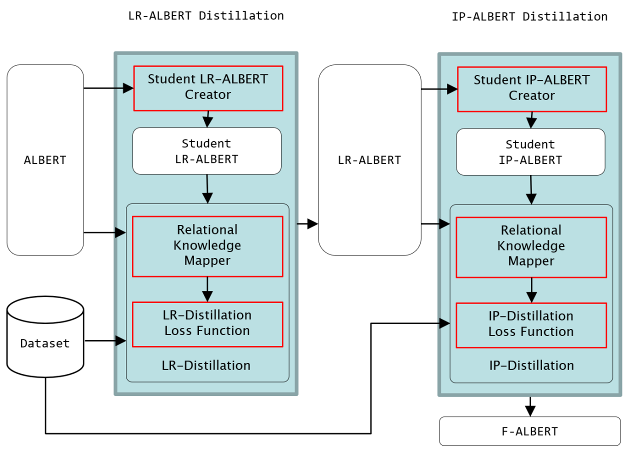
loss has been a problem [18,19]. A solution to this was proposed by compressing the teacher model's intermediate features and delivering them to the student model [20]. Our research focuses on developing this approach by minimizing information loss between teacher and student Transformer layers.

2. **Feature Representations** Transferring features in a processed rather than simple form can lead to more effective results [8,21–23]. Examples include transformation methods such as cosine representation or Euclidean distance. We have explored the effect of feature representations by leveraging these techniques.
3. **Combination of Weight Pruning and Knowledge Distillation:** Weight Pruning is effective for model compression but may lead to performance loss. Existing research has considered the parts pruned and has studied techniques to focus on distilling the pruned parts [24,25]. We extend this technique by applying it to the parameter-reusing technique.
4. **Extension of Mixture-of-Experts:** The method of dividing the high-dimensional Transition Module into several parts using the Mixture-of-Experts significantly reduces parameter retention and usage, but there are problems of performance degradation due to dimension reduction [26,27]. We have researched a way to improve this by varying the number of divided modules per layer, configuring some layers as high-dimensional and the rest as low-dimensional, ensuring both parameter efficiency and performance.

3. Methods

3.1. System Architecture

The F-ALBERT Distillation System performs the process of distilling ALBERT into F-ALBERT. The system carries out a two-step distillation process. The first distillation process is the LR-ALBERT Distillation, which distills the ALBERT model into LR-ALBERT. This process consists of two components: Student LR-ALBERT Creator and LR-Distillation. The Student LR-ALBERT Creator generates and initializes the Student LR-ALBERT using ALBERT. LR-Distillation distills the output results of ALBERT for all dataset samples into the Student LR-ALBERT. LR-ALBERT is the result of the first distillation process. LR-ALBERT is structurally identical to ALBERT, but it is a model with fewer layers reused. The second distillation process, IP-ALBERT Distillation, distills LR-ALBERT, which results from the first distillation process, into Student IP-ALBERT. This process consists of two components: Student IP-ALBERT Creator and IP-Distillation. The Student IP-ALBERT Creator creates and initializes the Student IP-ALBERT using LR-ALBERT. IP-Distillation distills the output results of LR-ALBERT for all given data samples into the Student IP-ALBERT. F-ALBERT is the final Student IP-ALBERT model after it goes through the distillation process. Figure 1 provides a summary of the entire process.



LR-ALBERT Distillation
IP-ALBERT Distillation
ALBERT
Student LR–ALBERT Creator
Student LR-ALBERT
Relational Knowledge Mapper
LR–Distillation Loss Function
LR–Distillation
Dataset
LR-ALBERT
Student IP–ALBERT Creator
Student IP-ALBERT
Relational Knowledge Mapper
IP–Distillation Loss Function
IP–Distillation
F-ALBERT

Figure 1. The F-ALBERT Distillation System distills ALBERT into F-ALBERT through two distillation processes (LR-ALBERT Distillation, IP-ALBERT Distillation).

3.2. LR-ALBERT Distillation

3.2.1. Student LR-ALBERT Creator

This component creates and initializes the Student LR-ALBERT model, the student model for the first distillation. The Student LR-ALBERT is generated to have the same structure as ALBERT. However, the number of times the layer parameters are reused is set to be less than it is with ALBERT. The initialization process involves copying ALBERT's parameters to the student model. Using the original ALBERT parameters can significantly reduce the distillation time. We distill LR-ALBERT with various reuses and measure its performance, thus proposing the most effective number of parameter reuses.

3.2.2. LR Distillation

LR Distillation involves teaching the Student LR-ALBERT model based on the output from the last layer of the ALBERT model for all given data samples. However, the student model has lower computational complexity than the teacher model. This results in a decrease in the expressiveness of the model, making it difficult to approximate the output generated by the teacher model directly [28]. If approximated directly, performance degradation is possible due to noise learning. Therefore, instead of directly learning the output of the teacher model, we describe the summarized relationships between the word vectors created by the teacher model and train that relationship [21]. The Transformer layer that makes up ALBERT takes vector sequences corresponding to word tokens as input and, with each layer iteration, returns an encoded vector sequence of the received sequences. Let us denote the encoded vector sequence output from an arbitrary layer as $V = (v_1, v_2, v_3, \ldots, v_n)$. The method for mapping this to Relational Knowledge Representation uses the following suggested Relational Knowledge Mapper $F(V)$.

