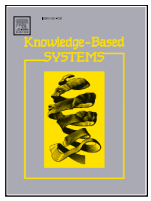
MAdaKron: a Mixture-of-AdaKron Adapters

Knowledge-Based Systems 334 (2026) 115086

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



MAdaKron: A mixture-of-AdaKron adapters

Marco Braga [a,b,*], Alessandro Raganato [a], Gabriella Pasi [a,*]

[a] University of Milan-Bicocca, Milan, Italy
[b] Politecnico di Torino, Turin, Italy

ARTICLE INFO

Keywords:
Large language models
Adapters
Mixture of experts

ABSTRACT

Adapting pre-trained Large Language Models to specific tasks has traditionally involved updating all of their parameters. Nonetheless, this technique becomes impractical for models containing billions of parameters. This has led to intensive research on Parameter-Efficient Fine-Tuning (PEFT) techniques, which aim to train a small fraction of the model's parameters while maintaining comparable performance to Full Fine-Tuning. A popular method is the Adapter, i.e. small trainable layers added to pre-trained models. Recently, we present AdaKron, an Adapter-based PEFT technique, which leverages the Kronecker product to combine the outputs of two small networks, training less than 0.55 % of the model's parameters while outperforming Full Fine-Tuning. In this paper, we put forward a novel technique, called MAdaKron, a Mixture-of-AdaKron model, which combines AdaKron with a Mixture of Experts approach. MAdaKron combines the flexibility of a Mixture of Experts architecture with the efficiency given by AdaKron to further enhance its performance. We then extensively evaluate MAdaKron on eighteen Natural Language Understanding and Generation benchmarks, showing that it achieves performance on par or even better than recent state-of-the-art PEFT methods, while reducing the number of trainable parameters. These findings highlight MAdaKron as an efficient solution for Fine-Tuning LLMs, offering substantial computational cost reductions without losing performance.

1. Introduction

Fine-Tuning neural models for specific tasks traditionally requires training all model parameters [1,2]. However, with Large Language Models (LLMs) characterized by billions of parameters [3], this conventional approach demands substantial memory and computational resources, making it impractical. This has led to the development of new methods, usually referred to as Parameter Efficient Fine-Tuning (PEFT) techniques, focusing on training only a small subset of model parameters while maintaining performance comparable to traditional Fine-Tuning methods. In this work, we focus specifically on Adapters [4], a class of PEFT techniques that have proven effective across a wide variety of tasks [5]. Adapters generally consist of two fully connected layers: an initial layer, the so-called down projection, that maps the input vector into an intermediate dimension, followed by a non-linear activation function, and by an up projection to the model's hidden dimension [6]. The performance of an Adapter largely depends on its intermediate dimension: larger dimensions typically yield better results, even though recent studies [7] suggest that a high number of trainable parameters can introduce instability during training. Based on the above findings, to reduce adapter parameters while maintaining performance, we recently proposed AdaKron [8], which incorporates the Kronecker product into an Adapter. In contrast to recent works [9], which decompose weight matrices into smaller low-dimensional ones through the Kronecker product, AdaKron employs the Kronecker product between the output vectors of two Feed-Forward Networks (FFNs), which compose the down projection of the Adapter. The Kronecker product produces a higher-dimensional vector by scaling one input vector with each element of the other. We interpret the Kronecker product as heuristically analogous to a Mixture of Experts (MoE) system [10], where the first vector acts as a gating mechanism (the weights) that scales the second vector (the expert output vector). However, unlike traditional MoE systems, which combine expert outputs through a weighted sum, the Kronecker product concatenates the scaled versions of the second vector, preserving all individual contributions distinctly. This approach is inherently parameter-efficient, as it does not require learning separate expert outputs.

In order to show that AdaKron can be seamlessly integrated with state-of-the-art PEFT approaches, we introduce a novel method, MAdaKron, which combines AdaKron within a MoE framework, enhancing its performance and flexibility. MAdaKron transforms the heuristic MoE analogy of AdaKron into an explicit architecture. In MAdaKron, the weights vector of AdaKron is dynamically generated through a Mixture of Experts system, which allows the model to adaptively

* Corresponding authors.
E-mail addresses: m.braga@campus.unimib.it (M. Braga), alessandro.raganato@unimib.it (A. Raganato), gabriella.pasi@unimib.it (G. Pasi).

https://doi.org/10.1016/j.knosys.2025.115086
Received 8 September 2025; Received in revised form 24 November 2025; Accepted 4 December 2025
Available online 6 December 2025

modulate the Kronecker interaction according to the input, effectively enabling multiple expert adaptation behaviours within a single AdaKron layer. The resulting output remains a weighted concatenation, retaining the structural efficiency of AdaKron, but the weighting is context-dependent, enhancing expressiveness and flexibility without increasing the number of trainable parameters. We evaluate MAdaKron across eighteen public Natural Language Understanding and Generation datasets using eight different Pre-trained Language Models (PLMs). By applying MAdaKron and AdaKron while Fine-Tuning an LLM to a downstream task, we achieve a reduction of approximately one-third in the number of trainable parameters while achieving superior performance with respect to original Adapters [11] with the same intermediate dimension.

Specifically, the novel contributions are as follows: (i) we extend evaluation of AdaKron with a total of three new tasks across ten new benchmarks and five new PLMs; (ii) we propose a novel method, MAdaKron, which merges AdaKron with a Mixture of Experts approach, significantly improving its performance; (iii) we conduct a comprehensive evaluation of MAdaKron on eighteen public Natural Language Processing datasets, spanning both Natural Language Understanding and Generation tasks. MAdaKron is tested on PLMs of different sizes, outperforming recent state-of-the-art PEFT methods while training less than 0.55 % of the model's original parameters. (iv) we perform an in-depth analysis of each component of MAdaKron to justify our design choices. In addition, we publicly release the code for the proposed approaches, along with scripts for running all evaluations[1] to promote reproducibility and facilitate further comparisons of PEFT methods.

The remainder of this paper is organized as follows. Section 2 reviews the related work. In Section 3, we detail AdaKron and our novel MAdaKron approaches, discussing the corresponding design choices. Section 4 outlines the evaluation framework. Then, the experimental results and performance of our proposed approaches across various tasks are shown in Section 5, followed by an ablation study assessing the impact of our design choices in Section 5.1.

2. Related works

In this section, we provide a general overview of Parameter-Efficient Fine-Tuning techniques and the Mixture of Experts approach.

PEFT Approaches. Parameter Efficient Fine-Tuning (PEFT) techniques are specifically designed to train only a subset of the model's original parameters while adapting the model to a downstream task, and maintaining performance levels comparable to those achieved with traditional Fine-Tuning. One prominent PEFT approach is the Adapters, which have a bottleneck architecture composed of two linear layers and an internal residual connection; they are inserted into each Transformer layer after the Attention layer and after the Feed-Forward Network (FFN) layers, as in the Houlsby Adapter [6], or just only after the FFN layers as in the Pfeiffer Adapter [11]. Other PEFT approaches include BitFit [12], which tunes only bias terms; LoRA-based methods [13–18], which constrain weight updates to low-rank matrices; $(IA)^3$ [19], which rescales input vectors with task-specific parameters; and UNIPELT [20], which combines multiple PEFT strategies. Recent work has also explored more compact adapter formulations such as Compacter [9].

Mixture of Experts. The first large-scale success of Mixture of Experts (MoE) in Deep Learning, the so-called Sparse Mixture of Experts [10], defines a MoE layer between two LSTM layers. Each MoE layer consists of a set of n experts $E_1, \dots, E_n$, which are small neural networks, and a Gating network $\mathbb{G}$, whose output is a n-dimensional vector. Let us denote x the input vector and $E_i(x)$ and $\mathbb{G}(x)$ the output of the ith expert and the Gating network, respectively. The output of the MoE layer can be summarized as $\sum_{i=1}^{n} \mathbb{G}(x)_i E_i(x)$. Inspired by recent work [21], AdaMix [15] defines a MoE layer in each down projection layer of an Adapter with a random gate function, achieving state-of-the-art performances on GLUE [22].

3. Methodology

In this section, we first briefly present the Kronecker product and AdaKron. Next, we describe in detail our new proposed approach, MAdaKron.

Background: the Kronecker Product. Given two matrices, $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, the Kronecker product $A \otimes B$ is a block matrix of dimension $m \cdot p \times n \cdot q$, where each block is the product between an element of A and the entire matrix B. When applied to vectors, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, the Kronecker product yields

$$y \otimes x = vec(xy^T) = \begin{bmatrix} y_1 x, & \dots & , y_m x \end{bmatrix}, \tag{1}$$

which is a $n \times m$-dimensional vector and a weighted concatenation of vector x with weight values from vector y. For example, given $x = [1, 2, 3] \in \mathbb{R}^3$ and $y = [2, 5] \in \mathbb{R}^2$, then $y \otimes x = [2 \cdot [1, 2, 3], 5 \cdot [1, 2, 3]] = [2, 4, 6, 5, 10, 15] \in \mathbb{R}^6$, which is a concatenation of two vectors, i.e. $2 \cdot [1, 2, 3]$ and $5 \cdot [1, 2, 3]$, which are both vector x multiplied by a value from y. Furthermore, $y \otimes x \in \mathbb{R}^6$, where 6 is the product of the dimension of x, i.e. 3, and y, i.e. 2.

Background: AdaKron. AdaKron, which we introduced in [8], incorporates the Kronecker product within an Adapter module, which is positioned between the output vectors of two FFNs that form the down projection layer of the Adapter. Following Pfeiffer et al. [11], we add our Adapter only after the Feed-Forward Networks in each Transformer layer. AdaKron consists of two down projections and one up projection FFNs. This design choice is informed by recent studies [5], which indicate that reducing the up projection can negatively impact the Adapter's performance. We define two different down projection layers D_v and D_c, whose final outputs are $y_v = D_v(x) = \mathbf{W}_v x + b_v$ and $y_c = D_c(x) = \mathbf{W}_c x + b_c$, where $\mathbf{W}_v \in \mathbb{R}^{r_1 \times d}$, $\mathbf{W}_c \in \mathbb{R}^{r_2 \times d}$, $b_v \in \mathbb{R}^{r_1}$, $b_c \in \mathbb{R}^{r_2}$, $y_v \in \mathbb{R}^{r_1}$, $y_c \in \mathbb{R}^{r_2}$, $r_1, r_2 \ll d$ and x is a d-dimensional input. Next, we apply the Kronecker product and the GELU function to define the output of the down projection layers $g(y_c, y_v) \coloneqq GELU(y_c \otimes y_v) \in \mathbb{R}^{r_1 \cdot r_2}$, which is then fed to the up projection layer. Thus, the final intermediate dimension of our Adapter is $r = r_1 \cdot r_2 \ll d$ and, as shown in Eq. (1), the output of the down projections is a weighted concatenation of the output of layer D_v with the weights from the output vector from layer D_c.

MAdaKron. Building on the heuristic analogy between AdaKron and a Mixture of Experts (MoE) mechanism, we introduce MAdaKron, a Mixture of Experts AdaKron that transforms this intuition into an explicit architecture. In AdaKron, the Kronecker product implicitly behaves like a gating interaction: one vector scales the other, such as a gating network modulates expert activations in an MoE system. MAdaKron extends this idea by making the weights selection dynamic and input-dependent, explicitly incorporating an MoE architecture into AdaKron. In MAdaKron, the weight vector that modulates the Kronecker product is generated through a Mixture of Experts system, allowing the model to adaptively learn the weight representations. This design can be viewed as a natural evolution of AdaKron: while AdaKron heuristically mimics the behaviour of an expert system without explicitly instantiating one, MAdaKron realises this mechanism in full, enabling adaptive expert selection while maintaining the same parameter efficiency. We introduce two variations of MAdaKron, Partial and Full MAdaKron, that differ in how the MoE structure is integrated within the Adapter. As shown in Fig. 1, in the Partial MAdaKron configuration, only the down projection D_c is replaced with a MoE layer, which is responsible for generating the weights used for the weighted concatenation. In contrast, as shown in Fig. 2, in the Full MAdaKron configuration, both down projection layers

[1] https://github.com/DetectiveMB/AdaKron

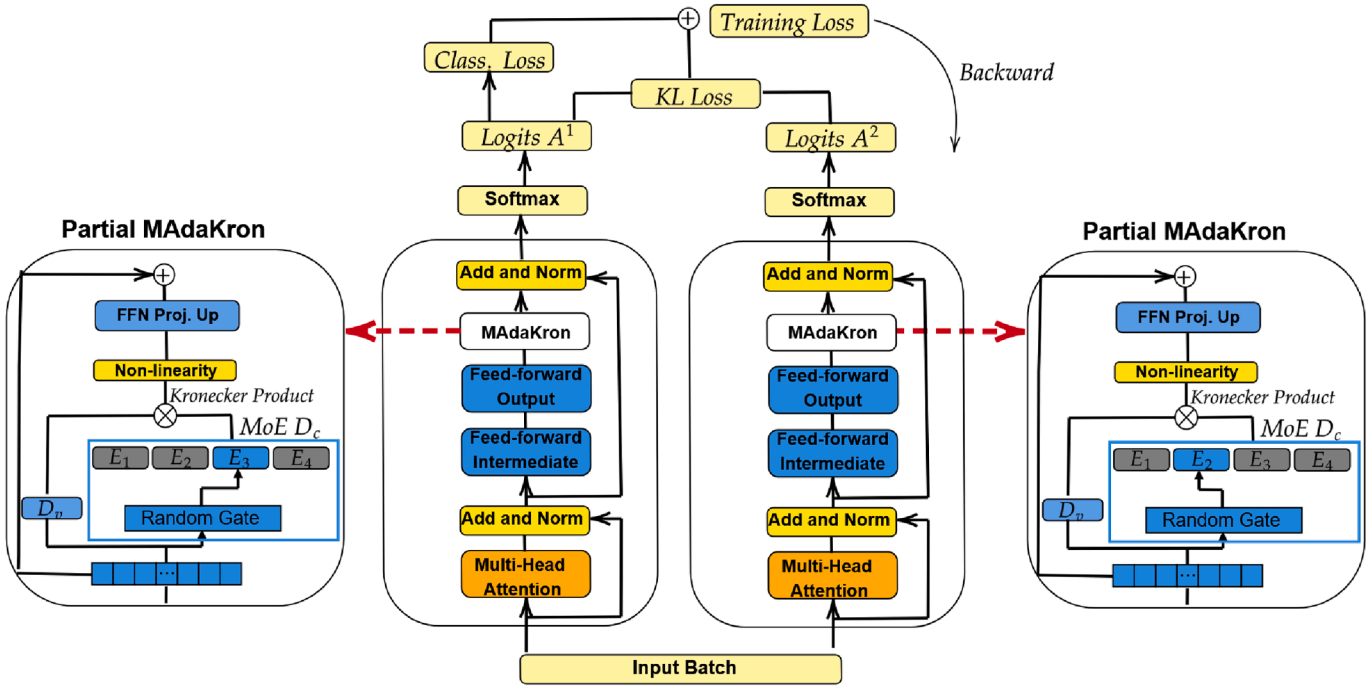


Fig. 1. Architecture of a Transformer layer with the integration of Partial MAdaKron. During each training step, MAdaKron requires two forward passes per batch, while selecting two expert configurations. Finally, we apply a Classification Loss (*Class. Loss*) and a Consistency loss based on the Kullback-Leibler divergence (*KL Loss*) between the output probabilities to ensure consistency between the two forward passages.

