MiniALBERT: Model Distillation via Parameter-Efficient Recursive Transformers

Mohammadmahdi Nouriborji[2†], **Omid Rohanian**[1,2†], **Samaneh Kouchaki**[3], **David A. Clifton**[1,4]
[1]Department of Engineering Science, University of Oxford, Oxford, UK
[2]NLPie Research, Oxford, UK
[3]Dept. Electrical and Electronic Engineering, University of Surrey, Guildford, UK
[4]Oxford-Suzhou Centre for Advanced Research, Suzhou, China
{m.nouriborji,omid}@nlpie.com
samaneh.kouchaki@surrey.ac.uk
{omid.rohanian,david.clifton}@eng.ox.ac.uk

Abstract

Pre-trained Language Models (LMs) have become an integral part of Natural Language Processing (NLP) in recent years, due to their superior performance in downstream applications. In spite of this resounding success, the usability of LMs is constrained by computational and time complexity, along with their increasing size; an issue that has been referred to as 'overparameterisation'. Different strategies have been proposed in the literature to alleviate these problems, with the aim to create effective compact models that nearly match the performance of their bloated counterparts with negligible performance losses. One of the most popular techniques in this area of research is model distillation. Another potent but underutilised technique is cross-layer parameter sharing. In this work, we combine these two strategies and present MiniALBERT, a technique for converting the knowledge of fully parameterised LMs (such as BERT) into a compact recursive student. In addition, we investigate the application of bottleneck adapters for layer-wise adaptation of our recursive student, and also explore the efficacy of adapter tuning for fine-tuning of compact models. We test our proposed models on a number of general and biomedical NLP tasks to demonstrate their viability and compare them with the state-of-the-art and other existing compact models. All the codes used in the experiments are available at https://github.com/nlpie-research/MiniALBERT. Our pre-trained compact models can be accessed from https://huggingface.co/nlpie.

[†]The two authors contributed equally to this work.

1 Introduction

Following the introduction of BERT (Devlin et al., 2019), generic pre-trained Language Models (LMs) have started to dominate the field of NLP. Virtually all state-of-the-art NLP models are built on top of some large pre-trained transformer as a backbone and are subsequently fine-tuned on their target dataset. While this pre-train and fine-tune approach has resulted in significant improvements across a wide range of NLP tasks, the widespread use of resource-exhaustive and overparameterised transformers has also raised concerns among researchers about their energy consumption, environmental impact, and ethical implications (Strubell et al., 2019; Bender et al., 2021).

As a response to this, different approaches have appeared with the aim to make large LMs more efficient, accessible, and environmentally friendly. Model compression is a line of research that has recently received considerable attention. It involves encoding a larger and slower but more performant model into a smaller and faster one with the aim to retain much of the former's performance capability (Bucilua et al., 2006). Knowledge distillation (Hinton et al., 2015), quantisation (Shen et al., 2020), and pruning (Ganesh et al., 2021) are three examples of such methods.

Adapter modules (Bapna and Firat, 2019; He et al., 2021) are recently introduced as an effective mechanism to address the parameter inefficiency of large pre-trained models. In this approach, several 'bottleneck adapters'(Houlsby et al., 2019a) are embedded inside different locations within the original network. During fine-tuning, the parameters of the

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 1161–1173

original model are kept fixed, and for each new task only the adapters are fine-tuned. This only adds a small number of parameters to the overall architecture and allows for a much faster and more efficient fine-tuning on different downstream tasks.

Another approach to improve efficiency of LM-based transformers is shared parameterisation, which was popularised by ALBERT (Lan et al., 2019). While the original formulation of transformers (Vaswani et al., 2017) employs full parameterisation wherein each model parameter is independent of other modules and used only once in the forward pass, shared parameterisation allows different modules of the network to share parameters, resulting in a more efficient use of resources given the same parameterisation budget. However, a common downside of this approach is slower inference time and reduced performance. Ge and Wei (2022) posits two different parameterisation methods as an attempt to address the compute and memory challenges of transformer models and explores layer-wise adaptation in an encoder-decoder architecture. These methods exploit cross-layer parameter sharing in a way that would allow for the model to be utilised on mobile devices with strict memory constraints while achieving state-of-the-art results on two seq2seq tasks for English.

In this work, we exploit some of the above approaches to create a number of compact and efficient encoder-only models distilled from much larger language models. The contributions of this work are as follows:

- To the best of our knowledge, we are the first to compress fully parameterised large language models using recursive transformers (i.e. ALBERT-like models that employ full parameter sharing).
- We demonstrate the effectiveness of our pre-trained bottleneck adapters by merely fine-tuning them on downstream tasks while still achieving competitive results.
- We present several light-weight transformers with parameters ranging from 12M for the smallest to 32M for the largest. These models are shown to perform at the same level with their fully parameterised versions.
- Finally, we evaluate our models on a wide range of tasks and datasets on general and biomedial NLP datasets.