$$F(V) = \begin{pmatrix} \frac{v_1 \cdot v_1}{\|v_1\|\|v_1\|} & \cdots & \frac{v_1 \cdot v_n}{\|v_1\|\|v_n\|} \\ \vdots & \ddots & \vdots \\ \frac{v_n \cdot v_1}{\|v_n\|\|v_1\|} & \cdots & \frac{v_n \cdot v_n}{\|v_n\|\|v_n\|} \end{pmatrix} \tag{1}$$

Relational Knowledge Mapper transforms all output word vectors into a cosine similarity matrix between words. This matrix summarizes the relationships between words and is lightweight and normalized for faster training speed. Furthermore, as this is a summarized representation of the original output, training through this minimizes noise and allows for learning only the critical features for relationship restoration. Let us call the sequence of output encoding word vectors of the teacher and student models T and S, respectively. The Loss Function used to train this representation in the LR-ALBERT model is as follows.

$$\text{Loss Function}(F(S), F(T)) = \sum \text{MSE}(F(S), F(T)) \tag{2}$$

This loss function is used to calculate the difference between the teacher and the student and update the parameters of the student. While there are various functions to convey Relational Knowledge, considering our experimental results, it is the most appropriate representation to deliver. LR-ALBERT is the intermediate model, distilled through the Mapper and Loss Function for all datasets. Figure 2 provides a summary of the entire process of LR-Distillation.

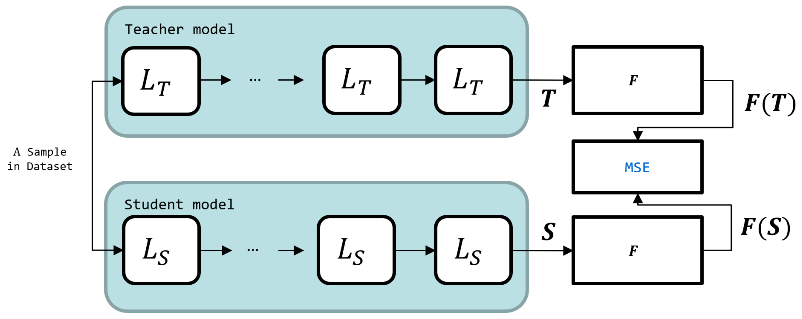


Figure 2. The LR-Distillation Loss Function converts the output of the teacher and student models for a given sample into relational knowledge. Training proceeds so that the student model approximates the relational knowledge of the teacher model.

3.3. Introduction to IP-ALBERT

IP-ALBERT is a modified version of the original ALBERT structure, using our proposed Partitioned Transition Module instead of the original ALBERT Transformer layer's Transition Module. The Partitioned Transition Module manages all the parameters of the existing Transition Module by partitioning them into equal parts [27], selecting and combining the divided parameters according to the number of reuses of the layer, and utilizing them. Additionally, we introduced an Iterative Weight Pruning technique that gradually reduces the number of partitions to be combined whenever a layer is reused. IP-ALBERT implements a Partitioned Transition Module and is a structure in which the Iterative Weight Pruning technique is applied to gradually reduce the number of parameters to be combined each time a layer is reused.

3.3.1. Partitioned Transition Module

The Partitioned Transition Module is a modified structure of ALBERT's existing Transition Module, enabling efficient parameter management. The key function of the Transition Module is to affine an input vector into a high-dimensional intermediate representation and then convert it back to its original dimension. This performs a total of two affine transformations. One affine transformation uses a linear transformation matrix parameter and a translation transformation vector parameter. A feature of the Partitioned Transition Module is that the parameter matrices used for transformation are divided into N equal parts and managed. This property provides efficiency and scalability in handling high-dimensional data transformations. This N is designated by pre-determining the number of parameters to be divided. The Partitioned Transition Module receives the connection length C as an input value each time it is called. C determines the number of pieces to be combined for each partitioned parameter matrix, and C pieces are selected from the front among the N equally divided parameters and combined. The combined parameters perform the vector mapping operation, which is the role of the Transition Module. The Partitioned Transition Module effectively manages computational complexity by enabling the flexible use of as many parameters as necessary.

3.3.2. Iterative Weight Pruning

Iterative Weight Pruning is a technique that optimizes the model's performance by gradually reducing the number of parameters while focusing on critical parameters. This technique is practically implemented by combining the Partitioned Transition Module, a core element of the IP-ALBERT model, and the connection length C. The Partitioned Transition Module manages parameters for complex operations in multiple parts, and C decides how many partitions to combine. As the model progresses, each time a layer is reused, the connection length C decreases gradually, thereby reducing the number of parameters participating in the operation. This process gradually eliminates unimportant parameters each time they are reused, allowing for more focus on important parameters.

Through this process, the overall computational complexity of the model can be reduced, and the performance of learning and inference can be improved.

3.4. IP-ALBERT Distillation

3.4.1. Student IP-ALBERT Creator

This component takes LR-ALBERT as input and generates and initializes the Student IP-ALBERT model, the second distillation student model. The Student IP-ALBERT is created to have the same structural hyperparameters and the same number of layer reuses as LR-ALBERT. However, the Partitioned Transition Module is created by partitioning each of the parameters constituting the existing Transition Module based on the parameter partition parameter N. The Student IP-ALBERT can significantly reduce the distillation time by copying and reusing the parameters of LR-ALBERT. At this time, the parameters of the Transition Module are copied after partitioning. Additionally, Student IP-ALBERT needs to set an appropriate connection length schedule $C = [C_0, C_1, C_2 \ldots, C_{Iter}]$ by setting an appropriate C_i for each layer module reuse count i. We propose a method to schedule the optimal C through experiments.