in AdaKron are replaced with MoE layers. For each input, the model randomly selects two experts, one from each MoE down projection layer, and their output vectors are combined using the Kronecker product. Since recent work [5,21] has shown that a stochastic routing function can perform on par or even better than learned routing mechanisms without requiring additional parameters, in both our approaches, a random routing function determines which experts are activated for each input. Furthermore, the gating function is no longer a neural network, which means no additional parameters or computation are required for expert selection, which is particularly important because it keeps MAdaKron's parameter count equal to AdaKron's, while still adding the flexibility and adaptability of the Mixture of Experts approach. Different from recent works [5], however, the experts in MAdaKron emerge from the Kronecker-based parameterisation of AdaKron, which induces implicit experts in a parameter-efficient way. By integrating these implicit experts with explicit, adaptive selection, MAdaKron combines the efficiency and compactness of AdaKron with the specialisation benefits of dynamic expert selection, yielding a model that is more expressive than AdaKron. Fig. 1 shows how the training of MAdaKron and the stochastic routing function work: the model processes each batch twice, each time selecting different expert configurations.

In detail, Let n be the number of Transformer layers and $k = 1, 2$ the indices of the two batch passes. For each layer i the selected expert in pass k is denoted $A^k = \{A_1, \dots, A_n\}$. For a given input x, the passage and routing through A^k produces logits $z_{(.)}^{A^k}(x)$. The two forward passes thus yield $z_{(.)}^{A^1}(x)$ and $z_{(.)}^{A^2}(x)$, as shown in Fig. 1. To ensure consistency, we balance the output probabilities of each batch iteration by using a consistency loss based on the Kullback-Leibler divergence [23]. The training loss is defined as $\mathcal{L} = -(\mathcal{L}_{Class.} + 1/2 * (\mathcal{L}_{KL}))$, where $\mathcal{L}_{Class.}$ is the classification loss, i.e. it indicates if the predicted label is the correct classification, and $\mathcal{L}_{KL}$ is the symmetric Kullback-Leibler divergence (KL) between the probabilities given by the two training iterations with the same batch. Given x as input:

$$\mathcal{L}_{Class} = \sum_{c=1}^{C} \mathcal{I}(x, c) log(softmax(z_c^{A^1}(x)), \tag{2}$$

$$\mathcal{L}_{KL} = KL(z_{(.)}^{A^1}(x) || z_{(.)}^{A^2}(x)) + KL(z_{(.)}^{A^2}(x) || z_{(.)}^{A^1}(x)), \tag{3}$$

where $\mathcal{I}(x, c) = 1$ if c is the correct label for x, and 0 otherwise. Finally, the matrix weights of the expert are merged during the inference stage [5] and each expert layer collapses into a single FFN layer.

Parameter Efficiency of MAdaKron. In this section, we compare the number of parameters of MAdaKron with Adapters. Instead of a single down projection layer with an output dimension of r, MAdaKron is composed of two Feed-Forward Networks (FFNs) with output dimensions equal to r_1 and r_2, where $r_2 \leq r$ is a reduction factor that corresponds to the number of weighted repetitions of the r_1-dimensional vector. Thus, the number of parameters in MAdaKron is determined by matrices $W_v \in \mathbb{R}^{r_1 \times d}$ and $W_c \in \mathbb{R}^{r_2 \times d}$. Assuming, without loss of generality, that $r_1 > r_2$, it follows that $r_1 = r/r_2$, where r_1 must be a natural number, implying that r_2 must be a divisor of r. Given d the input dimensionality of the network, the number of parameters in the down projection layer in a Pfeiffer Adapter [11], including biases, is $r \cdot d + r$, which we reduce to:

$$d \cdot (\frac{r}{r_2} + r_2) + \frac{r}{r_2} + r_2 = (d + 1) \cdot (\frac{r}{r_2} + r_2). \tag{4}$$

Finally, adding the parameter of the up projection layer, the total number of parameters is

$$(d + 1) \cdot (\frac{r}{r_2} + r_2) + r \cdot d + d \tag{5}$$

into an MAdaKron layer and $2rd + r + d$ into a Pfeiffer Adapter. The ratio between an MAdaKron and Pfeiffer adapter is:

$$\frac{(d + 1)(\frac{r}{r_2} + r_2 + r)}{2rd + r + d}. \tag{6}$$

Under the assumption that $d >> r >> r_2$, the term $d\left(r + \frac{r}{r_2} + r_2\right)$ dominates the numerator in Eq. (6), and $2rd + d$ dominates the denominator. This condition is typically satisfied for parameters commonly used in adapters [6], MAdaKron layers [8], and large-scale PLMs such as BERT [1] and RoBERTa [24], which have a hidden size of $d = 768$ and adapter

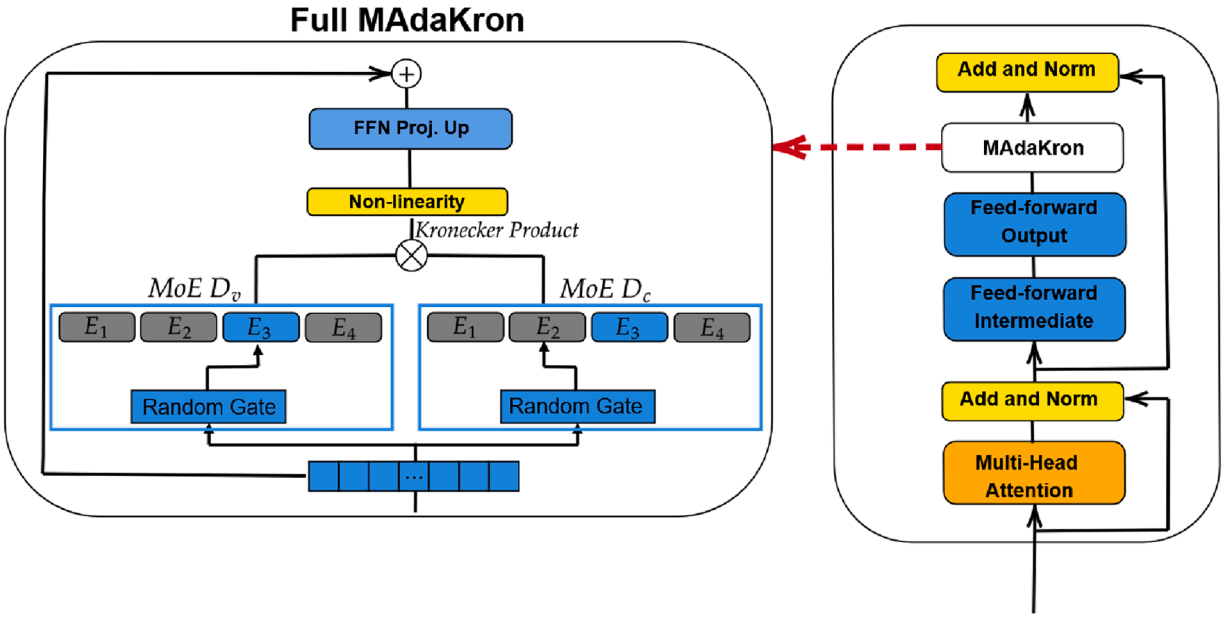


Fig. 2. Architecture of a Transformer layer with the integration of Full MAdaKron. Each MAdaKron D_c and D_v layer is defined as a MoE system. For illustration, we show a layer with 4 experts.

intermediate dimensions $r \in \{16, 32, 48, 64\}$. Thus, Eq. (6) can be approximated as:

$$\frac{(d+1)(\frac{r}{r_2}+r_2+r)}{2rd+r+d} \approx \frac{(r+\frac{r}{r_2}+r_2)d}{d(2r+1)} = \frac{r+\frac{r}{r_2}+r_2}{2r+1}. \tag{7}$$

As we will show in Section 5.1, r_2 is a hyperparameter of the model, whose optimal value is $r_2 = 4$ and substituting it into Eq. (7) gives:

$$\mathcal{R} = \frac{r+\frac{r}{r_2}+r_2}{2r+1} = \frac{r+\frac{r}{4}+4}{2r+1} = \frac{5r+16}{8r+4}. \tag{8}$$

Since $r \in \{16, 32, 48, 64\}$ are common values for the hidden adapter dimension, the ratio $\mathcal{R}$ is always less than 1, indicating that MAdaKron requires training fewer parameters with respect to a Pfeiffer Adapter. Furthermore, substituting the values of r into Eq. (8) shows that the ratio of trained parameters ranges between 0.72 and 0.65, meaning that MAdaKron allows us to train 30 % fewer parameters compared to a Pfeiffer Adapter.

Motivation and Relations with other PEFT approaches. Both of our approaches, AdaKron and MadaKron, rely on Adapters [6]. This differs from recent PEFT approaches, such as prefix- and prompt-tuning [25], because Adapters introduce learnable parameters after the feed-forward layers, whereas prefix- and prompt-tuning modify the embedding layer. The Kronecker product has recently emerged as a promising tool for enhancing Parameter-Efficient Fine-Tuning techniques by reducing both the number of trainable parameters and Floating Point Operations (FLOPs) [9,16]. In most prior work, the Feed-Forward Network weight matrices are decomposed into smaller submatrices that are recombined using the Kronecker product to approximate the full parameter matrix. For example, Compacter [9] applies Kronecker decomposition to adapter weights, and Krona [16] applies the same principle to attention weight matrices. Kronecker-based compression methods have also been successfully applied to transformer models such as BERT [26] and GPT-2 [27]. One appealing property of the Kronecker product is its ability to construct high-dimensional structures from compact components. Given matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, their Kronecker product $A \otimes B$ results in a matrix of shape $mp \times nq$. Thus, if a large parameter matrix $W \in \mathbb{R}^{mp\times nq}$ can be expressed as $A \otimes B$, we can learn the parameters in A and B, which together contain significantly fewer elements than W. Unlike other decomposition methods, such as low-rank matrix product, the Kronecker product imposes no rigid dimensional constraints on the factorized submatrices, enabling more flexible architectural design. Furthermore, the Kronecker product is a specific instantiation of the tensor product for matrices. It corresponds to the matrix representation of the tensor product of linear functions. Let $S : V \to X$ and $T : W \to Y$ be two linear functions, represented by matrices A and B, respectively, over vector spaces V, X, W, Y. Then, the Kronecker product $A \otimes B$ represents the tensor product function: $S \otimes T : V \otimes W \to X \otimes Y$. An important structural property of Kronecker products is their behaviour under matrix rank: $\text{rank}(A \otimes B) = \text{rank}(A) \times \text{rank}(B)$, which implies that the rank of the product is preserved multiplicatively. This characteristic supports efficient learning under constrained parameter budgets, without reducing the rank of the matrices. Beyond parameter savings, the Kronecker product can also contribute to computational efficiency. Edalati et al. [16] show that it is possible to avoid explicit reconstruction of the full Kronecker matrix during inference, by using the following identity: $(A \otimes B)x = \gamma\left(B, \eta_{b_2\times a_2}(x), A^\top\right)$, where $x \in \mathbb{R}^d$, $\eta_{b_2\times a_2}(y)$ is a mathematical operation that reshapes a vector $y \in \mathbb{R}^{b_2 a_2}$ in a matrix of size $b_2 \times a_2$ and $\gamma(Y)$ converts a matrix $Y \in \mathbb{R}^{a_2\times b_2}$ into a vector by stacking its columns. This formulation enables reducing FLOPs by avoiding the full matrix multiplication, further accelerating inference in low-resource settings. Furthermore, recently, Yu et al. propose MoKA (Mixture of Kronecker Product Adaptation) [17], which integrates a Mixture of Experts architecture with Kronecker-based LoRA updates, where each expert represents a Kronecker factor pair dynamically activated through an efficient sparse router, enabling conditional computation and high-rank adaptability. In contrast, Sadeghi et al. proposed MoKA (Mixture of Kronecker Adapters) [18] employs a learnable gating mechanism to weight multiple Kronecker adapters, achieving rank flexibility. Both MoKAs are LoRA-based approaches; thus, they update the attention parameters through a low-rank Kronecker product of learnable matrices. In contrast to previous methods that apply Kronecker products over full parameter matrices or only to the attention weights, our approach leverages the Kronecker product at the vector level. Specifically, we apply the Kronecker product to the output vectors of two parallel down projection layers within the Feed-Forward Networks. This strategy not only reduces the number of trainable parameters but also enables structured and expressive representations in the projected space.