2 Background

2.1 LM-based Transformers and Computational Complexity

Ever since the introduction of the transformer architecture (Vaswani et al., 2017), large LM-based transformers such as BERT (Devlin et al., 2019) have become increasingly more popular in NLP and lie at the heart of most state-of-the-art models. A transformer is primarily composed of a number of transformer blocks stacked on top of one another. BERT_{Base} , for instance, consists of 12 of these blocks. The most important component in a block is the multi-head self-attention module. To be useful for language tasks, transformers are pre-trained using a number of self-supervised auxiliary tasks (Xia et al., 2020); these usually include some variation of Language Modelling (LM) and an optional sentence-level prediction task. Examples of the former include Masked Language Modelling (MLM) and Casual Language Modelling (CLM). For the latter, BERT uses Next Sentence Prediction (NSP) and ALBERT (Lan et al., 2019) employs Sentence Order Prediction (SOP).

The standard approach to utilise these pre-trained models is to fine-tune them on a target task. Given N as the sequence length, the computational and time complexity of self-attention is N^2 (Keles et al., 2022). In recent years, different approaches have appeared in the literature to address this bottleneck by modifying the self-attention operation in order to improve the general efficiency of transformers (with different performance trade-offs). Tay et al. (2020) surveys the most common approaches to develop what is referred to as 'efficient transformers'.

The magnitude of the parameters of LM-based transformers is another significant issue that restricts their use. With new releases like GPT-3 and MT-NLG (Smith et al., 2022) that feature hundreds of billions of parameters, these models have become increasingly overparameterised due to the large number of layers and embedding sizes (Rogers et al., 2020).

2.2 Model Distillation

The overparameterisation issue has motivated research into developing methods to compress larger models into smaller and faster versions that perform reasonably close to their larger counterparts. Knowledge distillation (Hinton et al., 2015) is a prominent method that intended to distill a

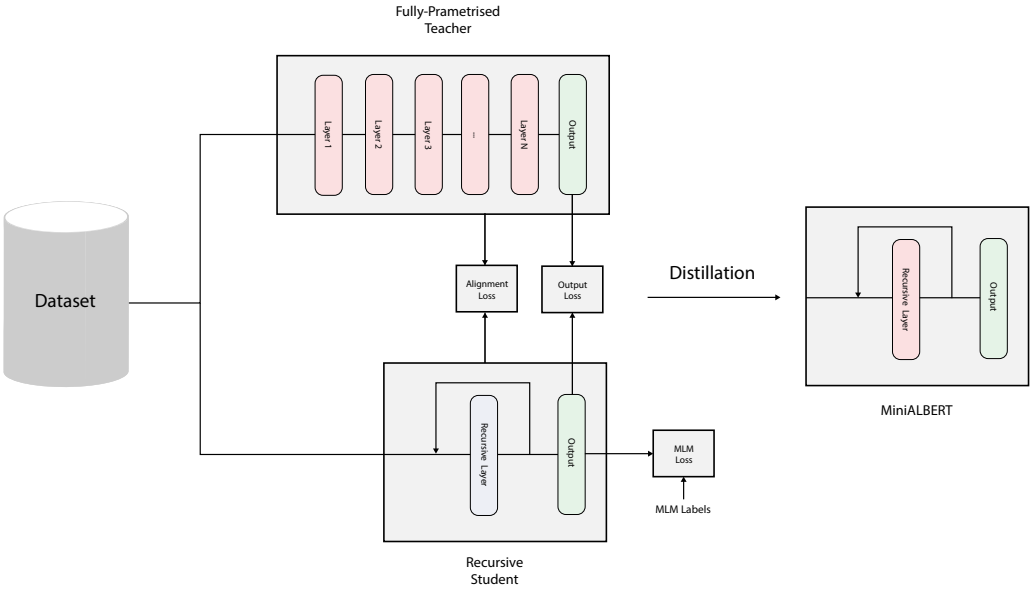


Figure 1: The layer-to-layer distillation procedure proposed for distilling the knowledge of a fully-parameterised teacher into a compact recursive student. While the teacher has fully parameterised layers, the recursive student has only one layer and the output is fed back into the same layer repeatedly. Despite this compact structure, our proposed distillation procedure is designed to align the output of each iteration of the recursive student with a particular layer of the fully-parameterised teacher, as if the student had fully-parameterised layers. Additional losses, namely, Output Loss, and MLM Loss, as shown above, are used for further knowledge distillation.

lightweight 'student' model from a larger 'teacher' network by using the outputs of the teacher netwrok as soft labels. Distillation can either be done task-specifically during fine-tuning, or task-agnostically by mimicking the MLM outputs or the intermediate representations of the teacher prior to the fine-tuning stage. The latter is more flexible and computationally less expensive (Wang et al., 2020). DistilBERT is a well-known example of a distilled model derived from BERT which is claimed to be 40% smaller in terms of parameters and 60% faster while retaining 97% of BERT's performance on a range of language understanding tasks (Sanh et al., 2019).

2.3 Efficient Fine-tuning Approaches

As discussed in Sec 2.1, LM-based transformers involve a large number of parameters and they are often fine-tuned on a target dataset. However, fine-tuning could become time-consuming as the size of the datasets grow. Different techniques exist in the literature to alleviate this bottleneck during fine-tuning. In this section we explore two of these techniques, namely, prompt tuning and bottleneck adapters.