3.4.2. IP Distillation

IP Distillation involves having the student model approximate the output occurring in all Transformer layers of the teacher for all provided data samples. Like LR-Distillation, it approximates the summarized Relational Knowledge instead of directly copying the Transformer layer outputs. The difference from LR Distillation is that the number of times the given model is reused is the same, so the outputs per reuse can be matched one-to-one. Following is the distillation loss function for module outputs.

$$\text{Loss Function} = \sum_{i=0}^{Iter} \text{MSE}(F(S_i), F(T_i)) \tag{3}$$

This provides more specific learning instructions to the student model, enabling faster and more accurate training of the model. In addition, it allows for training pruned weights to operate more efficiently for each reuse count. F-ALBERT is a result distilled with a mapper and loss function for all data sets. Figure 3 provides a summary of the entire process of IP-Distillation.

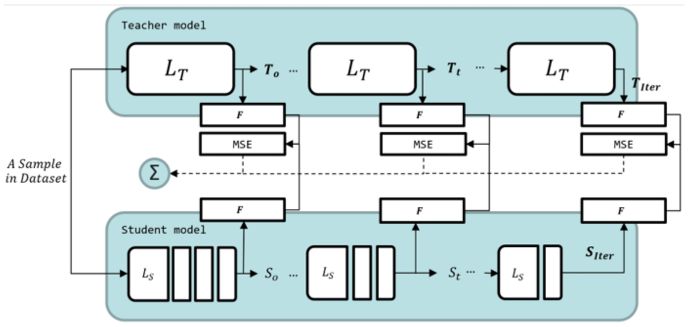


Figure 3. The IP Distillation Loss Function converts the outputs generated from iterations of all layers of teacher and student into relational knowledge and proceeds to approximate all layers.

4. Experiments

4.1. Experimental Setup

4.1.1. Teacher Models

We selected two ALBERT models according to the parameters scale: ALBERT-Base (hidden size = 768, layer iteration = 12) and ALBERT-Large (hidden size = 1024, layer iteration = 24). We evaluate the process of distilling these two models and the results of distillation. By comparing these results, we analyze the size-dependent variable of the proposed distillation techniques. Both models process the input stream using the same tokenizer and have the same embedding dictionary size (vocab size = 30,000). Both models were pre-trained with the same dataset: the BookCorpus [29] and the Wiki Corpus. These models were trained for 125,000 steps with a batch size of 4096. The learning rate started at 0.00176 and was updated using the LAMB optimizer [30].

4.1.2. Corpus Datasets

The existing teacher model was pre-trained on Wiki Corpus and BookCorpus. We utilized the C4 (Common Crawl's Web Crawl Corpus) Dataset [31] in the model distillation process. C4 is a dataset that extracts text from web pages and removes web page elements like HTML tags, scripts, styles, etc. Moreover, only text written in the main languages was extracted through language detection, and noise such as typos, grammar, informal language use, special characters, etc., was removed. Using a dataset from which noise has been removed using various techniques can prevent a decrease in training efficiency due to noise during the training process. By distilling the model using a new dataset instead of the existing one, we can obtain the effects of learning from various data and improving generalization performance.

4.1.3. GLUE Benchmark Datasets and Downstream Setup

The GLUE (General Language Understanding Evaluation) benchmark [32] is a collection of standardized datasets for evaluating natural language understanding (NLU). It comprises various natural language processing (NLP) tasks to assess a model's overall language understanding capability. The GLUE benchmark includes several subtasks:

- Multi-Genre Natural Language Inference (MNLI): Classifying the relationship between two sentences. Given two sentences, it determines whether the second sentence is in an entailment, neutral, or contradiction relationship with the first one.
- Quora Question Pairs (QQP): This binary classification task involves determining whether pairs of questions collected from Quora are semantically identical.
- Stanford Sentiment Treebank (SST): This binary classification task categorizes the sentiment of sentences extracted from movie reviews as positive or negative.
- Corpus of Linguistic Acceptability (CoLA): This binary classification task involves judging whether a sentence is grammatically correct.
- Microsoft Research Paraphrase Corpus (MRPC): This binary classification task involves determining whether two sentences are paraphrases of each other.
- Recognizing Textual Entailment (RTE): This binary classification task involves determining whether the second sentence can be inferred from the first when two sentences are given. This task is similar to the MNLI task but uses a smaller dataset.
- Question Natural Language Inference (QNLI): This binary classification task requires determining if a given sentence answers a specific question. It tests the model's ability to infer information within a question-answering context.

By synthesizing the evaluation results of each subtask, we evaluate the model's overall natural language understanding capability. Most recent NLP models are evaluated through the GLUE benchmark, and their performance continually improves.

4.2. Relational Knowledge Distillation

We configured the output as Relational Knowledge to effectively distill the Transformer Layer output of the language model. To check the effect of constructing Relational Knowledge Distillation using this expression, we compare the model's performance distilled using traditional Knowledge Distillation and the GLUE Benchmark. In addition, by actually applying various loss functions that convey relational knowledge, the proposed Loss Function proves to be effective.

4.2.1. Comparison of Traditional Knowledge Distillation

Typically, traditional Knowledge Distillation [10] transforms the output z_i of the teacher into a $Softmax$ classification probability q_i and trains it on the student model. Given the probability q_i of the teacher for the i-th class, the probability p_i of the student after the same $Softmax$ transformation and the temperature T, the loss function is composed using a KL Divergence function.