Both AdaKron and MAdaKron adopt this design by composing the Kronecker product over the projected vectors. Furthermore, as shown in Eq. (1), the Kronecker product between vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ can be expressed as $x \otimes y = vec(xy^T)$, where xy^T denotes the outer product (also referred to as their tensor product), and $vec(\cdot)$ is the operator that flattens a matrix into a vector by stacking its columns. Since both x and y are one-dimensional, the outer product matrix xy^T has rank 1, providing a highly compressed representation that is nonetheless expressive. Importantly, the Kronecker product of two vectors can also be interpreted as a weighted concatenation. Specifically, if $x = [x_1, x_2, \dots, x_n]$ and $y = [y_1, y_2, \dots, y_m]$, then $y \otimes x = [y_1 x, \dots, y_m x]$, which corresponds to the concatenation of scaled copies of y, where each copy is weighted by the corresponding element of x. This structure allows the Kronecker product to behave similarly to a Mixture of Experts (MoE) gating mechanism, where each output expert (copy of y) is modulated by a gate value (element in x). However, unlike traditional MoE models that require explicit gating networks and weighted summations over experts, the Kronecker formulation yields a parameter-efficient implicit mixture, computed via a single outer product operation. Therefore, our use of the Kronecker product not only reduces parameter count but can also be heuristically interpreted as an efficient substitute for a soft-gated expert system. This compact and differentiable formulation facilitates smooth learning dynamics and implicit routing without introducing the overhead typically associated with MoE architectures. Different from Adapterfusion [11], which combines the output of task-specific adapters through a new learnable layer during inference, we average the weights of the experts inspired by recent model merging approaches [5].

4. Experimental setup

The experiments reported in the next Section aim to answer the following research questions: (**RQ1**) Is AdaKron more effective and efficient compared to traditional Fine-Tuning and existing Parameter-Efficient Fine-Tuning (PEFT) methods? (**RQ2**) Does incorporating a Mixture of Expert system into an AdaKron Adapter (i.e., MAdaKron) improve the model's performance? (**RQ3**) Is MAdaKron more effective than previously proposed PEFT approaches?

To address the above research questions, we perform a comparative evaluation of both the effectiveness and the efficiency of different PEFT methods. In the following sections, we present the datasets we employ for conducting our evaluations, describe the selected baselines and the evaluation metrics used to assess model effectiveness, and finally, we outline the training setup.

Datasets and Baselines. We evaluate AdaKron and MAdaKron on GLUE [22] and SuperGLUE [28] benchmarks, which include eight and five Natural Language Understanding tasks, respectively. Following standard practice [12,13], we used the development set as test data in both GLUE and SuperGLUE since the original test set is hidden. We further evaluate our approaches on two Question-Answering (QA) datasets, i.e. SQuAD v1.1 [29] and v2.0 [30], and conduct experiments on three Natural Language Generation tasks, i.e. E2E [31] WebNLG [32] and DART [33]. Following prior studies [5,13], we rely on task-specific metrics to evaluate our models, such as Accuracy, F1 and Matthews's and Pearson's correlation. QA tasks are evaluated using Exact Match (EM) and F1. NLG tasks are evaluated using BLEU, NIST, METEOR (MET), ROUGE-L, CIDEr and TER. To assess the efficiency of our approaches, we report the number of trainable parameters in millions denoted as *# Params (M)*. We compare AdaKron and MAdaKron to full model Fine-Tuning applied to **BERT Base** and **Large** [1], **RoBERTa Base** and **Large** [24], **DeBERTa-v3-base** [34] and **v2-XXL** [35], **GPT-2 Medium** and **Large** [2] . Furthermore, we evaluate our approaches against several recently proposed PEFT methods: **Houlsby** [6] and **Pfeiffer Adapter** [11], **AdaMix** [5], **BitFit** [12], **LoRA** [13], **UNIPELT** [20], **Compacter(++)** [9], **(IA)^3** [19], **AdaLoRA** [15], and **Vera** [14]. When results from prior work are unavailable, we replicate the baseline experiments using the *Adapters* repository [4]. Replicated results are labelled as *repr.* in all tables to distinguish them from reported outcomes in the literature.

Implementation details. We implement the proposed AdaKron and MAdaKron approaches in Pytorch, using an A100 GPU. Our implementation is based on the publicly available *Hugging Face Transformers* codebase [36]. We follow the hyperparameter configuration in [5,15]. Hyperparameter settings for both our approaches and the reproduced baselines are reported in Appendix A in Tables A.1–A.4 for all the pre-trained language models and GLUE, SuperGLUE, SQuAD and NLG datasets. The number of experts in each MAdaKron layer is set to 4 for each NLU task and 8 for GPT-2-based models. We use AdaFactor [37], which does not require setting an initial learning rate, to optimise our models. For BERT- and RoBERTa-based models, the intermediate adapter dimension is set to 48 and 16, respectively, while for DeBERTa-based models and GPT-2-based models it is equal to 16 and 8, respectively. We choose these dimensions based on the ablation study reported in Section 5.1, where we study the impact of different intermediate dimensions. For all the models, the dimension of the down projection layer D_c is $r_2 = 4$. For each dataset, we run repeated experiments with four different seeds (0, 42, 1234, 4321) and report the mean performance for both our approaches and all the replicated baselines. For each results Table, we report in **bold** the best result, and underline the second best.

5. Results and discussions

In this section, we present the results of our comparative evaluation. First, we discuss the effectiveness and efficiency of AdaKron and MAdaKron in Fine-Tuning various Pre-trained Language Models on different downstream tasks, aiming to answer our research questions. Then, in Section 5.1, we perform an ablation study to assess the impact of the design choices of our proposal.

Natural Language Understanding. Tables 1 and 2 show the performance attained by our approaches, i.e. AdaKron and MAdaKron, and several recent PEFT models on the GLUE and SuperGLUE development dataset, respectively, using BERT-base and Large and RoBERTa-base and Large as PLMs. In both Tables 1 and 2, we report only the average performance across the GLUE and SuperGLUE datasets. Refer to Tables B.1-B.7 in Appendix B for detailed per-task results.

On the GLUE benchmark, MAdaKron consistently shows a superior balance between performance and parameter efficiency compared to both Full Fine-Tuning and existing PEFT methods. AdaKron already outperforms approaches that require substantially more parameters, such as UNIPELT, Compacter, (IA)^3, and Houlsby or Pfeiffer Adapters, showing that our efficient approach can match or exceed the effectiveness of heavier configurations. Conversely, compared to highly compact methods such as LoRA, BitFit, and Vera, AdaKron improves performance by at least one point on most models, with the only exception being RoBERTa-base, where results are comparable. Partial and Full MAdaKron further enhance AdaKron's results by achieving the highest performance on RoBERTa-base and BERT-Large while training the same number of parameters as AdaKron.

Finally, it is worth noting that, although MAdaKron scores slightly below AdaMix (−0.2 on BERT-base and −0.6 on RoBERTa-Large), it reduces by one-third the number of trained parameters, showing a better efficiency-performance trade-off.

Regarding SuperGLUE, we show only Partial MAdaKron on this dataset since it shows better performance than the Full approach in three out of four models on the GLUE benchmark. We first note that our direct competitors, i.e. AdaMix, score almost one point lower than MAdaKron on all the PLMs. Across all tested PLMs, MAdaKron attains either the best or second-best performance while training only one-third of the parameters required by the strongest baselines, which is the Houlsby Adapter in three out of four models. When not reaching the best performance, the gap remains below 0.3 points, highlighting its stable and

Table 1
Main results on the GLUE development set with BERT and RoBERTa models.

Model	# Params (M)	BERT-base	# Params (M)	RoBERTa-base	# Params (M)	BERT-Large	# Params (M)	RoBERTa-Large
Full Fine-Tuning [5,12]	110	82.7	125	85.3	336	84.1	355	88.9
UNIPELT ([5] + repr.)	1.4	83.5	1.4	84.9	3.2	84.7	3.2	89.3
Compacter (repr.)	1.3	79.7	1.3	83.0	2.3	82.9	2.3	75.2
Compacter++ (repr.)	1.3	79.9	1.3	84.8	2.2	82.6	2.2	88.6
$(IA)^3$ (repr.)	1.2	80.7	1.2	85.2	2.3	83.1	2.3	88.1
AdaMix Adapter ([5] + repr.)	0.9	**84.5**	0.3	86.0	2.4	84.6	0.8	**89.9**
Houlsby Adapter [5,38]	0.9	83.0	0.8	88.4	4.8	82.5	0.8	86.4
Pfeiffer Adapter (repr.)	0.9	82.7	0.3	85.9	2.4	84.6	0.8	87.9
LoRA [5,13]	0.3	82.2	0.3	**86.4**	0.8	82.7	0.2	88.6
BitFit [5,38]	0.1	82.3	0.1	84.6	0.3	84.1	0.3	88.5
Vera (repr.)	0.04	81.1	0.06	85.6	0.02	82.1	0.02	88.3
AdaKron ([8] + **new**)	0.6	83.6	0.2	85.7	1.6	84.2	0.6	89.1
Full MAdaKron	0.6	84.3	0.2	85.4	1.6	84.8	0.6	89.1
Partial MAdaKron	0.6	83.9	0.2	**86.4**	1.6	**85.1**	0.6	89.3

Table 2
Main results on the SuperGLUE development set with both BERT and RoBERTa.

Model	# Params (M)	BERT-base	# Params (M)	RoBERTa-base	# Params (M)	BERT-Large	# Params (M)	RoBERTa-Large
Full Fine-Tuning [39]	110	71.8	125	70.7	336	77.1	355	81.4
UNIPELT (repr.)	1.4	75.2	1.4	**78.7**	3.2	77.1	3.2	82.1
Compacter (repr.)	1.3	73.1	1.3	76.9	2.3	76.3	2.3	80.8
Compacter++ (repr.)	1.3	71.3	1.3	75.9	2.2	75.4	2.2	78.0
$(IA)^3$ (repr.)	1.2	74.0	1.2	78.6	2.3	76.8	2.3	81.6
Houlsby Adapter (repr.)	1.8	74.6	1.2	78.3	4.8	**77.4**	1.6	**82.6**
Pfeiffer Adapter (repr.)	0.9	72.8	0.3	76.8	2.4	76.1	0.8	81.7
AdaMix (repr.)	0.9	71.4	0.3	78.0	2.4	76.0	0.8	80.8
LoRA (repr.)	0.3	74.2	0.3	77.3	0.8	75.8	0.2	81.9
Vera (repr.)	0.04	72.7	0.06	74.7	0.02	73.5	0.02	80.2
AdaKron (**new**)	0.6	74.6	0.2	78.4	1.6	77.1	0.6	82.0
Partial MAdaKron	0.6	**75.8**	0.2	**78.7**	1.6	77.1	0.6	82.5

efficient behaviour across all the evaluated architectures. Furthermore, MAdaKron improves AdaKron's results by at least 0.3 points, except for BERT-Large, where our approaches achieve performance on par.

We further evaluate AdaKron and Partial MAdaKron on DeBERTa-v3-base and report the results in Table 3, which shows the performance of our approaches on GLUE and both SQuAD v1.1 and 2.0. AdaKron and MAdaKron consistently achieve strong results across benchmarks while training only a small fraction of model parameters. On GLUE and SQuAD v1.1, they achieve performance on par or even better than Full Fine-Tuning and most PEFT approaches, including the Houlsby Adapter, while requiring significantly fewer parameters. Regarding Question Answering datasets, both AdaKron and MAdaKron outperform Pfeiffer Adapters on SQuAD v1.1 and perform on par on SQuAD v2.0, again using one-third fewer parameters. They also outperform LoRA and the Houlsby Adapter across both SQuAD datasets, and achieve performance comparable to Full Fine-Tuning on most tasks. Regarding our direct competitors, MAdaKron outperforms AdaMix on SQuAD v2.0, while AdaMix requires one-third more parameters. It is worth noting that AdaLoRA leverages adaptive rank allocation aligned with DeBERTa's disentangled attention, prioritising high-rank updates in the most informative weights. This gives AdaLoRA a slight advantage on SQuAD v2.0, outperforming our approaches by approximately 0.3 points. As a final stress test for AdaKron, we scale up to 1.5B parameters using the DeBERTa-v2-XXL model, and evaluate it on GLUE datasets. Given the large cost of training DeBERTa-v2-XXL, we select a subsample of GLUE datasets, namely MRPC, CoLA, RTE and STS-B. We focus on these datasets since they include a diverse range of tasks, spanning both low-resource and high-resource scenarios, as well as covering Single-Sentence and Pairwise Text Classification. AdaKron achieves comparable performance to Full Fine-Tuning while training only 0.06 % of model parameters. AdaKron shows strong results across all four tasks: it reaches 90.4 on MRPC (vs. 92.0 with Full Fine-Tuning), 70.6 on CoLA (vs. 72.0), 88.9 on RTE (vs 89.9) and 91.9 (vs 92.9) on STS-B. While the average gap between AdaKron and Full Fine-Tuning is approximately three points, the performance on STS-B differs by just one point, despite AdaKron updating only around one million parameters.

In addition to the English benchmarks, we include a compact experiment on the development set of the recent SemEval-2025 "Bridging the Gap in Text-Based Emotion Detection" task [40], which covers a large set of high- and low-resource languages. From this pool, we select a representative subset consisting of four high-resource languages (Spanish, Hindi, Russian, Romanian) and three low-resource languages (Hausa, Nigerian Pidgin, Marathi), in order to keep the experiment lightweight and comparable to our main setup while still spanning both ends of the resource spectrum. Our goal here is not to compete with leaderboard systems, but to stress-test whether the advantages of AdaKron transfer beyond English under controlled conditions. For this reason, we restrict the comparison to PEFT baselines, i.e. LoRA and Pfeiffer Adapter, and we report results only on the official development set. Results of our evaluation are reported in Table 4. Across the selected languages, including the low-resource ones, AdaKron consistently outperforms both baselines, indicating that the benefits of our Kronecker-based parameterisation extend to multilingual, data-scarce settings, even though a full multilingual study remains outside the scope of this work.