Prompt tuning (Lester et al., 2021) is a technique in which the weights of a language model are kept frozen during the fine-tuning stage and fine-tuning is reformulated as a cloze-style task. Similar to T5, prompt tuning regards all tasks as a variation of text generation and conditions the generation using 'soft prompts'. A typical prompt consists of a text template with a masked token and a set of candidate label words to fill the mask. This turns the target task into another MLM objective in which the right candidates are chosen and soft prompts are learned. This method is especially useful for few-shot learning scenarios where there are not many target labels available for standard fine-tuning.

Bottleneck Adapters (BAs) (Houlsby et al., 2019b; Pfeiffer et al., 2021; Rücklé et al., 2020; Pfeiffer et al., 2020) are another mechanism used during fine-tuning to enhance efficiency of training. Each BA block consists of a linear down-projection, non-linearity, and up-projection along with residual connections. Several of these adapters are placed after the feed-forward or attention modules in a transformer. Similar to prompts, only the BAs are trained during fine-tuning.

2.4 Parameter Sharing via Recursion

Weight sharing is a strategy intended to reduce the overall number of parameters in a model. Lan et al.

(2019) introduced cross-layer parameter sharing in a recurrent-like encoder-only architecture where instead of having several transformer blocks with different parameters, there is only one transformer block whose outputs are recursively fed into itself a number of times. This drastically reduces the number of unique parameters in the model.

Edgeformer (Ge and Wei, 2022) is a recent work towards development of parameter-efficient encoder-decoder models specialised for on-device seq2seq generation. EdgeFormer employs two novel approaches for cost-effective parameterisation which improve on standard cross-layer parameter sharing. In addition to efficient parameterisation, Edgeformer explores the use of layer-wise adaptation in encoder-decoder models. To the best of our knowledge, however, the use of layer-wise adaptation in recursive encoder-only models (Sec 3.2) is yet to be explored.

3 Methods

In this work, we introduce a method to distil the knowledge of a fully parameterised transformer into a lightweight efficient recursive transformer via layer-to-layer distillation. We also experiment with layer-wise adaption of the recursive transformer via bottleneck adapters and factorise the embedding layer for extra parameter saving. In this section, we explain each component of our compact models in detail.

3.1 The Fully Parameterised Transformer

For each layer i, let the multi-head attention and feed-forward blocks be $f^i_{att}(x)$ and $f^i_{mlp}(x)$, respectively. The output of each layer is computed as follows:

$$O_i = f^i_{mlp}(f^i_{att}(O_{i-1})) \quad (1)$$

where O_i is the output of the i^{th} layer of the transformer.

3.2 The Recursive Student

The output of the recursive student is computed as follows:

$$O_i = f_{mlp}(f_{att}(O_{i-1})) \quad (2)$$

where O_i is the output of the i^{th} iteration of the recursive transformer. Note that unlike Equation 1, the multi-head attention and feed-forward blocks in this case are layer-agnostic, i.e. the same layers are used recursively.

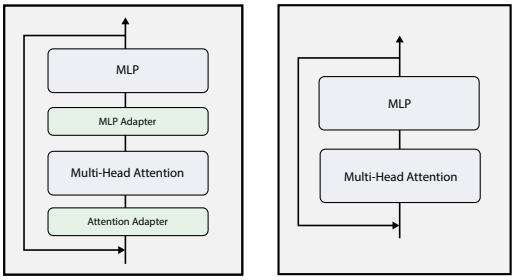


Figure 2: The two recursive students proposed in this work. (A) is a simple recursive student that employs cross-layer parameter sharing, which means that the multi-head attention and MLP blocks are shared across all layers. (B) is a recursive student with cross-layer parameter sharing which additionally uses layer-wise adaption via bottleneck adapters.

3.3 Recursive Student with Layer-wise Adaption

This formulation is identical to the recursive student defined in Section 3.2, except that, in addition to the shared parameters, here we employ a small number of trainable parameters. This will allow the model to capture distinct features in each iteration, similar to how transformer layers behave in a fully parameterised model. To this end, we use Bottleneck Adapters (BAs) which are small bottleneck blocks followed by residual connections, as defined in the following equation:

$$\phi(X) = W_{up}\ \ \sigma(W_{down}\ \ X) + X \quad (3)$$

where $\phi(.)$ represents the BA and $\sigma(.)$ is a non-linearity function such as *RELU* or *GELU*. The recursive student with layer-wise adaption can be formulated as follows:

$$O_i = f_{mlp}(\phi^i_{mlp}(f_{att}(\phi^i_{att}(O_{i-1})))) \quad (4)$$

where $\phi^i_{att}(.)$ and $\phi^i_{mlp}(.)$ are the BAs for the multi-head attention and feed-forward blocks of the i^{th} iteration of the recursive transformer.

3.4 Embedding Factorisation

Following the work of ALBERT (Lan et al., 2019), instead of a full-rank embedding matrix, we use a low-rank matrix with size $|V| \times r$ where V is the vocabulary and r is the rank of the embedding matrix. We additionally use a projection weight with the size of $r \times d$ where d is the hidden dimension of the transformer. This is set to 768 in our

experiments. Factorisation can be mathematically expressed as

$$E = E_{low}\, W_e \tag{5}$$

where $E_{low} \in \mathbb{R}^{|V|\times r}$, $W_e \in \mathbb{R}^{r\times d}$ and $E \in \mathbb{R}^{|V|\times d}$. In our experiments, we employ factorisation with ranks 312 and 128, and explore initialising the 312 versions with pre-trained embeddings obtained from the TinyBERT (Jiao et al., 2019) for the general models and TinyBioBERT (Rohanian et al., 2022) for the biomedical models.