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \quad \text{loss} = \sum_j \text{KL}(q_i, p_i) \tag{4}$$

We configured student models from the ALBERT-Base and ALBERT-Large teacher models, each having half the original layer reuse number. We performed distillation using LR-Distillation and traditional Knowledge Distillation methods for the ALBERT-Base and ALBERT-Large models and compared their GLUE performances. Figure 4 shows the result.

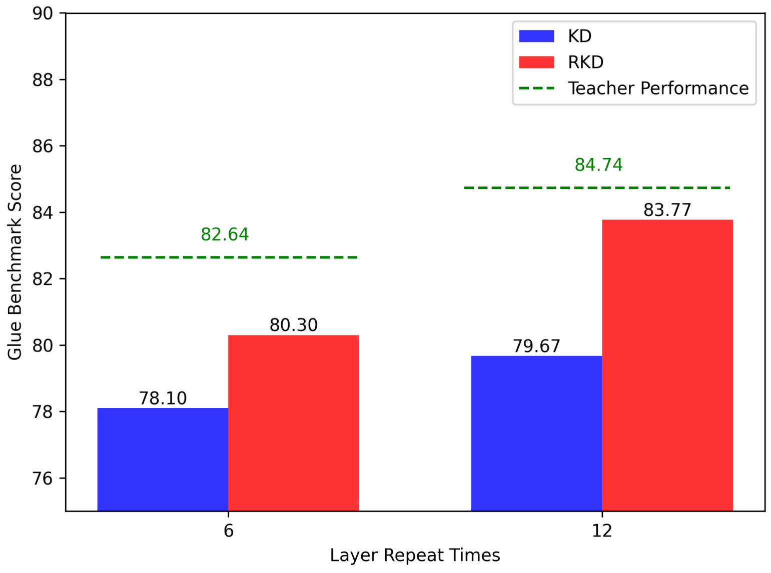


Figure 4. Distillation Results of ALBERT-Base and ALBERT-Large with Reduced Layer Reuse.

Experimental results showed that the model distilled through Relational Knowledge Distillation performed better in the GLUE benchmark score. Existing traditional, probability-based distillation techniques are suitable for distilling the information generated by the head layer for classification. They are not ideal for directly training the knowledge of the complex encoding layer generated by the model. Also, using the $softmax$ function, information corresponding to a specific range is lost or underestimated. Additionally, it can distill the results of the classification head used for learning the existing language model, but it is inefficient regarding the data set and learning cost.

4.2.2. Loss Function Comparison

Various loss functions can be used to distill relational knowledge. We constructed and evaluated the LR-Distillation Loss function using the frequently used regression loss function (MAE, MSE). Using the previous method, we build a student model with half layers and large teacher models in each base and proceed with distillation. The Shannon information content change in the vector generated in the last layer of LR-ALBERT distilled in each method was measured. The Shannon information quantity is a unit for measuring the quantity of information in terms of information theory. If the amount of information is high, it can be indirectly confirmed that the corresponding vector captures complex or various features well. $softmax$ probability $p(x)$ is converted to for the model's output vector x, and the Shannon entropy H is measured as

$$H(x) = -\sum_x p(x) \log_2 p(x) \tag{5}$$

We prepared 10,000,000 data samples and proceeded with distillation for one epoch. One epoch comprises six steps, and the Shannon entropy of the learned model output at each training step was measured and recorded in Figure 5.

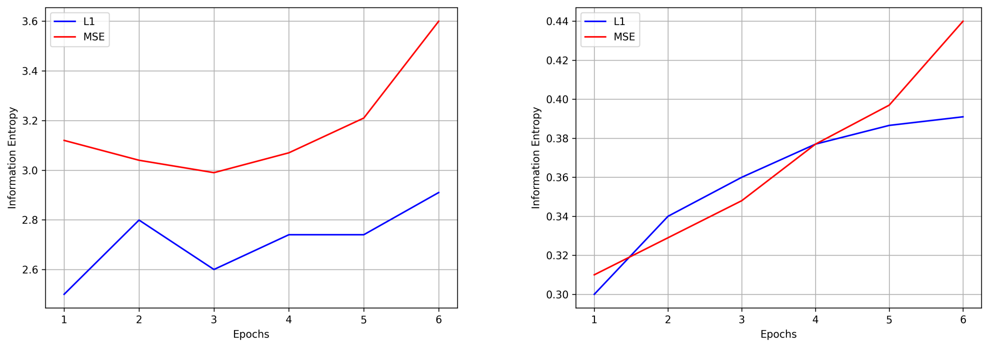


Figure 5. Entropy trend during distillation using the MAE and MSE methods, according to the learning process (**Left**: Base-size model, **Right**: Large-size model).

Overall, the base model has more information in the model's output trained by MSE. However, the large model had more information in the model's output trained with MAE up to the fourth step. This reflects the tendency of the MAE function to distill a model quickly but ignore some complex patterns. MSE takes more time to learn complex patterns and sparse features but can have a richer amount of information. Since the output of the language model is significant for very sparse pattern features, MSE may be more appropriate. We constructed an additional experiment to see if the model with a large amount of information performed excellently in the fine-tuning task. The GLUE benchmark performance of the Base and Large models trained up to six steps with MAE and MSE was measured and compared. Table 1 presents the measurement results.