NLG tasks. Table 5 shows the performance of our approaches and several baseline models, using GPT-2 Medium as PLM on three NLG tasks. AdaKron outperforms Full Fine-Tuning on the E2E dataset with respect to all the metrics used, showing its efficiency and efficacy in Fine-Tuning an LLM to a downstream NLG task. AdaKron achieves better performance than Houlsby Adapter and Fine-Tuning only the top two layers, but training only 0.3 M parameters. Regarding DART and WebNLG tasks, AdaKron achieves performance in the same ballpark as all baselines proposed. MAdaKron enhances AdaKron performance on both datasets, outperforming both Full Fine-Tuning and Fine-Tuning Top2. Furthermore, MAdaKron achieves performance in the same ballpark as AdaMix and LoRA while requiring fewer trainable parameters. Regarding GPT-2 Large (774M of parameters) applied to the E2E benchmark, AdaKron

Table 3
Main results on the GLUE and SQuAD development sets with DeBERTa-v3-base.

Model	# Params (M)	GLUE Avg.	SQuAD v1.1 EM	SQuAD v1.1 F1	SQuAD v2.0 EM	SQuAD v2.0 F1
Full Fine-Tuning [15]	184	87.0	86.0	92.7	85.4	88.4
Houlsby Adapter [15]	0.6	87.9	86.1	92.7	84.9	87.9
Pfeiffer Adapter [15]	0.6	88.0	86.2	92.8	84.9	87.8
AdaMix (repr.)	0.6	88.7	**87.7**	**93.6**	85.2	88.1
Houlsby Adapter [15]	0.3	88.1	85.3	92.1	84.3	87.3
Pfeiffer Adapter [15]	0.3	88.1	85.9	92.5	84.5	87.6
LoRA [15]	0.2	88.3	86.4	92.8	84.7	87.2
AdaLoRA [15]	0.2	**89.0**	87.2	93.4	**85.6**	**88.7**
AdaKron (**new**)	0.4	88.6	87.0	93.2	84.7	87.6
Partial MAdaKron	0.4	88.4	87.5	92.2	85.3	88.3

Table 4
Main results (Micro F_1) on the development set of SemEval 2025-"Bridging the Gap in Text-Based Emotion Detection" Task.

Model	# Params (M)	Spanish	Hindi	Russian	Romanian	Marathi	Hausa	Nigerian Pidgin
LoRA (repr.)	0.3	67.3	70.6	**82.6**	**66.8**	86.5	59.5	**55.8**
Pfeiffer Adapter (repr.)	0.9	69.8	70.1	81.1	62.9	84.9	**60.2**	52.0
AdaKron (**new**)	0.6	**72.1**	**72.4**	82.1	66.4	**87.2**	59.4	54.7

Table 5
Main results on NLG tasks test sets with GPT-2 Medium.

Model	# Params (M)	E2E BLEU	E2E NIST	E2E MET	E2E ROUGE-L	E2E CIDEr	DART BLEU	DART TER ↓	WebNLG BLEU	WebNLG TER ↓
Full Fine-Tuning [5]	355	68.2	8.62	46.2	71.0	2.47	46.2	**0.46**	46.5	0.53
Fine-Tuning Top2 [5]	25.0	68.1	8.59	46.0	70.8	2.41	41.0	0.56	36.0	0.72
Houlsby Adapter [5]	11.0	68.9	8.71	46.1	71.3	2.47	45.2	**0.46**	54.9	**0.39**
AdaMix Adapter [5]	0.4	69.8	8.75	**46.8**	**71.9**	2.52	**47.7**	0.47	54.9	**0.39**
LoRA [5]	0.3	**70.4**	**8.85**	**46.8**	71.8	**2.53**	47.1	**0.46**	**55.3**	**0.39**
DyLoRA [41]	0.4	69.2	8.75	46.3	70.8	2.46	46.5	0.48	54.5	**0.39**
AdaKron (**new**)	0.3	69.3	8.65	45.8	71.3	2.41	45.0	0.50	51.6	0.43
Partial MAdaKron	0.3	69.2	8.68	46.3	71.1	2.46	46.6	0.49	53.3	0.42

and MAdaKron achieve 68.8 and 69.5 in BLEU, respectively, outperforming Full Fine-Tuning (68.5 BLEU) while only training 0.6M parameters. Furthermore, MAdaKron outperforms Houlsby Adapter (69.1 BLEU) and is on par with LoRA (70.4 BLEU) while training fewer parameters (0.6M for MAdaKron vs 0.9M for Houlsby Adapter and 0.7M for LoRA).

5.1. Ablation study and analysis

In this section, we conduct an ablation study to assess our design choices. Furthermore, we analyse the training and inference efficiency of our methods.

Intermediate dimension. We perform an ablation study to assess the impact of different intermediate dimensions of AdaKron layers and the output dimensions of the two FFNs comprising the down projections. The intermediate size and the reduction factor r_2, defined in Section 3, play a pivotal role in controlling the parameter efficiency of AdaKron: smaller intermediate sizes result in a reduced parameter count, but they may potentially impact performance negatively, while a smaller reduction factor introduces more parameters. We explore a range of different intermediate sizes, i.e. 8, 16, 32, 48, 64 and 256, coupled with two reduction factors, i.e. 2 and 4. Table 6 reports the results of the different configurations using BERT-base on four GLUE datasets, i.e. SST2, CoLA, RTE and STS-B, which include a diverse range of tasks, spanning both high- and low-resource scenarios, as well as covering Single-Sentence and Pairwise Text Classification. We split the original training set, using 85 % for training and 15 % for evaluation. The intermediate dimension of 48 with a reduction factor of 4 achieves the best average result overall. We note that the best average performance overall has been achieved when the reduction factor r_2 is equal to 4, and thus the output vector of AdaKron is a 48-dimensional vector with blocks of length 12. This configuration maintains sufficient representational capacity while minimising information loss, which is the result of the projection to a small intermediate dimension. Conversely, with the same intermediate dimension $r = 48$, a reduction factor of $r_2 = 12$ yields 4-dimensional blocks that are too small to retain all input information, leading to performance degradation. Furthermore, a large intermediate dimension r typically leads to improved performance compared to a small one. For example, both $r = 64$ and $r = 256$ outperform all other dimensions when the reduction factor is equal to 2, but require more parameters to be trained. Consequently, we opt for the optimal configuration of AdaKron, featuring an intermediate dimension of 48 and a reduction factor of 4.

Furthermore, we evaluate different AdaKron's intermediate dimensions using DeBERTa-v3-base on the four new GLUE datasets and we expect the results to generalise well with other Natural Language Understanding tasks. We evaluate four different intermediate dimensions, i.e. 8, 16, 32 and 64 as suggested in [15]. Detailed results are reported in Appendix B in Table B.12. The intermediate dimensions of 16 achieve the best AdaKron performance on all the proposed datasets, while larger sizes, such as 32 and 64 are one point lower on specific datasets, such as SST2 and STS-B, compared to dimension 16. This finding contrasts with the results obtained with BERT as Language Model, where a dimension of 32 outperforms 16. The superior performance of a small dimension in DeBERTa-v3-base suggests that its architecture benefits compact representations, while a large dimension may introduce overfitting. Thus, we use intermediate dimensions of 16 while training DeBERTa-v3-base with our approaches.

Variations of Kronecker Product. In this section, we present an ablation study to analyse the impact of the Kronecker product within the AdaKron layer. While the original AdaKron configuration applies the Kronecker product only to the down projection layers, we also explore variants where it is applied to the up projections, or both the up and down projections. Furthermore, since the Kronecker product is non-

Table 6
Ablation study on the four new GLUE development datasets with BERT-base to study the impact of the intermediate dimension.

Intermediate Dim. $r = r_1 \cdot r_2$	8	16	32	48	64	256	8	16	32	48	64	256	32	48
Down projections Dim. r_1, r_2	2,4	2,8	2,16	2,24	2,32	2,128	4,2	4,4	4,8	4,12	4,16	4,64	8,4	12,4
# Params (M)	0.1	0.2	0.4	0.6	0.9	3.5	0.1	0.2	0.3	0.6	0.8	3.0	0.3	0.6
Average Performance	86.6	87.4	87.7	87.7	88.1	88.2	86.9	87.5	87.8	**88.3**	88.2	88.2	87.6	88.0

commutative, we introduce Symmetric AdaKron, a variation designed to explore the impact of input order. The Symmetric AdaKron Adapter applies the Kronecker product twice, switching the order of the input vectors, and then sum up the results. Furthermore, we added two more variants of Kronecker product based on the Kronecker sum between two vectors y_v and y_c as $y_v \oplus y_c = y_v \odot \mathbf{1}_{r_2} + y_c \odot \mathbf{1}_{r_1}$, where $\mathbf{1}_n = (1, 1, \dots, 1) \in \mathbb{R}^n$ is the all-ones vector. To evaluate the effect of using the Kronecker sum, we replace the Kronecker product with the sum and additionally consider a variant that combines the outputs of both the product and the sum. Beyond the Kronecker product, we evaluate alternative matrix products, namely the penetrating face product, which is a generalization of the Hadamard product to vectors with different dimensions, and the Khatri-Rao product. The penetrating face product between y_c and y_v (as defined in Section 3) is: $y_c[\odot]y_v = [(y_c)_1 \odot y_v | \dots | (y_c)_n \odot y_v] \in \mathbb{R}^{r_1}$, where $\odot$ is the Hadamard product. Compared to the Kronecker product, which produces a vector of $r_1 \cdot r_2$, this formulation results in a lower-dimensional output of length r_1. We also investigate the Khatri-Rao product, which is a generalization of the Kronecker product and it is defined as: $y_c \star y_v = [(y_c)_1 \otimes y_v | \dots | (y_c)_{r_1} \otimes y_v] \in \mathbb{R}^{r_1 \cdot r_2}$, where each subvector $(y_c)_i$ is an r_2-dimensional block of y_c, yielding an output vector with the same dimensionality as the Kronecker product. The results of these ablations are reported in Table 7, using the Pfeiffer Adapter as baseline. AdaKron, with the Kronecker product applied only in the down projections, achieves the best average performance, exceeding all other variants by at least 0.4 points and being the only configuration that outperforms the Pfeiffer Adapter baseline. Applying the Kronecker product in the up projections further reduces the number of trained parameters into an AdaKron Adapter, but downgrades the average performance by at least one point. Since the output of the up projection is summed with the input vector of an Adapter, these results suggest that training a unique layer can produce a vector in the same space as the input one, while applying the Kronecker product can generate a block representation in a different space. AdaKron outperforms the Symmetric and the Kronecker sum-based variants because the Kronecker product preserves the separable contributions of the input vectors, while summation in both approaches discards this structure. It also surpasses the Hadamard and Khatri-Rao variants, highlighting the superior representational capacity of the Kronecker composition: compared to Hadamard, it produces higher-dimensional outputs that enable richer representations, while unlike Khatri-Rao, it avoids partitioning the input into subvectors, thus retaining global contextual information.

Table 7
Ablation study on the effect of the Kronecker Product.

Model	Avg.
Pfeiffer Adapter (repr.)	88.1
AdaKron	**88.3**
Kronecker in Up proj.	87.2
Kronecker in Down and Up proj.	85.2
Symmetric AdaKron	87.8
Kronecker Sum	87.7
AdaKron + Kronecker Sum	87.6
Hadamard product	87.2
Khatri-Rao product	87.9

Variations of Mixture of Experts. We conduct an ablation study to evaluate the contribution of each component within the Mixture of Experts (MoE) framework employed in MAdaKron. First, we analyse the expert utilisation frequency. Since experts are selected uniformly at random, each expert is actually chosen in approximately 25 % of the iterations during the training on all the datasets used for the ablation studies. Importantly, no expert is activated more frequently than the others, which ensures that, when averaging the expert outputs, all experts contribute equally and none is undertrained. Furthermore, to isolate the effects of individual components, we modify the architecture in four ways: first, we remove the consistency regularisation loss during training to evaluate its impact on performance; second, we replace the uniform probability of the random gate with a biased probability where the first expert receives the input with a probability of 0.7 and the three remaining experts with a probability of 0.1; third, we replace expert merging during inference with random routing of each input to a single expert; and fourth, we substitute the random gating mechanism with a trainable softmax gate, which requires an additional load-balancing loss, is trained jointly with the MAdaKron layers, and selects the top-2 experts during inference. Table 8 summarizes the results of our ablation study. Across all evaluated variants, we fix the number of experts to four and set the intermediate dimensionality of the Adapter layers to 48. Among all configurations, MAdaKron, which incorporates the consistency regularization loss during training and expert merging during inference, achieves the best overall performance. Removing the consistency loss leads to an average performance drop of 0.6 points, highlighting the important role of this regularization objective: each expert minimizes its own prediction error via the cross-entropy loss while simultaneously learning to match the predictions of another expert, effectively treating it as a teacher and ensuring consistent outputs across experts without restricting their independent optimization. Furthermore, employing a biased expert selection probability in place of a uniform one yields performance comparable to that of MAdaKron. When the biased selection is applied without the Consistency loss, performance decreases by approximately one point. These results emphasise the critical role of the Consistency loss: even under non-uniform selection probabilities, it ensures that all experts learn to produce similar output probability distributions. Disabling the merging strategy in favour of random selection at inference results in a performance decrease of at least 0.3 points, showing that averaging experts yields more robust and accurate results than relying on a single one [5]. Replacing random gating with a learned softmax gate consistently decreases performance by at least 0.5 points, despite increasing the model size by approximately 0.1 million parameters. This reduction is due to load-balancing issues and expert collapse in softmax-based gating [21], which prevent the effective utilisation of all experts. Furthermore, we note that the learnable gate achieves performance better than the random one in STS-B, which is interestingly a low-resource dataset, compared to SST2, CoLA and RTE. Finally, the selection frequencies of the learnable gate of the four experts during the training on SST2, CoLA, RTE, STS-B are 24.4 %, 28.3 %, 23.5 %, and 23.8 % for the first, second, third, and fourth experts, respectively. Although the second expert is chosen slightly more often than the others, the balancing loss ensures that the activation probabilities of all experts remain balanced, preventing reliance on a single expert or the repeated activation of the same expert pair. Our findings thus provide further evidence that, in the context of MAdaKron, stochastic expert selection offers both efficiency and effectiveness advantages over learned gating mechanisms.