In our experiments, we employed factorisation with ranks of 312 and 128 and initialised the 312 versions with pre-trained embeddings from TinyBERT (Jiao et al., 2019) for general models and TinyBioBERT (Rohanian et al., 2022) for biomedical models.

3.5 Distillation Procedure

We use three different loss terms, namely, MLM, alignment, and output. The MLM loss, which is the original loss used in Masked Language Modelling, is defined as follows:

$$L_{MLM}(X, Y) = \sum_{n=1}^{N} CE(f_s(X)^n, Y^n) \tag{6}$$

where X denotes the the model's input, N the number of input tokens, CE the cross-entropy loss function, Y^n the one-hot encoded label for the n^{th} token[1], and $f_s(X)^n$ the output distribution of the student for the n^{th} token. Y^n and $f_s(X)^n$ are both $|V|$-dimensional.

Because the number of layers in the student network is less than that of the teacher, alignment is typically achieved by comparing the student's layers to a subset of the teacher's. However, in our case, the student is recursive and lacks multiple layers. Therefore, in each iteration, we map the output of the student's layer to a specific layer of the teacher, i.e. the first iteration is considered the first layer, the second iteration is the second layer, and so on. The alignment loss is a combination of two terms, namely, attention and hidden losses. Attention loss is used to align the student's attention maps with those of the teacher and is defined as follows:

$$L_{att}(\hat{A}, A) = \frac{1}{HN} \sum_{h=1}^{H} \sum_{n=1}^{N} D_{KL}(\hat{A}_n^h \,||\, A_n^h) \tag{7}$$

$\hat{A}$ and A are the inputs to the loss function, and they correspond to the attention maps of a certain layer of the student and its associated layer of the teacher, respectively. The cosine-based hidden loss is used to align the hidden states of the student and teacher, and is defined as:

$$L_{hidden}(\hat{H}, H) = \frac{1}{N} \sum_{n=1}^{N} 1 - \psi(\hat{H}^n, H^n) \tag{8}$$

where $\hat{H}$ and H are the hidden states of a particular layer in the student and teacher networks. The ψ function denotes cosine similarity[2]. The alignment loss is defined as:

$$\begin{aligned} L_{align}(\hat{A}, A, \hat{H}, H) = \quad & L_{att}(\hat{A}, A) \\ & + L_{hidden}(\hat{H}, H) \end{aligned} \tag{9}$$

The output loss is based on KL divergence and is inteneded to align the output distribution of the student with the teacher on the MLM objective. This loss term is defined as below:

$$L_{out}(X) = \sum_{n=1}^{N} W_n D_{KL}(f_s(X)^n \,||\, f_t(X)^n) \tag{10}$$

where $f_s(X)^n$ and $f_t(X)^n$ are the output distributions of the student and teacher for the n^{th} token, respectively. W_n is 1 if the n^{th} token is masked and 0 otherwise. This ensures that only the masked tokens will contribute to the loss.

The complete layer-to-layer distillation loss used in this study is expressed by the following equation:

$$\begin{aligned} L(X, Y, A_s, A_t, H_s, H_t) = \quad & \\ & \lambda_1\, L_{MLM}(X, Y) \\ & + \lambda_2 \sum_{l=1}^{L} L_{align}(A_s^l, A_t^{g(l)}, H_s^l, H_t^{g(l)}) \\ & + \lambda_3\, L_{out}(X). \end{aligned} \tag{11}$$

A_s and A_t in Equation 11 are collections of attention maps for the student and teacher, respectively. H_s and H_t are sets of hidden states for the student and teacher. L is the number of iterations (layers) of the recursive student. $g(.)$ is a mapping function that connects each iteration of the student to a specific layer of the teacher. Finally, λ_1 to λ_3 are hyperparameters used for controlling the importance of each component of the loss function (we use $\lambda_1 = 1.0$, $\lambda_2 = 3.0$, $\lambda_3 = 5.0$ in our experiments).

[1] Y^n is a zero vector if the n^{th} token is not masked.

[2] $\psi(\vec{u}, \vec{v}) = \frac{\vec{u}.\vec{v}}{||\vec{u}||_2 ||\vec{v}||_2}$

Table 1: The DEV set results on the GLUE benchmark. ALBERT_6 and ALBERT_{12} denote ALBERT models with 6 and 12 layers respectively, and an embedding size of 128, which are trained on the same data for the same number of training steps as MiniALBERT. DistilBERT_{base} is a DistilBERT model trained with the same distillation setting as Sanh et al. (2019) and for the same number of training steps as MiniALBERT. Adapter denotes layer-wise adaption and EF denotes Embedding Factorisation. The metrics used for reporting the results on each dataset is the same as the official GLUE benchmark. † denotes that the models were trained using adapter tuning in which all weights of the model except bottleneck adapters are kept frozen during fine-tuning. $*$ shows that the bottleneck adapters were initialised randomly since the model has not used bottleneck adapters at the pre-training stage. N/A means that the model did not learn anything from the target dataset. #Params denotes the number of tunable parameters which are used during fine-tuning. Note that for fine-tuning TinyBERT on the downstream tasks, unlike the original paper (Jiao et al., 2019), we do not employ task-specific distillation; this is to ensure the comparison with other models is fair.