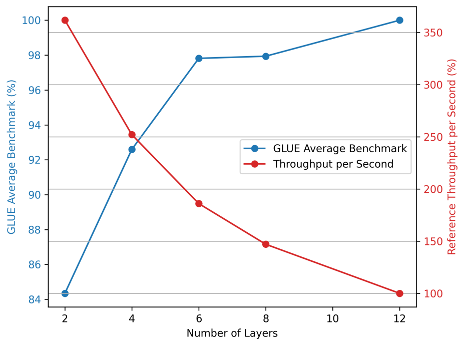
Table 1. MAE vs. MSE GLUE benchmark.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	AVG
MAE BASE	81.1	90.1	87.3	71.1	90.6	86.9	48.1	79.31
MSE BASE	82.3	89.4	87.3	73.8	90.2	86.7	52.5	80.3
MAE LARGE	83.3	90.8	90.1	75.1	91.8	89	56.3	82.34
MSE LARGE	85.3	91.1	89.5	78.5	92.7	90	59.3	83.77

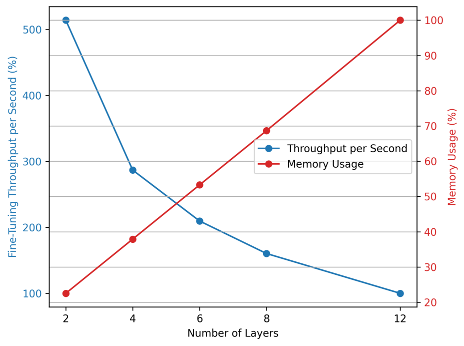
Even in the GLUE benchmark performance, the average score of the model trained by MSE is higher. In particular, tasks for small-scale learning data environments such as RTE and CoLA were much higher. This shows that the model trained with the MSE function can grasp the complex patterns of the language model and simultaneously have outstanding generalization ability for new data.

4.3. *LR-ALBERT Performance Test*

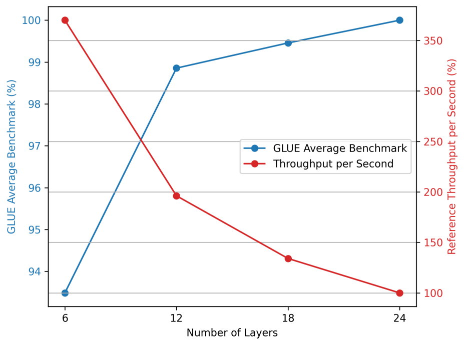
LR Distillation distills the teacher model into the student model based on RKD. At this time, the student model can have various reuse counts. We distill the ALBERT-Base and ALBERT-Large models into student models with various reuse counts. The GLUE benchmark table and computing performance (Memory Usage, Fine-tuning, Throughput per Second) of LR-ALBERT after distillation are presented. The performance presented in Figure 6 is normalized as a percentage by dividing the measurement result of LR-ALBERT by the measurement result of the existing teacher model.



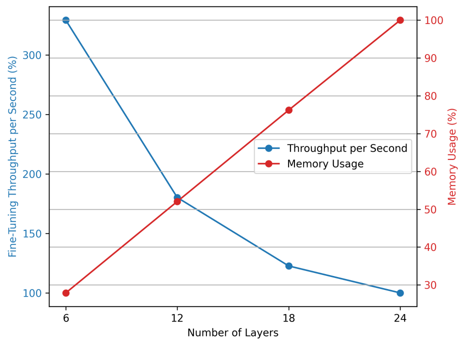
(a)



(b)



(c)



(d)

Figure 6. GLUE Average Benchmark scores and Throughput per Second performance according to the number of layer reuses ((**a**): Base-size model, (**c**): Large-size model), Fine-Tuning Throughput per Second and Memory Usage according to the number of layer reuses ((**b**): Base size model, (**d**): Large size model).

According to the results, the base and large models exhibited good performance when distilled into LR-ALBERT, which has approximately half the number of layer reuse instances. The GLUE benchmark performance of the LR-ALBERT model, with a half level of layer reuse, dropped by around 1–2%, but the memory usage decreased to about half, and the processing speed increased by roughly two times. We suggest setting the layer-reuse frequency of LR-ALBERT to half of its original number as a reasonable benchmark.

4.4. IP-Distillation Scheduling Experiments

To maximize the calculation speed and generate F-ALBERT with minimal performance degradation, it is necessary to schedule an effective connection length according to the number of reuses of IP-ALBERT. There are many very diverse cases for the connection length schedule. We propose effective schedules for the distilled LR-ALBERT base and large models with half the number of layer reuses. To construct an effective schedule, we analyze the change in output as the number of reuses of LR-ALBERT increases and suggest several schedules based on this. According to the proposed schedule, we proceed with IP-Distillation and present the final performance of F-ALBERT.

4.4.1. Analysis of Difference in Relational Knowledge in LR-ALBERT

IP-ALBERT combines and utilizes partitioned parameters according to the number of layers reused. A layer takes a recursive configuration, receiving the previous layer's output as input. Since the distribution of input changes as layers are repeated, even if a module composed of the same parameters is reused, each layer operates differently. We assume that with each layer iteration, the parameter and amount of parameters primarily involved in interpreting the input gradually decrease. We prepare LR-ALBERT (Base) and LR-ALBERT (Large) by distilling ALBERT (Base) and ALBERT (Large) into half layers to measure the size of the change per layer. We prepared an evaluation dataset from a corpus not used in training (sample = 15,000) and transformed the output results of LR-ALBERT, depending on the parameter reuse, into Relational Knowledge. Figure 7 shows the comparison of the Euclidean Distance of the final layer result.