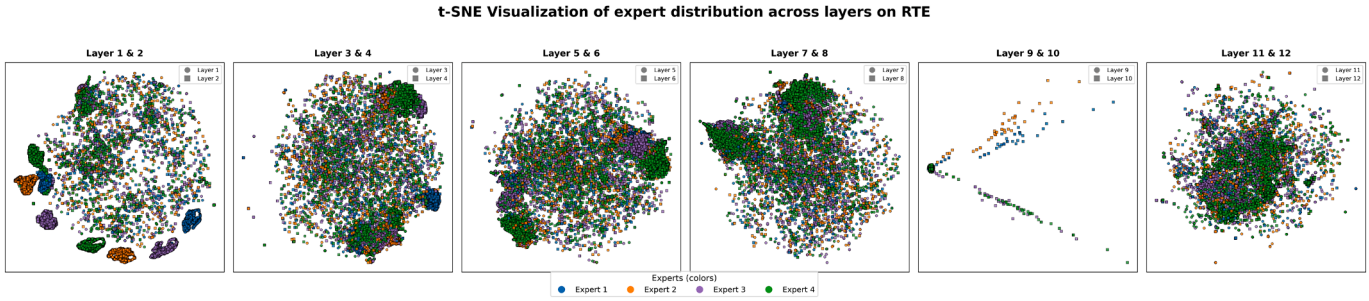


Fig. 3. t-SNE projection of the 48-dimensional output vectors produced by MAdaKron for a single batch from the RTE dataset. Visualisations are reported sequentially for each pair of consecutive layers to illustrate the evolution of expert representations throughout the model.

Table 8
Ablation study on different MoE architectures.

Model	Avg.
Pfeiffer Adapter (repr.)	88.1
MAdaKron	**88.5**
w/o Consistency	87.9
w/o Merging, with Consistency	88.2
w/o Merging, w/o consistency	87.7
Biased Prob., with Consistency	88.4
Biased Prob., w/o Consistency	87.4
Softmax Gating, with balancing	88.0
Softmax Gating, w/o balancing	87.7

Table 9
Ablation study on the number of experts.

Model	Num. of Experts	Avg.
Pfeiffer Adapter (repr.)	0	88.1
Partial MAdaKron	4	**88.5**
Partial MAdaKron	2	87.5
Partial MAdaKron	6	88.1
Partial MAdaKron	8	87.3
Full MAdaKron	2	87.7
Full MAdaKron	4	87.9
Full MAdaKron	6	84.2
Full MAdaKron	8	81.4

Number of Experts. We perform an ablation study to assess the impact of different numbers of experts in MAdaKron layers with BERT-base. We evaluate different numbers of experts, i.e. 2, 4, 6 and 8, on four GLUE tasks, and we report the results of our analysis in Table 9. Overall, MAdaKron with 4 experts achieves the best result, regardless of the Partial or Full approach. Consequently, we opt for the optimal configuration, featuring 4 as the number of experts. As the number of experts increases, leading to greater sparsity, we observe that low-resource tasks like RTE, and STS-B, which have a small training set (i.e. less than $7k$ data), degrade in performance compared to a small number of experts. For example, the performance on RTE and STS-B with 2 experts outperforms the one with 8 experts by 1.8 and 0.8, respectively. In contrast, high-resource tasks like SST2, which has $67k$ training data, show performance on par with both a high and small number of experts. A similar trend is also shown with GPT2-Large as PLMs, where increasing the number of experts enhances performance on a high-resource benchmark such as E2E, which has $42k$ training data: MAdaKron achieves 73.8 in BLEU with 8 experts and 72.8 with 4 experts. This finding suggests the benefits of an increased model capacity through additional experts in tasks with large training sets, while low-resource tasks may struggle due to the large number of parameters Fig. 3.

Visualisation of MAdaKron Experts. In this section, we examine the evolution of MAdaKron output representations when fine-tuning BERT-base on the RTE dataset, to understand how expert representations evolve across layers and where discriminative patterns emerge. To this end, we apply t-SNE to project the 48-dimensional output vectors of MAdaKron of a single batch of dimension 4 and sequence length of 128 into a two-dimensional space for visualisation. We limit the batch dimension to 4 for a better interpretation of the final plot.

In the earliest layers (1-2), the outputs of the four experts form clearly separated regions, indicating that each expert captures distinct aspects of the input. In layers 3-4, while some separation remains, the expert representations begin to overlap, suggesting a gradual convergence. These early stages appear to encode primarily lexical and semantic information, before task-specific features become prominent. The middle layers (5-8) mark a transition phase. Layers 5-6 and 7-8 exhibit similar behaviour, where the representations become increasingly intermixed and start shifting from semantic toward task-specific encoding. This stage corresponds to the emergence of discriminative structure in the representation space, where features relevant to the RTE classification begin to dominate. In layers 9-10, a form of degenerate routing behaviour is observed, with the outputs predominantly concentrated around a single expert, except for a few distinct points in the tenth layer. Finally, in layers 11-12, the outputs become almost fully blended across experts, showing a complete integration of features before the final prediction. Overall, this layer-wise analysis reveals a coherent progression: the early layers encode general semantic information, the middle layers develop task-specific discriminative representations, and the final layers consolidate these features.

Efficiency Analysis. As discussed by Lialin et al. [42], parameter efficiency is a multidimensional concept encompassing memory utilisation, training speed, inference overhead and energy consumption. While the previous section analysed the effectiveness of MAdaKron on downstream tasks, this section focuses on its computational efficiency. To this end, we conduct experiments using a BERT-base model fine-tuned on the RTE dataset for a single training epoch, using a batch size of 32 and a sequence length of 512 tokens. All experiments are executed on an NVIDIA A100 GPU. We report peak GPU memory usage, total training time (in seconds), TFLOPs, MACs, estimated energy consumption, and corresponding CO_2 emissions for the single training epoch. In detail, TFLOPs are TeraFLOPs, i.e. 10^{12} FLOPs, where a FLOP is defined as the number of Floating-Point arithmetic operations the system performs [43], while MACs (Multiply-Accumulate Operation) count multiply-accumulate operations (compute) [44]. Table 10 presents the results of our analysis. Both AdaKron and MAdaKron reduce the number of trainable parameters by one-third compared to Pfeiffer Adapter and AdaMix, enhancing parameter efficiency. The Kronecker product further lowers computational cost: training BERT-base on RTE for one epoch requires 3.11 and 6.22 TFLOPs for AdaKron and MAdaKron, respectively, compared to 3.12 for Pfeiffer Adapter and 6.24 for AdaMix. AdaKron achieves lower TFLOP requirements than the original Pfeiffer Adapter, while MAdaKron is more efficient than its direct competitor AdaMix, showing that integrating the Kronecker product within the adapter layer decreases both operational complexity and computational cost. AdaKron

Table 10
Training and Inference Efficiency. The experiments are conducted on BERT-base, and we set both training and inference batch sizes equal to 32 and sequence length equal to 512. Training and Inference Times are measured in seconds (s), Memory Usage in GB, Energy Consumption in Joules and the Carbon Emissions in milligrams.

Model	# Params (M)	Training Phase						Inference Stage
		TFLOPs	MACs	Time (s)	Memory Usage (GB)	Energy (J)	CO_2 (mg)	Inference Time (s)
LoRA	0.3	3.09	1.54	33.2	15.0	48.28	5.36	1.31
Pfeiffer Adapter	0.9	3.12	1.56	35.0	19.9	48.67	5.40	1.36
AdaMix	0.9	6.24	3.12	64.8	39.4	97.34	10.82	1.38
AdaKron	0.6	3.11	1.55	36.5	19.8	48.52	5.39	1.37
Partial MAdaKron	0.6	6.22	3.11	65.2	39.3	97.03	10.78	1.39

achieves training time and memory usage of 36.5 s and 19.8 GB, close to Pfeiffer Adapter's 35.0 s and 19.9 GB, while MAdaKron requires 65.2 s and 39.3 GB, comparable to AdaMix (64.8 s and 39.4 GB). Estimated energy consumption, assuming 15.6 pJ per FP32 FLOP and 400 g CO_2/kWh (as reported in vendor documentation for NVIDIA A100 GPUs), is 48.5 Joule (J) and 5.39 mg CO_2 for AdaKron, 48.67 J and 5.40 mg CO_2 for Pfeiffer Adapter, and double for MAdaKron and AdaMix due to the consistency regularisation, though MAdaKron emits slightly less CO_2 than AdaMix (difference of 0.1 mg). Despite achieving the best performance in the majority of the datasets and models, both AdaMix and MAdaKron process each batch twice, and thus both require twice TFLOPs, MACs, Energy and CO_2 consumption compared to AdaKron, and this may negatively affect the environmental sustainability of training.

During inference, all Adapter-based methods exhibit similar latency (1.36-1.39 s), slightly higher than LoRA (1.31 s). This discrepancy arises because LoRA does not introduce additional layers into the Transformers, whereas Adapter-based methods insert new layers, increasing the inference cost.

6. Conclusions

In this paper, we present MAdaKron, an Adapter-based Fine-Tuning technique that leverages the Kronecker product and incorporates a Mixture of Experts system. Our approach aims to significantly reduce the number of trainable parameters while maintaining competitive performance. Our extensive evaluation, guided by the research questions presented earlier in Section 4, shows both the effectiveness and the efficiency of both methods. Indeed, RQ1 focuses on the efficiency and performance of AdaKron, showing that it delivers competitive results across a wide range of tasks while requiring a fraction of the parameters compared to traditional Fine-Tuning and other PEFT methods. Building on this, RQ2 examines the impact of integrating a MoE system into AdaKron, leading to the development of MAdaKron. We integrate AdaKron with a Mixture of Experts architecture to further enhance performance while keeping parameter efficiency intact, showing the adaptability of our approach. Lastly, RQ3 investigates MAdaKron's effectiveness compared to other PEFT techniques. Our extensive evaluation on eighteen different datasets through eight PLMs shows that MAdaKron consistently outperforms existing methods across several NLP tasks, including Natural Language Inference and Generation and Question Answering. In summary, our findings highlight MAdaKron as a promising method for improving the efficiency of Fine-Tuning large models without compromising their performance, offering practical solutions for resource-constrained environments.

Limitations and Future Works. While the AdaKron and MAdaKron frameworks offer strong performance, they remain computationally demanding due to the need to Fine-Tune large-scale language models. Similarly, the proposed MAdaKron method incurs higher training costs compared to conventional PEFT techniques, such as LoRA and Pfeiffer Adapter. Based on empirical observations, MAdaKron generally requires approximately twice the number of training iterations relative to standard PEFT methods, due to the inclusion of consistency regularization. This added computational burden may negatively affect the environmental sustainability of training, particularly in terms of carbon emissions. We evaluate our approaches on models containing up to 1.5 billion parameters, which corresponds to the largest publicly available encoder-only architectures. Due to computational constraints, we do not extend our evaluation to decoder-only language models exceeding 3 billion parameters. Compared to LoRA-based methods, which do not modify the model architecture, adapter-based approaches, such as AdaKron and MAdaKron, are less scalable since they introduce additional trainable layers into the Transformer backbone, whose definition and training are unfeasible for closed-source models or decoder-only models with 3B parameters or more, such as Llama-2-3B and -7B [45]. While this architectural modification results in a slightly higher number of trainable parameters, it enables greater expressivity and flexibility. As shown in Section 5, our methods consistently outperform LoRA-based baselines across all evaluated Natural Language Understanding datasets and models, and they are on par on Natural Language Generation tasks, highlighting the effectiveness of our design. Despite these costs, AdaKron is orthogonal to existing PEFT approaches and can be flexibly integrated into a wide range of Fine-Tuning strategies to potentially enhance their performance. We plan to extend MAdaKron to open-source multimodal models to assess its effectiveness in multi-task and cross-modal adaptation scenarios, as well as in multilingual models.

CRediT authorship contribution statement

Marco Braga: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Alessandro Raganato:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Resources, Methodology, Investigation, Conceptualization; **Gabriella Pasi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization.

Data availability

Data will be made available on request.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of generative AI and AI-assisted technologies in the writing process.

During the preparation of this work the authors used GPT-4 in order to: Grammar and spelling check, Paraphrase and reword. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support.

Appendix A. Evaluation settings

Table A.1
Hyperparameter configurations for BERT-based and RoBERTa-based models used to evaluate both AdaKron and MAdaKron on GLUE datasets.

Dataset	Epochs	Batch Size	Warmup ratio	Weight Decay	Adapter Size
BERT-base and Large					
CoLA	100	16	0.06	0.1	48
SST2	40	64	0.06	0.1	48
MNLI	40	64	0.06	0.1	48
RTE	80	64	0.06	0.1	48
QQP	60	64	0.06	0.1	48
MRPC	100	16	0.06	0.1	48
QNLI	20	64	0.06	0.1	48
STS-B	80	32	0.06	0.1	48
RoBERTa-base and Large					
CoLA	80	64	0.6	0.1	16
SST2	20	64	0.6	0.1	16
MNLI	20	64	0.6	0.1	16
RTE	60	64	0.6	0.1	16
QQP	80	64	0.6	0.1	16
MRPC	60	64	0.6	0.1	16
QNLI	20	64	0.6	0.1	16
STS-B	80	64	0.6	0.1	16

In this Section, we report the detailed datasets description, implementation details and the hyperparameters used to Fine-Tune the Pretrained Language Models with both AdaKron and MAdaKron.