Model	Adapter	EF	#Params	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
BERT-base	-	-	110M	**84.78/84.78**	**87.91**	**91.56**	**92.77**	**57.19**	**89.20**	**91.46**	**74.72**	**83.81**
DistilBERT	-	-	65M	82.17/82.33	87.08	89.47	90.25	53.61	87.20	88.66	66.42	80.79
MobileBERT	-	-	25M	84.03/83.84	87.41	91.17	91.16	51.55	88.12	90.59	66.78	81.62
TinyBERT	-	-	15M	80.22/80.65	84.44	85.31	88.87	17.53	87.77	89.41	66.06	75.58
DistilBERT_{base}	-	-	65M	**82.08/82.27**	**86.01**	86.91	**91.28**	**50.33**	83.92	88.06	59.56	**78.93**
ALBERT_6	-	✓	11M	76.35/77.01	84.73	84.51	86.12	29.54	82.76	86.39	59.56	74.10
ALBERT_{12}	-	✓	11M	78.93/80.03	85.45	**88.12**	86.92	41.73	**86.21**	**90.65**	**63.17**	77.91
MiniALBERT_{768}	×	×	31M	80.71/81.80	86.25	88.13	89.79	43.71	**86.93**	89.00	61.37	78.63
MiniALBERT_{312}	×	✓	17M	80.55/81.29	86.46	87.46	89.56	45.82	85.51	89.22	62.09	78.66
MiniALBERT_{768}	✓	×	32M	**81.04/82.05**	**86.48**	**88.79**	89.79	46.33	86.92	88.46	67.50	79.70
MiniALBERT_{312}	✓	✓	18M	80.78/81.67	86.35	88.57	**90.36**	45.50	86.60	88.96	**69.31**	79.78
MiniALBERT_{128}	✓	✓	12M	80.64/81.33	86.29	88.08	89.67	**47.93**	86.62	**89.77**	68.95	**79.92**
$\text{MiniALBERT}_{768}^{\dagger,*}$	×	×	0.9M	74.61/75.72	79.72	80.70	85.20	N/A	83.90	81.72	52.70	68.25
$\text{MiniALBERT}_{312}^{\dagger,*}$	×	✓	0.9M	74.48/75.70	79.76	82.39	83.48	N/A	80.94	81.22	54.15	68.01
$\text{MiniALBERT}_{768}^{\dagger}$	✓	×	0.9M	**79.48**/80.06	**85.29**	**87.84**	90.02	42.26	**86.33**	87.74	**67.14**	**78.46**
$\text{MiniALBERT}_{312}^{\dagger}$	✓	✓	0.9M	79.13/**80.16**	85.27	86.10	89.90	**45.25**	85.34	87.91	65.34	78.26
$\text{MiniALBERT}_{128}^{\dagger}$	✓	✓	0.9M	78.05/79.66	84.94	87.40	**90.36**	44.72	84.62	**89.08**	66.78	78.40

4 Experiments and Results

We evaluate our general models on the widely used GLUE benchmark (Wang et al., 2018). All the models were pre-trained on four Tesla V100 32GB GPUs with a total batch size of 192 (48 each) and fine-tuning was done using only one Tesla V100. A random seed of 42 was used consistently throughout training for fair comparison. For all of the datasets, in order to do full fine-tuning, we use a learning rate from the set {5e-5, 3e-5, 1e-5}. For large datasets (MNLI, QQP, QNLI, and SST-2), we train models for a maximum of 5 epochs, and up to ten epochs on other datasets. The hyperparameters used for full-finetuning are listed in Table 7.

For adapter-tuning, the learning rate was selected from the set {5e-5, 5e-4, 1e-3}. Models were trained for a maximum of 10 epochs for large datasets and up to 15 epochs for other datasets. Table 8 details the hyperparameters used during adapter tuning on the GLUE benchmark. The results of the baselines and our general models are available in Table 1.

The biomedical models are evaluated on the task of Named Entity Recognition (NER), which is one of the most prominent tasks in biomedical NLP. We use four well-known datasets, namely, NCBI-disease (Doğan et al., 2014), BC5CDR-disease (Li et al., 2016), BC5CDR-chem (Li et al., 2016), and BC2GM (Smith et al., 2008).

We generally follow the same pre-processing pipeline as Rohanian et al. (2022). For biomedical NER, we use pre-processed datasets from Lee et al. (2020). We fine-tune the models using a learning rate from the set {5e-5, 3e-5, 1e-5} and perform adapter-tuning with a learning rate from {5e-5, 5e-4, 1e-3}. The hyperparameters for the biomedical datasets are presented in Tables 9 and 10. The results of the baselines and our biomedical models are available in Table 2.