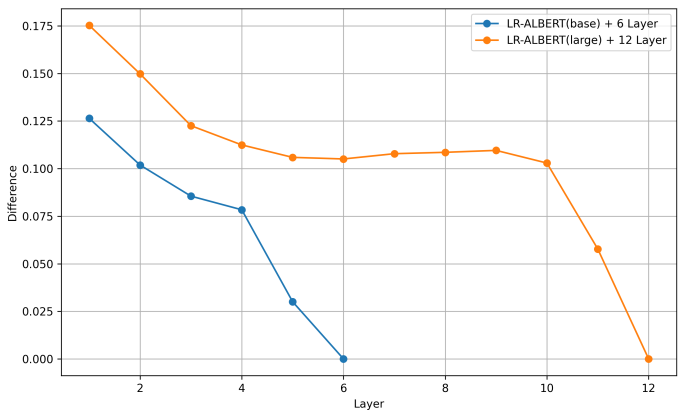


Figure 7. Trend of the change in Euclidean Distance between the final output Relational Knowledge of LR-ALBERT and the Relational Knowledge of each layer.

It can be observed that the amount of change varies significantly until the reuse of the second layer, and the amount of change decreases considerably from the reuse of the second layer before output. The rest of the layers show similar variations. We propose scheduling the connection length by partitioning it into three steps using this result. We now set the partition size to four through empirical results. Since the layer module significantly changes the vector in the first step, all partitions are connected. In the second step, the amount of change is less than that of the first step, but the difference is maintained at a certain level. Therefore, this step connects all partitions or removes one partition. The last step reduces the variance, significantly reducing the connection partition. The following Schedule is the scheduling for the LR-ALBERT base model and the Large model distilled with half the number of reuses, respectively.

4.4.2. F-ALBERT Performance

F-ALBERT is the result of distilling LR-ALBERT into IP-ALBERT. Based on LR-ALBERT, which reuses half of the layers as before, we proceeded with distillation on IP-ALBERT with the schedule presented above. The performance presented in Figure 8 is normalized as a percentage by dividing F-ALBERT's measurement result by that of the existing LR-ALBERT model.

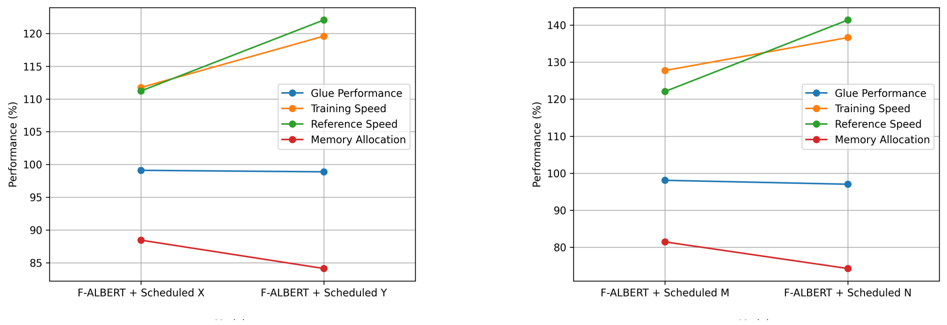


Figure 8. Measurement of F-ALBERT's computing and GLUE average score performance based on different schedule configurations (**left**: Base-size model, **right**: Large-size model).

Our experiments confirmed that when the connection length was additionally reduced in reusing the second step, a decrease in the Glue benchmark performance of 1 to 2% occurred. Through this, it was confirmed that the scheduling technique based on the amount of change was valid. We present the final F-ALBERT standard schedule as Y for the base model and N for the large model. Compared to LR-ALBERT, the base model increased in speed by 20% and decreased in memory usage by 15%, and the large model increased in speed by about 40% and decreased in memory usage by 15%. Table 2 is the final GLUE-benchmark result of F-ALBERT.

Table 2. NLU benchmark performance according to schedule configuration for each model.

Schedule	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	AVG
X	81	88.8	89	71.1	91.7	85.2	52.4	79.886
Y	80.6	88.4	89.3	72.3	91.5	87.5	48.3	79.7
M	83.3	90.2	89.1	76.1	92.4	89.8	54.3	82.17
N	82.6	89.6	89	75.2	91.3	89.2	52.1	81.29

F-ALBERT comprises various hyperparameters that can directly affect the model's generalization performance. We identify crucial parameters that exert a direct influence on F-ALBERT's functionality, proposing hyperparameter settings rooted in previously conducted experiments. Hyperparameters can be divided into two facets: learning progression and model composition.

In the learning dimension, the epoch count is vital. We employed the C4 dataset for distillation, observing that when the number of epochs exceeded two, there was a marginal reduction in benchmark performance. Therefore, we suggest training on a range of non-overlapping datasets or configuring training with a lower number of epochs for distillation tasks.

Regarding model composition, the careful configuration of LR-ALBERT's constituent layers is essential. As per Tables 3 and 4, if LR-ALBERT is configured with fewer than half of the original layers, performance markedly declines. LR-ALBERT exhibited optimal performance when composed of half the number of the original teacher layers.