A.1. Datasets

We evaluate AdaKron and MAdaKron on the General Language Understanding Evaluation (GLUE) benchmark [22], which includes eight types of Natural Language Understanding tasks including Linguistic Acceptability (CoLA [46]), Sentiment Analysis (SST2 [47]), Similarity and Paraphrase tasks (MRPC [48], STS-B [49], QQP [22]), and Natural Language Inference (MNLI [50], QNLI [51], RTE [52,53]). Following prior studies [1,6], we do not include the WNLI dataset [54] as there are issues with the construction of this dataset such that BERT-like models do not outperform the majority baseline.[2] Additionally, we assess our approaches on the SuperGLUE benchmark [28] covering five Natural Language Understanding tasks: Question Answering (BoolQ [55]), Natural Language Inference (RTE), Word Sense Disambiguation (WiC [56]) and Commonsense Reasoning (WSC [54], CB [57]). Following standard practice [4,5,12,13], we used the development set in GLUE and SuperGLUE as test data since the original test set is hidden. We further evaluate our approaches on two Question-Answering (QA) datasets (SQuAD v1.1 [29] and SQuAD v2.0 [30]), and conduct experiments on three Natural Language Generation tasks: E2E [31], which requires generating texts in the restaurant domain, WebNLG [32] and DART [33], which both requires mapping sets of RDF triples to text.

[2] See (12) in gluebenchmark.com/faq.

Table A.2
Hyperparameter configurations for DeBERTa-based models used to evaluate both AdaKron and MAdaKron on GLUE datasets and SQuAD v1.1 and v2.0.

Dataset	Epochs	Batch Size	Warmup steps	Weight Decay	Adapter Size
DeBERTa-v3-base					
CoLA	50	32	800	0.5	16
SST2	25	32	6000	0.1	16
MNLI	20	32	8000	0.1	16
RTE	50	32	600	0.3	16
QQP	50	32	8000	0.1	16
MRPC	40	32	600	0.1	16
QNLI	20	32	2000	0.1	16
STS-B	50	32	800	0.1	16
SQuAD v1.1	10	16	1000	0.1	16
SQuAD v2.0	12	64	1000	0.1	16
DeBERTa-v2-XXL					
CoLA	10	32	800	0.0	8
RTE	11	32	600	0.01	8
MRPC	30	32	600	0.01	8
STS-B	10	32	800	0.1	8

Table A.3
Hyperparameter configurations for BERT-based, RoBERTa-based models used to evaluate both AdaKron and MAdaKron on SuperGLUE datasets.

Dataset	Epochs	Batch Size	Warmup ratio	Weight Decay	Adapter Size
BERT-base and Large					
BoolQ	100	16	0.06	0.1	48
RTE	40	64	0.06	0.1	48
WiC	40	64	0.06	0.1	48
WSC	80	64	0.06	0.1	48
CB	60	64	0.06	0.1	48
RoBERTa-base and Large					
BoolQ	100	16	0.06	0.1	48
RTE	40	64	0.06	0.1	48
WiC	40	64	0.06	0.1	48
WSC	80	64	0.06	0.1	48
CB	60	64	0.06	0.1	48

Table A.4
Hyperparameter configurations for GPT-2-based models used to evaluate both AdaKron and MAdaKron on NLG tasks.

Dataset	Epochs	Batch Size	Warmup Steps	Adapter Size
GPT-2-Medium and Large				
E2E	20	8	2000	8
WebNLG	25	8	2500	8
DART	20	8	2000	8

A.2. Models and baselines

We compare AdaKron to full model Fine-Tuning and several PEFT methods applied to **BERT Base** [1] (12-layer, 768-hidden, 12-heads, 110M parameters)[3] and **Large** [1] (24-layer, 1024-hidden, 16-heads, 355M parameters),[4] **RoBERTa Base** [24] (12-layer, 768-hidden, 12-heads, 110M parameters)[5] and **Large** [24] (24-layer, 1024-hidden, 16-heads, 355M parameters),[6] **DeBERTa-v3-base** [34] (12-layer, 768-hidden, 12-heads, 184M parameters)[7] and **DeBERTa-v2-XXL** [35] (48-layer, 1536-hidden, 24-heads, 1.5B parameters),[8] **GPT-2 Medium** [2]

[3] bert-base-uncased
[4] bert-large-uncased
[5] roberta-base
[6] roberta-large
[7] deberta-v3-base
[8] deberta-v2-xxl

(24-layer, 1024-hidden, 16-heads, 355M parameters)[9] and **Large** [2] (36-layer, 1280-hidden, 20-heads, 774M parameters).[10] When results from prior work are unavailable, we replicate the baseline experiments using the Adapters repository [4]. Replicated results are labelled as *repr.* in all tables to clearly distinguish them from reported outcomes in the literature.

A.3. Implementation details

We implement the proposed AdaKron and MAdaKron approaches in Pytorch, using an A100 GPU for our experiments. Our implementation is based on the publicly available *Hugging Face Transformers* codebase [36]. Following the hyperparameter configuration in [5] and [15], the number of experts in each MAdaKron layer is set to 4 for each NLU task and 8 for GPT-2 based models. We use AdaFactor [37], which does not require setting an initial learning rate, to optimise our models. Intermediate Adapter dimensions are chosen from $r \in \{8, 16, 32, 48\}$, and the dimension of the down projection layer D_c is $r_2 = 4$. Hyperparameter settings are reported in Tables A.1–A.4 for all the pre-trained language models and GLUE, SuperGLUE, SQuAD and NLG datasets, respectively. For all the reproduced baselines, we use the same hyperparameters as AdaKron and MAdaKron to compare the different methods in the same setting. Furthermore, for LoRA and VeRA we put the low-rank dimension $r = 8$, as suggested by the original work [13]. Regarding Houlsby and Pfeiffer, we use the same dimensions as our approach. For Compacter and Compacter++, we use the best one in the original work [9], i.e. 24. Regarding the number of trained parameters, when results are taken directly from other works, we report the parameter counts stated in those works; otherwise, we provide the values corresponding to our own reproduced configuration.

Appendix B. Extended results

In this section, we report the detailed results for each task and model with standard deviation for all our approaches. Furthermore, we apply a statistically significant test, a one-sided test, with p-value $p < 0.05$ (which corresponds to the 95 % interval), to understand if the mean of the four different runs of our approach is significantly greater than the baselines. In each Table, we report in **bold** and underlined the best and second best results, respectively, for each column. * and † represent significant improvements of our approaches over the best and second best baselines, respectively. *Avg.* is the average performance across the selected datasets Tables B.2–B.6, B.8–B.10, and B.11.

Regarding BERT-base on the GLUE benchmark, our proposed approaches yield significant improvements on SST2, CoLA, and RTE, highlighting their ability to enhance both sentiment classification and entailment recognition. For BERT-large, the most significant gain is on QNLI, indicating that MAdaKron effectively leverages its adaptive expert mechanism to improve performance in natural language inference tasks. On RoBERTa-base, significant improvements are achieved on RTE and STS-B, while RoBERTa-large shows the largest gains on SST2, showing consistent improvements across model scales and task types.

Overall, while the average performance across models remains broadly comparable to other PEFT methods, especially on SQuAD and NLG tasks, our approaches exhibit superior efficiency and effectiveness.

These results confirm that MAdaKron achieves performance on par with or even better than the proposed baseline while requiring fewer trainable parameters.

Tables B.13–B.15 present a comprehensive ablation analyses conducted to assess the contribution of individual architectural components within AdaKron and MAdaKron. Specifically, these experiments aim to understand the effect of the Kronecker product and its alternative formulations, as well as the role of the Mixture of Experts mechanism and its variants.

In Tables B.13 and B.14, we compare seven AdaKron variants and seven MAdaKron variants, respectively, on four GLUE datasets, i.e. SST2, CoLA, RTE and STS-B.

Finally, Table B.15 investigates the effect of the number of experts in MAdaKron.

Table B.1
Main results on the GLUE development set with BERT-base.

Model	# Params (M)	MNLI	QNLI	SST2	QQP	MRPC	CoLA	RTE	STS-B	Avg.
Baselines										
Full Fine-Tuning [5]	110	83.2	90.0	91.6	87.4	90.9	62.1	66.4	89.8	82.7
UNIPELT [5]	1.4	83.9	90.5	91.5	85.5	90.2	58.6	73.7	88.9	83.5
Compacter (repr.)	1.3	81.6	89.2	87.9	85.1	88.7	58.7	59.8	86.3	79.7
Compacter++ (repr.)	1.3	80.0	89.8	91.6	83.7	89.5	57.8	60.5	86.2	79.9
$(IA)^3$ (repr.)	1.2	79.4	88.8	92.3	82.7	90.1	59.9	65.4	86.8	80.7
AdaMix Adapter [5]	0.9	**84.7**	**91.5**	92.4	**87.6**	**92.4**	**62.9**	74.7	**89.9**	**84.5**
Houlsby Adapter [5]	0.9	83.1	90.6	91.9	86.8	89.9	61.5	71.8	88.6	83.0
Pfeiffer Adapter	0.9	83.3	91.1	92.0	87.5	90.7	60.3	67.6	89.6	82.7
LoRA [5]	0.3	82.5	89.9	91.5	86.0	90.0	60.5	71.5	85.7	82.2
BitFit [5]	0.1	81.4	90.2	92.1	84.0	90.4	58.8	72.3	89.2	82.3
Vera (repr.)	0.04	83.1	90.5	92.3	85.9	89.9	59.0	61.0	86.8	81.1
Hadamard-Adapter [58]	0.03	80.4	89.7	92.4	85.9	90.2	58.4	71.9	88.5	82.2
Our approaches										
AdaKron [8]	0.6	$83.5_{0.18}$	$91.1_{0.45}$	$92.0_{0.36}$	$87.1_{0.14}$	$90.8_{1.24}$	$61.1_{0.35}$	$73.8_{1.05}$	$89.4_{0.24}$	83.6
Full MAdaKron	0.6	$83.9_{0.20}$	$91.3_{0.47}$	$\mathbf{92.8}^{*,\dagger}_{0.13}$	$87.4_{0.08}$	$91.5_{0.59}$	$62.3^{\dagger}_{0.35}$	$\mathbf{76.0}^{*,\dagger}_{0.44}$	$89.2_{0.13}$	84.3
Partial MAdaKron	0.6	$83.9_{0.05}$	$91.1_{0.52}$	$92.3_{0.08}$	$\mathbf{87.6}^{\dagger}_{0.03}$	$91.1_{0.41}$	$61.8_{0.90}$	$74.2_{0.75}$	$89.4_{0.42}$	83.9

[9] gpt-2-medium
[10] gpt-2-large

Table B.2
Main results on the GLUE development set with BERT-Large.

Model	# Params (M)	MNLI	QNLI	SST2	QQP	MRPC	CoLA	RTE	STS-B	Avg.
Baselines										
Full Fine-Tuning [12]	336	85.5	91.7	93.4	87.5	90.7	62.2	71.9	90.0	84.1
Houlsby Adapter (repr)	4.8	85.1	92.0	93.0	**88.7**	83.1	58.3	70.4	89.7	82.5
Pfeiffer Adapter (repr.)	2.4	85.2	91.7	93.5	87.9	90.4	64.3	74.2	89.7	84.6
UNIPELT (repr.)	3.2	**85.7**	91.6	**94.1**	85.7	91.1	**66.0**	75.2	88.8	84.7
Compacter (repr.)	2.3	84.3	91.4	93.8	85.5	91.1	60.6	68.7	88.0	82.9
$(IA)^3$ (repr.)	2.3	80.7	90.0	93.4	81.7	90.8	64.3	75.0	88.5	83.1
Compacter++ (repr.)	2.2	82.5	91.1	93.7	83.4	90.3	62.9	68.7	88.1	82.6
AdaMix (repr.)	2.4	**85.7**	92.3	94.0	87.8	91.6	59.6	75.4	90.1	84.6
LoRA (repr.)	0.8	85.3	91.5	93.8	85.6	91.2	59.5	66.7	90.0	82.7
BitFit [12]	0.3	84.4	91.4	93.2	85.4	**91.7**	63.6	73.2	**90.3**	84.1
Vera (repr.)	0.02	83.9	91.6	93.8	84.5	91.1	58.6	66.0	87.5	82.1
Our approaches										
AdaKron	1.6	$84.8_{0.22}$	$91.7_{0.17}$	$93.3_{0.29}$	$87.5_{0.15}$	$90.1_{0.64}$	$62.9_{0.48}$	$73.4_{0.96}$	$90.0_{0.14}$	84.2
Full MAdaKron	1.6	$84.7_{0.14}$	$\mathbf{92.8}^{*,\dagger}_{0.17}$	$93.6_{0.35}$	$86.4_{0.26}$	$90.9_{0.66}$	$64.8_{1.92}$	$75.3_{2.93}$	$89.7_{0.30}$	84.8
Partial MAdaKron	1.6	$85.4_{0.19}$	$92.4^{\dagger}_{0.06}$	$93.7_{0.33}$	$88.0_{0.13}$	$91.4_{0.30}$	$64.1_{0.79}$	$\mathbf{75.6}_{0.34}$	$90.2_{0.16}$	**85.1**

Table B.3
Main results on the GLUE development set with RoBERTa-base.