5 Discussion

We trained our recursive students both with and without adapters and found that generally having adapters would increase the learning capacity of the model since each iteration provides an extra degree of freedom to the model in order to capture a specific type of feature. As shown in Tables 1

Table 2: The results of biomedical NER for the biomedical baselines and our models distilled from BioBERT-v1.1 on the PubMed dataset. BioALBERT_6 and BioALBERT_{12} represent ALBERT models with 6 and 12 layers, respectively, and an embedding size of 128. These models were trained for the same number of steps as BioMiniALBERT using the same data. "Adapter" refers to layer-wise adaptation and "EF" stands for Embedding Factorization. The results are reported using the F1-score as the evaluation metric. The notations here are consistent with Table 1.

Model	Adapter	EF	#Params	NCBI-disease	BC5CDR-disease	BC5CDR-chem	BC2GM	Avg
DistilBioBERT	-	-	65M	87.93	85.42	94.53	86.60	88.62
CompactBioBERT	-	-	65M	**88.67**	85.38	94.31	86.71	88.76
TinyBioBERT	-	-	15M	85.22	81.28	92.20	82.52	85.30
BioMobileBERT	-	-	25M	87.21	84.62	94.23	85.26	87.83
BioBERT	-	-	110M	88.62	**86.67**	**94.73**	**87.62**	**89.41**
BioALBERT_6	-	✓	11M	**86.07**	**82.00**	**93.19**	**84.51**	**86.44**
BioALBERT_{12}	-	✓	11M	**86.07**	81.94	93.11	84.33	86.36
$\text{BioMiniALBERT}_{768}$	×	×	31M	87.44	84.40	94.18	86.06	88.02
$\text{BioMiniALBERT}_{312}$	×	✓	17M	87.94	84.45	94.03	86.03	88.11
$\text{BioMiniALBERT}_{768}$	✓	×	32M	88.02	**84.98**	**94.49**	86.10	**88.39**
$\text{BioMiniALBERT}_{312}$	✓	✓	18M	**88.03**	84.75	94.23	**86.14**	88.28
$\text{BioMiniALBERT}_{128}$	✓	✓	12M	87.16	84.58	94.20	86.00	87.98
$\text{BioMiniALBERT}_{768}^{\dagger,*}$	×	×	0.9M	85.61	82.31	92.80	84.89	86.40
$\text{BioMiniALBERT}_{312}^{\dagger,*}$	×	✓	0.9M	85.98	81.72	91.99	84.65	86.08
$\text{BioMiniALBERT}_{768}^{\dagger}$	✓	×	0.9M	**87.80**	**84.64**	94.20	**86.02**	**88.16**
$\text{BioMiniALBERT}_{312}^{\dagger}$	✓	✓	0.9M	87.61	84.55	94.12	85.60	87.97
$\text{BioMiniALBERT}_{128}^{\dagger}$	✓	✓	0.9M	87.71	84.48	**94.22**	85.87	88.07

and 2, in virtually all of the studies, models with adapters outperformed models of comparable size without adapters. In general, both models with and without adapter have outperformed their baseline by a significant margin (as shown in Tables 2 and 1), demonstrating the effectiveness of the proposed layer-to-layer distillation loss for constructing powerful compact recursive models. Comparison between the attention maps of our trained student and teacher models (Figure 3 in appendix) suggests the specific recursive architecture we have introduced in this work is indeed capable of mimicking the components of a larger model.

In addition, our experiments revealed that utilising adapters in the pre-training stage enables the model to effectively use adapter-tuning with only minor performance drops. However, adapter-tuning in models that have not used adapters in the pre-training stage causes major performance drops. We also found that adapter tuning is nearly 30% faster than full fine-tuning. Therefore, with adapter-tuning, the models can be trained for a larger number of epochs given the same training time, which can potentially increase the performance of the model. In general, a higher learning rate was required for adapter-tuning than for full fine-tuning. For both general and biomedical tasks we found that a learning rate of $5e-4$ results in the best performance, however, in some cases, a higher or lower learning rate such as $5e-5$ or $1e-3$ was deemed better.

Another method explored in this work for parameter saving is Embedding Factorisation. In our experiments, we observed that regardless of drastic parameter reduction, models using this approach are still able to perform on par or even in some cases better than models with full-rank embedding.

6 Ablation Studies

6.1 The Effect of The Alignment Loss

One of the main losses explored in this work for knowledge distillation is the alignment loss as explained in Equation 9. Alignment loss is used for mapping each iteration of the recursive student to a specific layer of the teacher, so the knowledge of each fully-parameterised layer is explicitly encoded into a specific iteration of the recursive student. The alignment loss consists of two losses, one for aligning the attention maps and one for aligning the hidden states. For ablation studies, we trained our best model for 20k steps on the Wikipedia dataset, with different alignment losses, and evaluated the resulting models on the GLUE dataset. The results are shown in Table 3.

As Shown in Table 3, without alignment, the performance of the recursive student drops significantly which shows the importance of using a layer-to-layer distillation technique. Furthermore, we discovered that aligning hidden states is far more

Table 3: Ablation study on the alignment loss. The performance drop is computed based on the difference with the average score of the model with full alignment (i.e. when all the alignment losses are used).