Table 3. GLUE the benchmark for various layers of base LR-ALBERT.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	AVG
ALBERT-Base +12 Layer	84.7	90.5	89	77.9	92	88.9	55.5	82.64
LR-ALBERT +8 Layer	81.9	89.1	89.9	74.3	91.6	85.3	53.1	80.74
LR-ALBERT +6 Layer	81.3	89.4	89.3	73.8	91.2	86.7	52.5	80.6
LR-ALBERT +4 Layer	79.2	86.5	86.6	64.6	91.2	81.1	45.3	76.3
LR-ALBERT +2 Layer	72.7	80.8	86	63.5	87	76.5	20	69.5

Table 4. GLUE benchmark for various layers of large LR-ALBERT.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	AVG
ALBERT-Large +24 Layer	87.1	91.8	90.1	80.3	92	91	60.9	84.74
LR-ALBERT +18 Layer	85.3	91.76	90.3	80.5	93.6	90.2	58.3	84.28
LR-ALBERT +12 Layer	85.3	91.1	89.5	78.5	92.7	90	59.3	83.77
LR-ALBERT +6 Layer	81.8	88.5	89	68.5	92.3	86.9	47.6	79.23

Moreover, IP-ALBERT's schedule is a key model hyperparameter. In Tables 5 and 6, we have presented schedules for LR-ALBERT according to its sizes. Specifically, Schedules Y and N are those with minimized computational requirements that can ensure generalization performance. Based on empirical findings, establishing a schedule with even less computation led to a generalization performance decrease of more than 15%. Therefore, we recommend employing the proposed Y and N schedules.

Table 5. LR-ALBERT (base) IP-Distillation Schedules.

Schedule	**1**	**2**	**3**	**4**	**5**	**6**
X	4	4	4	4	2	1
Y	4	4	3	3	2	1

Table 6. LR-ALBERT (large) IP-Distillation Schedules.

Schedule	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
M	4	4	4	4	4	4	4	4	4	4	2	1
N	4	4	3	3	3	3	3	3	3	3	2	1

The Table 7 compares distilled models possessing the same structure of the Transformer layer as the proposed F-ALBERT, including TinyBERT [33], DistillBERT [34], ALP-KD [19], and LAD [20], all of which are composed of six Transformer layers of similar size to F-ALBERT. Since F-ALBERT reuses parameters, it exhibits approximately six times the parameter efficiency of existing methods. This allows it to operate efficiently in embedded environments where GPU memory resources are limited. Additionally, while existing methods focus solely on reducing layers through distillation, our approach gradually reduces computation parameters layer by layer through additional distillation in the IP-ALBERT structure, recording a speed increase of approximately 20%. Moreover, by distilling Relational Knowledge, we have secured generalized performance across various NLU tasks. Furthermore, F-ALBERT, with its large size, demonstrated higher compression ratios compared to its base size, along with a 30% improvement in speed. However, parameter reuse ultimately led to a performance decrease of up to 5% due to insufficient parameters compared to other methods. We verified that this can be resolved by expanding the size of one Transformer, as demonstrated in F-ALBERT (base) and F-ALBERT (large).

Table 7. Performance analysis of the proposed F-ALBERT compared to existing methodologies. The speedup factor measures how much faster F-ALBERT (base) is relative to the corresponding method. Additionally, the experiment recorded inference scores for various NLP tasks.

Model	**Parameters**	**Speedup**	**CoLA**	**RTE**	**MNLI**	**MRPC**	**QNLI**
F-ALBERT (base)	12 M	1.0×	48.3	72.3	80.6	87.5	88.4
TinyBERT	66 M	1.2×	50.1	70.0	81.2	87.3	90.4
DistillBERT	66 M	1.2×	45.3	65.5	80.5	85	89
ALP-KD	66 M	1.2×	-	68.59	81.86	85.05	89.67
LAD	66 M	1.2×	-	68.59	83.78	84.56	90.74
F-ALBERT (large)	18 M	2.5×	52.1	75.2	82.6	89.2	89.6
BERT-PKD (large)	180 M	3.0×	56.4	76.5	84.1	90	91.1

5. Conclusions

In this study, we presented a novel approach to enhance the efficiency of the ALBERT model for natural language processing tasks. Utilizing Knowledge Distillation, we developed a system that transformed the existing ALBERT model into F-ALBERT, a significantly faster variant.

Our research contributions can be categorized into two primary areas. Firstly, we proposed a speed-optimization technique for models based on Knowledge Distillation. We introduced an innovative knowledge representation method specifically designed for distilling ALBERT. Additionally, we developed effective strategies for knowledge transfer through LR-Distillation and IP-Distillation. Secondly, we presented a new Iterative Weight Pruning technique, which systematically pruned computational parameters at each instance of parameter reuse. We proposed a Partitioned Transition Module to implement this technique, which replaced the existing Transition Module within a layer.

We demonstrated that our final distilled model, F-ALBERT, substantially reduced the computational load by approximately half compared to the original ALBERT. Remarkably, despite the reduction in computational load, F-ALBERT retained nearly 98% of the original model's performance. The results of our research underlined the potential of distillation techniques for model optimization, suggesting a promising avenue for achieving a balance between high performance and computational efficiency in natural language processing tasks.

In our research, the F-ALBERT model has demonstrated strengths in high parameter efficiency and low memory utilization. Future research will focus on enhancing throughput in embedded environments through the quantization, optimization, and expansion of the IP-ALBERT structure, leveraging these advantages. Quantization will enable its efficient execution in small-scale embedded settings, while the IP-ALBERT structure is anticipated to improve performance by exploring optimized parameter connections for each layer. Moreover, we will concentrate on the development and optimization of models specialized for specific industries and application areas through the extension of these technologies. In this way, we plan to explore performance enhancements and new possibilities in various environments and application fields, maximizing the efficiency and versatility of F-ALBERT.