Model	# Params (M)	MNLI	QNLI	SST2	QQP	MRPC	CoLA	RTE	STS-B	Avg.
Baselines										
Full Fine-Tuning [12]	125	86.4	92.3	94.2	88.0	92.5	61.1	77.4	90.6	85.3
UNIPELT [59]	1.4	87.2	92.0	93.9	87.7	91.7	61.9	74.3	90.3	84.9
Compacter [4]	1.3	86.1	92.4	94.1	86.7	90.4	55.5	68.6	90.0	83.0
Compacter++ (repr)	1.3	85.2	92.0	94.1	84.6	91.9	64.1	76.2	90.5	84.8
$(IA)^3$ [4]	1.2	86.2	91.9	94.4	86.4	92.3	63.0	76.9	90.6	85.2
AdaMix Adapter (repr.)	0.8	83.2	93.3	94.9	86.7	92.3	64.5	82.3	90.8	86.0
Pfeiffer Adapter (repr.)	0.6	87.1	92.5	94.3	87.9	92.6	63.0	79.0	90.8	85.9
LoRA [4]	0.3	**87.6**	**93.1**	**95.0**	88.5	92.6	64.0	80.3	90.7	**86.4**
DyLoRA [41]	0.3	87.0	93.0	94.3	**90.0**	91.7	60.5	76.9	91.0	85.5
AdaLoRA [60]	0.3	87.0	93.0	94.0	89.9	89.4	58.8	75.9	90.6	85.0
AutoLoRA [60]	0.3	87.0	92.9	94.9	90.3	89.4	61.3	77.0	90.8	85.5
Bit-Fit [12]	0.1	84.8	91.3	93.7	84.5	92.0	61.8	77.8	90.8	84.6
RoAd [61]	0.09	86.3	91.9	93.9	89.6	89.2	64.4	78.9	90.5	85.6
Vera (repr+ [14])	0.04	87.1	91.8	94.6	87.0	89.5	**65.6**	78.7	90.7	85.6
LoReFT [62]	0.02	83.1	91.2	93.4	87.4	89.2	60.4	79.0	90.0	84.2
DoRA [63]	0.6	85.3	91.9	93.4	87.9	87.8	56.5	74.8	88.5	83.3
MELoRA [63]	0.6	85.0	91.5	93.0	87.5	88.7	54.4	75.5	87.3	82.9
HydraLoRA [63]	0.7	85.5	91.9	93.2	87.6	89.5	57.0	73.7	88.5	83.4
TopLoRA [63]	0.9	85.7	92.1	93.4	88.4	88.7	60.3	78.3	88.9	84.5
BoRA [64]	0.3	85.7	91.8	93.1	87.9	88.2	57.4	76.9	89.3	83.8
Our approaches										
AdaKron	0.2	$86.7_{0.14}$	$92.4_{0.29}$	$94.0_{0.17}$	$87.5_{0.05}$	$92.6_{0.49}$	$62.2_{0.91}$	$79.4_{1.74}$	$90.9_{0.12}$	85.7
Full MAdaKron	0.2	$86.4_{0.17}$	$92.6_{0.10}$	$94.8_{0.21}$	$87.4_{0.14}$	$93.1_{0.40}$	$62.6_{0.73}$	$76.5_{0.37}$	$90.0_{0.07}$	85.4
Partial MAdaKron	0.2	$86.6_{0.06}$	$92.8_{0.37}$	$94.7_{0.19}$	$87.5_{0.18}$	$\mathbf{93.3}_{0.45}$	$64.6_{0.84}$	$\mathbf{80.5}^{\dagger}_{0.45}$	$\mathbf{91.1}^{\dagger}_{0.10}$	**86.4**

Table B.4
Main results on the GLUE development set with RoBERTa-Large.

Model	# Params (M)	MNLI	QNLI	SST2	QQP	MRPC	CoLA	RTE	STS-B	Avg.
Baselines										
Full Fine-Tuning [5]	355	90.2	94.7	96.4	92.2	90.9	68.0	86.6	**92.4**	88.9
UNIPELT (repr.)	3.2	90.3	94.0	96.3	88.9	93.2	**72.1**	**89.8**	90.2	89.3
$(IA)^3$ (repr.)	2.3	87.8	92.7	95.2	87.8	92.9	70.1	86.3	92.0	88.1
Compacter (repr.)	2.3	67.6	80.6	82.1	78.8	85.3	68.3	60.4	78.6	75.2
Compacter++ (repr.)	2.2	89.5	94.6	96.0	88.7	93.1	68.3	86.7	91.9	88.6
Houlsby Adapter [5]	0.8	90.3	94.7	96.3	91.5	87.7	66.3	72.9	91.5	86.4
Pfeiffer Adapter	0.8	90.5	94.8	96.6	**91.7**	89.7	67.8	80.1	91.9	87.9
AdaMix [5]	0.8	**90.9**	**95.4**	**97.1**	89.8	**94.1**	70.2	89.2	**92.4**	**89.9**
LoRA [5]	0.8	90.6	94.8	96.2	91.6	90.2	68.2	85.2	92.3	88.6
MoKA [17]	0.2	90.6	94.9	96.3	91.1	91.2	69.3	87.4	92.2	89.1
BitFit [38]	0.1	90.0	94.5	96.1	86.4	93.4	68.1	87.3	91.9	88.5
Vera (repr+ [14])	0.02	90.0	94.4	96.1	89.7	90.9	68.0	85.9	91.7	88.3
DoRA [63]	1.6	89.3	93.5	95.7	88.3	88.7	61.7	80.1	90.9	86.0
MELoRA [63]	1.6	88.9	93.2	95.9	87.9	87.8	60.6	79.5	90.2	85.5
HydraLoRA [63]	1.7	89.3	93.4	95.8	88.3	89.5	61.1	79.4	90.6	85.9
TopLoRA [63]	2.4	89.9	94.2	95.6	88.9	90.4	64.6	85.2	91.5	87.6
BoRA [64]	0.9	89.8	93.8	95.3	88.6	89.0	60.6	83.8	91.3	86.5
Our approaches										
AdaKron	0.6	$90.2_{0.19}$	$94.8_{0.25}$	$96.9^{*}_{0.10}$	$91.0_{0.33}$	$90.9_{0.74}$	$69.2_{2.26}$	$87.4_{1.13}$	$92.1_{0.08}$	89.1
Full MAdaKron	0.6	$90.4_{0.14}$	$95.0_{0.26}$	$96.5_{0.11}$	$88.8_{0.18}$	$92.0_{1.48}$	$68.0_{0.96}$	$87.7_{0.17}$	$90.8_{0.20}$	88.6
Partial MAdaKron	0.6	$90.6_{0.17}$	$95.0_{0.21}$	$96.6_{0.06}$	$88.5_{0.23}$	$94.0_{0.41}$	$69.0_{0.63}$	$88.9_{0.23}$	$92.0_{0.20}$	89.3

Table B.5
Main results on the GLUE development set with DeBERTa-v3-base.

Model	# Params (M)	MNLI	QNLI	SST2	QQP	MRPC	CoLA	RTE	STS-B	Avg.
Baselines										
Full Fine-Tuning [15]	184	90.1	94.0	95.6	**89.8**	89.5	69.2	83.7	91.6	87.0
UNIPELT (APL) [59]	0.9	89.1	93.7	95.6	89.6	92.4	68.0	81.6	**91.7**	87.7
Compacter (repr.)	0.7	89.4	93.9	95.7	86.6	92.5	69.3	85.6	91.2	88.1
Compacter++ (repr.)	0.6	89.7	94.1	95.7	87.3	93.1	69.4	85.9	91.2	88.1
Houlsby Adapter [15]	0.6	90.2	93.8	95.3	88.9	89.2	67.9	85.5	91.3	88.1
Pfeiffer Adapter [15]	0.6	90.3	94.0	95.5	88.9	89.2	69.5	84.1	91.5	88.2
AdaMix (repr.)	0.6	90.1	94.3	95.7	88.6	91.1	**71.0**	87.4	**91.7**	88.7
$(IA)^3$ (repr.)	0.6	87.7	92.2	95.2	86.2	**93.4**	69.6	86.0	91.3	87.7
LoRA [15]	0.3	90.4	94.0	94.9	88.9	89.7	68.7	85.6	**91.7**	88.3
AdaLoRA [15]	0.3	**90.7**	94.5	**95.8**	89.2	90.4	70.0	87.4	91.6	**89.0**
Pfeiffer Adapter [15]	0.3	90.1	93.9	94.7	88.6	89.7	69.1	84.5	91.4	88.1
Vera [65]	0.2	90.2	93.5	95.1	87.6	90.9	70.7	87.3	91.1	88.3
BitFit [15]	0.1	89.9	92.2	94.8	84.9	87.7	66.9	78.7	91.3	86.3
Our approaches										
AdaKron_8	0.2	$89.9_{0.12}$	$94.0_{0.21}$	$95.7_{0.20}$	$88.4_{0.15}$	$89.9_{1.51}$	$69.8_{1.08}$	$88.1_{0.52}$	$\mathbf{91.7}_{0.29}$	88.5
AdaKron_{16}	0.4	$89.9_{0.16}$	$93.9_{0.12}$	$95.6_{0.06}$	$88.2_{0.10}$	$90.4_{0.71}$	$70.4_{1.84}$	$\mathbf{88.6}^{*,\dagger}_{0.41}$	$91.6_{0.11}$	88.6
Partial MAdaKron_{16}	0.4	$89.8_{0.13}$	$\mathbf{94.3}_{0.18}$	$95.3_{0.11}$	$87.2_{0.12}$	$90.2_{0.93}$	$70.6_{0.96}$	$88.0_{0.43}$	$\mathbf{91.7}_{0.19}$	88.4

Table B.6
Main results on the SuperGLUE development set with both BERT-base and Large.

	Bert-base							Bert-Large						
Model	# Params (M)	BoolQ	RTE	WiC	WSC	CB	Avg.	# Params (M)	BoolQ	RTE	WiC	WSC	CB	Avg.
Baselines														
Full Fine-Tuning	110	72.9	68.4	71.1	63.5	83.0	71.8	336	77.7	70.4	**74.9**	**68.3**	94.6	77.1
Houlsby Adapter (repr.)	1.8	**76.7**	71.0	**71.8**	66.1	87.5	74.6	4.8	79.2	75.9	73.6	65.9	92.4	**77.4**
Compacter (repr.)	1.3	74.3	68.1	69.4	66.3	87.5	73.1	2.3	76.7	74.1	69.8	65.6	**95.5**	76.3
Compacter++ (repr.)	1.3	73.3	66.5	69.1	65.6	82.2	71.3	2.2	76.6	74.4	69.6	65.1	91.1	75.4
UNIPELT (repr.)	1.4	75.7	74.7	71.6	65.6	88.4	75.2	3.2	77.6	**78.2**	72.0	67.2	91.1	77.1
$(IA)^3$ (repr.)	1.2	74.5	73.0	70.6	65.4	86.6	74.0	2.3	77.2	77.8	71.1	67.0	91.1	76.8
$\text{Pfeiffer Adapter}_{48}$ (repr.)	0.9	75.1	**77.4**	71.0	64.7	78.6	72.8	2.4	77.5	77.3	71.7	66.6	87.5	76.1
AdaMix (repr.)	0.9	75.2	65.0	71.1	65.4	80.3	71.4	1.8	77.4	75.2	71.6	65.4	90.4	76.0
$\text{Pfeiffer Adapter}_{32}$ (repr.)	0.6	74.6	72.6	70.4	65.2	79.5	72.4	1.6	75.8	76.9	71.7	66.6	87.5	76.1
LoRA (repr.)	0.3	75.4	67.9	71.4	66.1	90.2	74.2	0.8	78.0	74.3	69.6	65.6	91.6	75.8
Vera (repr.)	0.04	73.7	67.0	71.1	66.1	85.7	72.7	0.02	70.2	71.8	68.8	65.4	91.1	73.5
Our approaches														
AdaKron	0.6	$73.7_{0.50}$	$72.8_{1.97}$	$70.5_{0.28}$	$65.7_{0.51}$	$90.2_{1.27}$	74.6	1.6	$76.4_{0.71}$	$76.5_{0.46}$	$72.9_{0.70}$	$67.0_{0.51}$	$92.5_{0.95}$	77.1
Partial MAdaKron	0.6	$74.5_{0.91}$	$74.4_{0.86}$	$71.5_{1.27}$	$\mathbf{67.7}^{*,\dagger}_{0.51}$	$\mathbf{90.8}_{1.20}$	**75.8**	1.6	$77.2_{0.82}$	$74.4_{0.36}$	$73.6_{0.42}$	$67.0_{0.71}$	$93.3_{0.61}$	77.1

Table B.7
Main results on the SuperGLUE development sets with both RoBERTa-base and Large.