Alignment Type	Performance Drop
Hidden-Only	-0.5
Attention-Only	-3.27
No-Alignment	-5.27

important than aligning attention maps. We even noticed occasional improvements in some tasks by solely aligning hidden states, but the average performance of full alignment remains higher than hidden states alignment.

6.2 The Effect of an Extra Embedding Loss

Following the work of Jiao et al. (2019) and as part of our ablation tests, we investigated employing an extra loss for aligning the embeddings of the recursive student and fully-parameterised teacher. This embedding loss is defined as follows:

$$L_{embed}(\hat{E}, E) = \frac{1}{N}\sum_{n=1}^{N} 1 - \psi(\hat{E}^n, E^n) \quad (12)$$

$\hat{E}$ and E are the inputs to the loss function and represent the embeddings of the student and teacher. The embedding weights are not aligned globally in this formulation; instead, the local embeddings created for each training sample are compared before entering the transformer encoder. For our ablation studies, we trained models on the Wikipedia dataset for 20k steps, with and without this extra alignment loss, and evaluated them on the GLUE benchmark.The average performance of the models are reported in Table 4.

Table 4: Ablation study on the embedding loss. 'Full-rank' denotes models without embedding factorisation, and 'Loss' denotes extra embedding loss during distillation.

Model	Avg Performance
Full-rank	76.01
Full-rank + Loss	75.24
Factorised	76.31
Factorised + Loss	76.14

We discovered that, unlike the trend seen in fully-parameterised models, embedding loss reduces the performance of the student model (Table 4). This difference between the recursive student and the fully parameterised teacher implies that forcing the recursive student's embedding to match that of the fully parameterised teacher may not be beneficial, and that allowing the student to learn embeddings independently works better.

7 Conclusion and Future Works

In this work, we explored distilling the knowledge of fully-parameterised language models into recursive students with cross-layer parameter sharing. We used a layer-to-layer distillation technique to observe the learning capacity of the recursive students compared to their fully-parameterised teachers. We used bottleneck adapters for improving the performance of our recursive students and also assessed the benefits of adapter-tuning at the fine-tuning stage. Furthermore, an embedding factorisation technique was used for additional parameter reduction, which was evaluated with and without an extra distillation loss to match the student's embeddings with the teacher. Finally, by integrating all of the strategies outlined above, we were able to train compact recursive students with no more than 12M parameters, yielding competitive performance on both general and biomedical NLP. In the future, we hope to investigate various parameter-sharing and embedding factorisation strategies, as well as other layer-wise adaption techniques such as prompt-tuning. We would also like to train recursive students with larger hidden sizes and more training iterations to compress massive fully-parameterised models with minor performance drops.

Funding

This work was supported in part by the National Institute for Health Research (NIHR) Oxford Biomedical Research Centre (BRC), and in part by an InnoHK Project at the Hong Kong Centre for Cerebro-cardiovascular Health Engineering (COCHE). OR acknowledges the support of the Medical Research Council (grant number MR/W01761X/). DAC was supported by an NIHR Research Professorship, an RAEng Research Chair, COCHE, and the Pandemic Sciences Institute at the University of Oxford. The views expressed are those of the authors and not necessarily those of the NHS, NIHR, MRC, COCHE, or the University of Oxford.

Limitations

Regardless of the parameter reduction induced by our proposed recursive architecture, the resultant models have the same inference latency and memory complexity as fully-parameterised models of comparable size, which for our models is DistilBERT.

In general, our models' capacity for learning semantic and grammatical knowledge is limited, and they may perform poorly on tasks that necessitate a significant amount of reasoning and understanding, such as Question Answering or Semantic Acceptability. More analysis is required to determine the source of this limitation, i.e. whether it is a result of the architecture used or a consequence of the particular cross-layer sharing method etc.

Ethics Statement

In this study, we aimed to create efficient lightweight versions of large and less accessible NLP models. This area of research aims to make AI/NLP models more readily available, with fewer computational resources required to run them and potentially less negative environmental impact.

This work does not use any private or sensitive data and instead relies on widely used publicly available datasets that have been utilised by other researchers in the field with references provided in the paper for more information. All the codes and models are going to be made available for reproducibility purposes.

References

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

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

Tao Ge and Furu Wei. 2022. Edgeformer: A parameter-efficient transformer for on-device seq2seq generation. *arXiv preprint arXiv:2202.07959*.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2208–2222, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Cite arxiv:1503.02531Comment: NIPS 2014 Deep Learning Workshop.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019a. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019b. Parameter-efficient transfer learning for nlp.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. 2022. On the computational complexity of self-attention. *arXiv preprint arXiv:2209.04881*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Omid Rohanian, Mohammadmahdi Nouriborji, Samaneh Kouchaki, and David A Clifton. 2022. On the effectiveness of compact biomedical transformers. *arXiv preprint arXiv:2209.03182*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Larry Smith, Lorraine K Tanabe, Cheng-Ju Kuo, I Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, Manabu Torii, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):1–19.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Patrick Xia, Shijie Wu, and Benjamin Van Durme. 2020. Which *BERT? A survey organizing contextualized encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7516–7533, Online. Association for Computational Linguistics.