Author Contributions: Conceptualization, K.-H.K. and C.-S.J.; methodology, K.-H.K. and C.-S.J.; software, K.-H.K.; validation, K.-H.K. and C.-S.J.; formal analysis, K.-H.K. and C.-S.J.; investigation, K.-H.K.; resources, K.-H.K.; data curation, K.-H.K.; writing—original draft preparation, K.-H.K.; writing—review and editing, K.-H.K. and C.-S.J.; visualization, K.-H.K.; supervision, C.-S.J.; project administration, K.-H.K. and C.-S.J.; funding acquisition, C.-S.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: This work was supported by the artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT (MSIT, Korea) and Gwangju Metropolitan City.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *2017*, 5999–6009.
2. Mars, M. From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough. *Appl. Sci.* **2022**, *12*, 8805. [CrossRef]
3. Garrido-Muñoz, I.; Montejo-Ráez, A.; Martínez-Santiago, F.; Ureña-López, L.A. A survey on bias in deep NLP. *Appl. Sci.* **2021**, *11*, 3184. [CrossRef]
4. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT. In Proceedings of the NAACL HLT 2019—2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186.
5. Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; Zhou, D. MobileBERT. *arXiv* **2020**. [CrossRef]
6. Mehta, S.; Ghazvininejad, M.; Iyer, S.; Zettlemoyer, L.; Hajishirzi, H. Delight: Deep and light-weight transformer. *arXiv* **2020**, arXiv:2008.00623.
7. Wang, Z.; Wohlwend, J.; Lei, T. Structured pruning of large language models. In Proceedings of the EMNLP 2020—2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, Virtual, 16–20 November 2020; pp. 6151–6162. [CrossRef]
8. Liu, H.; Yan, H.; Xia, J.; Ai, Y. Entropy targets for adaptive distillation. In Proceedings of the ICRAI '20: Proceedings of the 6th International Conference on Robotics and Artificial Intelligence, Singapore, 20–22 November 2020; pp. 179–183. [CrossRef]
9. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv* **2019**, arXiv:1909.11942.

10. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
11. Clark, K.; Khandelwal, U.; Levy, O.; Manning, C.D. What does bert look at? an analysis of bert's attention. *arXiv* **2019**, arXiv:1906.04341.
12. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
13. Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism *arXiv* **2019**, arXiv:1909.08053.
14. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
15. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
16. Tang, J.; Shivanna, R.; Zhao, Z.; Lin, D.; Singh, A.; Chi, E.H.; Jain, S. Understanding and Improving Knowledge Distillation. *arXiv* **2018**, arXiv:2002.03532.
17. Fournier, Q.; Caron, G.M.; Aloise, D. A practical survey on faster and lighter transformers. *ACM Comput. Surv.* **2021**, *55*, 304. [CrossRef]
18. Sun, S.; Cheng, Y.; Gan, Z.; Liu, J. Patient Knowledge Distillation for BERT Model Compression. *arXiv* **2019**, arXiv:1908.09355.
19. Passban, P.; Wu, Y.; Rezagholizadeh, M.; Liu, Q. ALP-KD: Attention-Based Layer Projection for Knowledge Distillation. *arXiv* **2020**, arXiv:2012.14022.
20. Lin, Y.J.; Chen, K.Y.; Kao, H.Y. LAD: Layer-Wise Adaptive Distillation for BERT Model Compression. *Sensors* **2023**, *23*, 1483. [CrossRef]
21. Park, W.; Corp, K.; Kim, D.; Lu, Y. Relational Knowledge Distillation (CVPR, 2019). In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3967–3976.
22. Heo, B.; Kim, J.; Yun, S.; Park, H.; Kwak, N.; Choi, J.Y. A comprehensive overhaul of feature distillation. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1921–1930. [CrossRef]
23. Wu, X.; Liu, Y.; Zhou, X.; Yu, D. Distilling knowledge from pre-trained language models via text smoothing. *arXiv* **2020**, arXiv:2005.03848.
24. Wang, T.; Zhou, W.; Zeng, Y.; Zhang, X. EfficientVLM: Fast and Accurate Vision-Language Models via Knowledge Distillation and Modal-adaptive Pruning. *arXiv* **2022**, arXiv:2210.07795.
25. Peng, Y.; Sudo, Y.; Muhammad, S.; Watanabe, S. DPHuBERT: Joint Distillation and Pruning of Self-Supervised Speech Models. *arXiv* **2023**, arXiv:2305.17651.
26. Zuo, S.; Zhang, Q.; Liang, C.; He, P.; Zhao, T.; Chen, W. MoEBERT: From BERT to Mixture-of-Experts via Importance-Guided Adaptation. *arXiv* **2022**, arXiv:2204.07675.
27. Fedus, W.; Zoph, B.; Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv* **2021**, arXiv:2101.03961.
28. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. In Proceedings of the NAACL HLT 2018—2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LO, USA, 1–6 June 2018; Volume 2, pp. 464–468. [CrossRef]
29. Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; Fidler, S. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015.
30. You, Y.; Li, J.; Reddi, S.; Hseu, J.; Kumar, S.; Bhojanapalli, S.; Song, X.; Demmel, J.; Keutzer, K.; Hsieh, C.J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv* **2019**, arXiv:1904.00962.
31. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
32. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv* **2018**. [CrossRef]
33. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. Tinybert: Distilling bert for natural language understanding. *arXiv* **2019**, arXiv:1909.10351.
34. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.