A Details of the Pre-Training

A.1 General Models

For pre-training our general models we use the English subset of the Wikipedia dataset which is available on the Huggingface platform. For preprocessing the Wikipedia dataset, we use the 'bert-base-uncased' tokeniser and apply a sliding window with a size of 256 tokens and a stride size of 128. Due to computational restrictions, we limit the maximum number of tokenised samples per article to 10, resulting in a total of 21 million training samples of 256 tokens each. We then follow BERT's

masking approach, with a masking probability of 15% for each token.

Table 5: Hyperparameters used for pre-training models on Wikipedia dataset

Param	Value
learning rate	$5e-4$
scheduler	Linear
optimiser	AdamW
weight decay	$1e-4$
total batch size	192
warmup steps	5000
epochs	1

A.2 Biomedical Models

We pre-trained the biomedical models using the PubMed Abstracts dataset which consists of 31 million abstracts from PubMed articles. In the pre-processing stage, we employed the 'bert-base-cased' tokeniser with a maximum length of 256 and adhered to the same masking strategy as used in the training of the general models.

Table 6: Hyperparameters used for pre-training models on Wikipedia dataset

Param	Value
learning rate	$5e-4$
scheduler	Linear
optimiser	AdamW
weight decay	$1e-4$
total batch size	192
warmup steps	5000
training steps	100K

B Finetuning Details

C Comparison of Teacher and Student Attention Maps

Figure 3 shows attention maps of a teacher model and one of the proposed students on the input sentence "This is the first book I've ever done". The first and second rows of the student contain four attention heads belonging to the 0^{th} and 4^{th} iteration of the recursive student, respectively. The first and second row of teacher contain attention heads belonging to the 1^{st} and 9^{th} layer of the teacher.

Table 7: Hyperparameters used for full fine-tuning of the models on the GLUE benchmark

Param	Value
learning rate	{5e-5, 3e-5, 1e-5}
scheduler	Linear
optimiser	AdamW
weight decay	0.01
batch size	{8, 16, 32}
epochs	{3, 5, 10}

Table 8: Hyperparameters used for adapter-tuning of the models on the GLUE benchmark

Param	Value
learning rate	{5e-5, 5e-4, 1e-3}
scheduler	Linear
optimiser	AdamW
weight decay	0.01
batch size	{8, 16, 32}
epochs	{5, 10, 15}

Table 9: Hyperparameters used for full fine-tuning of the models on the Biomedical datasets

Param	Value
learning rate	{5e-5, 3e-5, 1e-5}
scheduler	Linear
optimiser	AdamW
weight decay	0.01
batch size	16
epochs	5

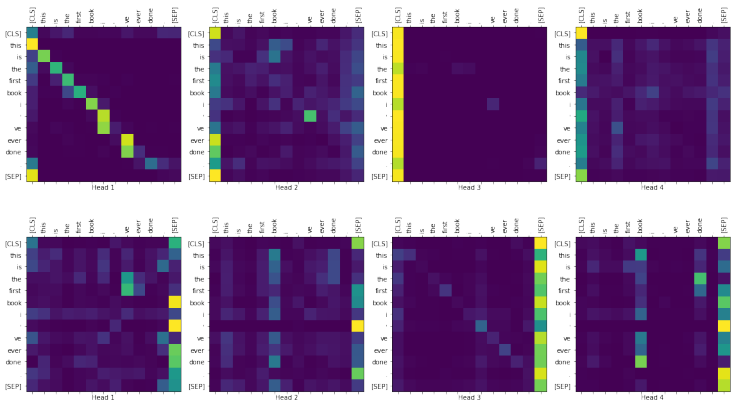
Table 10: Hyperparameters used for adapter tuning of the models on the Biomedical datasets

Param	Value
learning rate	{5e-5, 5e-4, 1e-3}
scheduler	Linear
optimiser	AdamW
weight decay	0.01
batch size	16
epochs	{5, 10}

During training, these sets of layers have been compared together in order to compute the alignment

loss. Despite the fact that the recursive student has only one layer, it has been able to perfectly mimic its teacher in some of the heads which shows the efficiency of the layer-to-layer distillation loss.

Teacher



Student

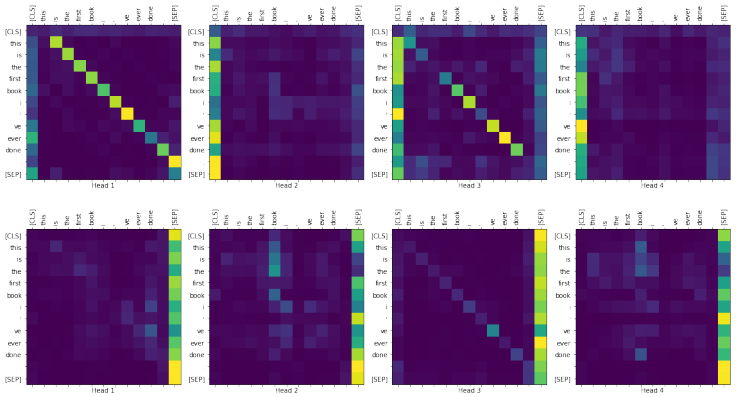


Figure 3: The attention maps for the teacher and one of the proposed recursive students on the input “This is the first book I’ve ever done”.