	RoBERTa-base							RoBERTa-Large						
Model	# Params (M)	BoolQ	RTE	WiC	WSC	CB	Avg.	# Params (M)	BoolQ	RTE	WiC	WSC	CB	Avg.
Baselines														
Full Fine-Tuning [39]	125	83.5	52.7	69.3	58.7	89.3	70.7	355	**86.9**	86.6	**75.6**	63.5	94.6	77.2
Compacter (repr.)	1.3	79.2	77.4	68.9	64.7	94.2	76.9	2.3	79.2	77.4	68.9	64.7	94.2	76.9
Compacter + + (repr.)	1.3	79.7	77.7	67.2	64.9	90.2	75.9	2.2	85.5	86.8	70.9	64.4	82.2	78.0
UNIPELT (repr.)	1.4	81.6	81.5	69.3	65.1	96.4	**78.7**	3.2	85.8	89.0	71.6	64.7	99.5	82.1
$(IA)^3$ (repr.)	1.2	80.2	79.8	**69.5**	**67.0**	96.4	78.6	2.3	85.3	87.6	71.3	65.9	97.7	81.6
Pfeiffer Adapter (repr.)	0.6	80.8	82.6	61.6	64.7	92.0	76.3	0.8	86.5	**89.4**	70.4	65.6	98.2	82.0
LoRA (repr.)	0.3	80.7	78.2	67.2	64.5	96.0	77.3	0.2	80.7	78.2	67.2	64.5	96.0	77.3
Houlsby Adapter (repr.)	0.2	81.8	80.4	70.6	64.9	93.8	78.3	0.8	81.8	80.7	70.6	64.9	93.8	78.3
AdaMix (repr.)	0.3	**81.2**	82.3	66.9	63.5	96.0	78.0	0.8	85.7	88.1	73.4	64.4	92.3	80.8
Pfeiffer Adapter_{16} (repr.)	0.3	80.2	81.0	64.4	64.4	92.9	76.8	0.8	86.3	89.0	72.1	65.1	96.0	81.7
Vera (repr.)	0.06	77.2	73.9	65.5	64.2	92.4	74.7	0.02	85.4	85.6	71.5	65.3	93.3	80.2
Our approaches														
AdaKron	0.2	$79.7_{0.32}$	$81.1_{0.77}$	$69.4_{0.81}$	$65.4^{*}_{0.03}$	$96.4_{1.75}$	78.4	0.6	$86.0_{0.17}$	$87.8_{0.61}$	$71.1_{0.36}$	$64.9_{0.70}$	$\mathbf{100}^{*,\dagger}_{0.10}$	82.0
Partial MAdaKron	0.2	$80.5_{0.28}$	$\mathbf{82.7}_{0.35}$	$68.8_{0.77}$	$64.4_{0.14}$	$\mathbf{97.3}_{1.03}$	**78.7**	0.6	$86.5_{0.17}$	$89.0_{0.40}$	$72.2_{0.87}$	$64.9_{0.70}$	$\mathbf{100}^{*,\dagger}_{0.11}$	**82.5**

Table B.8
Main results on the SQuAD development sets with DeBERTa-v3-base.

Model	# Params (M)	SQuAD v1.1 EM	SQuAD v1.1 F1	SQuAD v2.0 EM	SQuAD v2.0 F1
Baselines					
Full Fine-Tuning [15]	184	86.0	92.7	85.4	88.4
Houlsby Adapter [15]	0.6	86.1	92.7	84.9	87.9
Pfeiffer Adapter [15]	0.6	86.2	92.8	84.9	87.8
AdaMix (repr.)	0.6	**87.7**	**93.6**	85.2	88.1
Houlsby Adapter [15]	0.3	85.3	92.1	84.3	87.3
Pfeiffer Adapter [15]	0.3	85.9	92.5	84.5	87.6
LoRA [15]	0.2	86.4	92.8	84.7	87.2
AdaLoRA [15]	0.2	87.2	**93.4**	**85.6**	**88.7**
Our approaches					
AdaKron_{16}	0.4	$87.0_{0.17}$	$93.2_{0.17}$	$84.7_{0.14}$	$87.6_{0.28}$
AdaKron_{8}	0.2	$87.2_{0.20}$	$93.2_{0.10}$	$84.3_{0.26}$	$87.3_{0.26}$
AdaKron_{32}	0.40	$87.0_{0.09}$	$93.2_{0.05}$	$84.1_{0.20}$	$87.2_{0.20}$
AdaKron_{64}	0.77	$87.1_{0.14}$	$93.1_{0.01}$	$82.8_{0.05}$	$85.8_{0.05}$
Partial MAdaKron_{8}	0.2	$87.5^{*}_{0.08}$	$92.2_{0.07}$	$85.3_{0.11}$	$88.3_{0.13}$

Table B.9
Main results on E2E test set with GPT-2-medium.

Model	# Params (M)	BLEU	NIST	MET	ROUGE-L	CIDEr
Baselines						
Full Fine-Tuning [5]	355	68.2	8.62	46.2	71.0	2.47
Fine-Tuning Top2 [5]	25.0	68.1	8.59	46.0	70.8	2.41
Houlsby Adapter [5]	11.0	68.9	8.71	46.1	71.3	2.47
AdaMix Adapter [5]	0.4	69.8	8.75	46.8	**71.9**	2.52
DyLoRA [41]	0.4	69.2	8.75	46.3	70.8	2.46
Vera [14]	0.4	70.1	8.81	46.6	71.5	2.50
Prefix-Tuning [5]	0.3	69.7	8.81	46.1	71.4	2.49
LoRA [5]	0.3	**70.4**	**8.85**	46.8	71.8	**2.53**
FourierFT [66]	0.048	69.1	8.82	**47.0**	71.8	2.51
Our approaches						
AdaKron_{8}	0.3	$69.3_{0.91}$	$8.65_{0.14}$	$45.8_{0.78}$	$71.3_{0.70}$	$2.41_{0.08}$
Partial MAdaKron_{8}	0.3	$69.2_{0.07}$	$8.68_{0.04}$	$46.3_{0.05}$	$71.1_{0.22}$	$2.46_{0.01}$

Table B.10
Main results on WebNLG and DART test sets with GPT-2-medium. ↓ means that the metrics (TER) should be minimised.

Model	# Params (M)	DART BLEU	DART TER ↓	WebNLG BLEU	WebNLG TER ↓
Baselines					
Full Fine-Tuning [5]	355	46.2	**0.46**	46.5	0.53
Fine-Tuning Top2 [5]	25	41.0	0.56	36.0	0.72
Houlsby Adapter [5]	11	45.2	**0.46**	54.9	**0.39**
AdaMix Adapter [5]	0.4	**47.7**	0.47	54.9	**0.39**
DyLoRA [41]	0.4	46.5	0.48	54.5	**0.39**
Prefix-Tuning [5]	0.3	46.4	**0.46**	55.1	0.40
LoRA [5]	0.3	47.1	**0.46**	**55.3**	**0.39**
Our approaches					
AdaKron$_8$	0.3	$45.0_{0.37}$	$0.50_{0.01}$	$51.6_{0.42}$	$0.43_{0.01}$
Partial MAdaKron$_8$	0.3	$46.6_{0.22}$	$0.49_{0.01}$	$53.3_{0.54}$	$0.42_{0.01}$

Table B.11
Main results on E2E test set with GPT-2-Large.

Model	# Params (M)	BLEU	NIST	MET	ROUGE-L	CIDEr
Baselines						
Full Fine-Tuning [13]	774	68.5	8.78	46.0	69.9	2.45
Houlsby Adapter [13]	0.9	69.1	8.68	46.3	71.4	2.49
Prefix-Tuning [13]	0.7	70.3	8.85	46.2	71.7	2.47
LoRA [13]	0.7	**70.4**	8.89	46.8	**72.0**	2.47
Vera [14]	0.2	70.3	8.85	46.9	71.6	**2.54**
FourierFT [66]	0.072	70.2	**8.92**	**47.0**	71.8	2.50
Our approaches						
AdaKron$_8$	0.6	$68.8_{0.46}$	$8.70_{0.01}$	$46.4_{0.37}$	$71.5_{0.34}$	$2.45_{0.01}$
Partial MAdaKron$_8$	0.6	$69.5_{0.55}$	$8.76_{0.06}$	$46.6_{0.15}$	$71.4_{0.28}$	$2.49_{0.01}$

Table B.12
Ablation study on the four new GLUE development datasets with DeBERTa-v3-base as Pretrained Language Model to find the best intermediate dimension for both AdaKron and MAdaKron.

Dim. r	# Params (M)	SST2 Acc	CoLA Mcc	RTE Acc	STS-B Pearson	Avg.
8	0.1	95.0	**85.2**	**93.4**	94.5	**92.0**
16	0.2	**96.5**	83.7	92.2	**94.8**	91.8
32	0.4	95.7	85.0	92.1	92.7	91.3
64	0.8	95.8	83.3	89.7	91.1	90.0

Table B.13
Ablation study on the four new GLUE development datasets with BERT-base to study the impact of the Kronecker product applied in the up projection layer and different products between the Down projection vectors.

Model	# Params (M)	SST2 Acc	CoLA Mcc	RTE Acc	STS-B Pearson	Avg.
Pfeiffer Adapter (repr.)	0.9	96.8	78.7	84.1	**92.8**	88.1
AdaKron	0.6	**96.9**	78.8	84.7	**92.8**	**88.3**
Kronecker product in Up projection	0.5	96.7	77.6	**84.9**	89.7	87.2
Kronecker product in both Down and Up projection	0.3	96.4	**81.6**	72.3	90.5	85.2
Symmetric AdaKron	0.6	96.6	77.5	84.8	92.4	87.8
Kronecker Sum	0.6	96.0	78.5	83.7	92.5	87.7
AdaKron + Kronecker Sum	0.6	95.2	78.0	84.7	92.6	87.6
Hadamard product	0.2	96.5	76.8	83.7	91.9	87.2
Khatri-Rao product	0.6	96.7	78.1	84.7	92.4	87.9

Table B.14
Ablation study on the four new GLUE development datasets with BERT-base to study the impact of the MAdaKron architecture, the merging mechanism during inference, the application of the Consistency loss the definition of a learnable gating.

MAdaKron	# Params (M)	SST2 Acc	CoLA Mcc	RTE Acc	STS-B Pearson	Avg.
Pfeiffer Adapter (repr.)	0.9	96.9	78.7	84.1	**92.9**	88.1
MAdaKron	0.6	**97.0**	79.1	85.6	92.2	**88.5**
w/o Consistency	0.6	96.5	78.7	84.3	92.1	87.9
w/o Merging, with Consistency	0.6	96.6	78.7	85.2	92.3	88.2
w/o Merging, w/o consistency	0.6	96.1	78.0	84.5	92.1	87.7
biased probability, with Consistency	0.6	96.4	78.8	**86.3**	92.3	88.4
biased probability, w/o consistency	0.6	95.3	**79.6**	82.5	92.3	87.4
Softmax Gating, with balancing	0.7	96.6	78.2	84.7	92.8	88.0
Softmax Gating, w/o balancing	0.7	96.7	78.0	83.4	92.7	87.7

Table B.15
Ablation study on the four new GLUE development sets with BERT-base as Pretrained Language Model to evaluate the impact of the number of experts on MAdaKron.

MAdaKron Model	# Experts	SST2	CoLA	RTE	STS-B	Avg.
Full	2	96.9	77.4	85.4	91.2	87.7
Full	6	95.9	73.3	80.6	87.0	84.2
Full	8	95.2	67.4	83.0	80.0	81.4
Partial	2	96.9	76.0	84.8	**92.3**	87.5
Partial	6	**97.0**	**79.2**	84.7	91.6	88.1
Partial	8	**97.0**	77.9	83.0	91.5	87.3
Full	4	96.7	78.7	83.7	**92.3**	87.9
Partial	4	**97.0**	79.1	**85.6**	92.2	**88.5**

References

[1] J. Devlin, M. Chang, K. Lee, K. Toutanova, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, 33, Curran Associates, Inc., 2020, pp. 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] C. Poth, H. Sterz, I. Paul, S. Purkayastha, L. Engländer, T. Imhof, I. Vulić, S. Ruder, I. Gurevych, J. Pfeiffer, Adapters: a unified library for parameter-efficient and modular transfer learning, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Singapore, 2023, pp. 149–160. https://doi.org/10.18653/v1/2023.emnlp-demo.13

[5] Y. Wang, S. Agarwal, S. Mukherjee, X. Liu, J. Gao, A.H. Awadallah, J. Gao, AdaMix: mixture-of-adaptations for parameter-efficient model tuning, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 5744–5760. https://doi.org/10.18653/v1/2022.emnlp-main.388

[6] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 2790–2799. https://proceedings.mlr.press/v97/houlsby19a.html.

[7] H. Zhou, X. Wan, I. Vulić, A. Korhonen, AutoPEFT: automatic configuration search for parameter-efficient fine-tuning, Trans. Assoc. Comput. Ling. (2024) 525–542. https://doi.org/10.1162/tacl_a_00662

[8] M. Braga, A. Raganato, G. Pasi, AdaKron: an adapter-based parameter efficient model tuning with Kronecker product, in: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 350–357. https://aclanthology.org/2024.lrec-main.32/.

[9] R. Karimi Mahabadi, J. Henderson, S. Ruder, Compacter: efficient low-rank hypercomplex adapter layers, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, 34, Curran Associates, Inc., 2021, pp. 1022–1035. https://proceedings.neurips.cc/paper_files/paper/2021/file/081be9fdff07f3bc808f935906ef70c0-Paper.pdf.

[10] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: the sparsely-gated mixture-of-experts layer, in: Int. Conf. Learn. Representations, 2017. https://openreview.net/forum?id=B1ckMDqlg.

[11] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, I. Gurevych, 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, Association for Computational Linguistics, Online, 2021, pp. 487–503. https://doi.org/10.18653/v1/2021.eacl-main.39

[12] E. Ben Zaken, Y. Goldberg, S. Ravfogel, BitFit: simple parameter-efficient fine-tuning for transformer-based masked language-models, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 1–9. https://doi.org/10.18653/v1/2022.acl-short.1

[13] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: low-rank adaptation of large language models, in: International Conference on Learning Representations, 2022. https://openreview.net/forum?id=nZeVKeeFYf9.

[14] D.J. Kopiczko, T. Blankevoort, Y.M. Asano, VeRA: vector-based random matrix adaptation, in: The Twelfth International Conference on Learning Representations, 2024. https://openreview.net/forum?id=NjNfLdxr3A.

[15] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, T. Zhao, Adaptive budget allocation for parameter-efficient fine-tuning, in: The Eleventh International Conference on Learning Representations, 2023. https://openreview.net/forum?id=lq62uWRJjiY.

[16] A. Edalati, M. Tahaei, I. Kobyzev, V.P. Nia, J.J. Clark, M. Rezagholizadeh, Krona: parameter efficient tuning with kronecker adapter, in: The Third Workshop on Efficient Natural Language and Speech Processing (ENLSP-III), 2022. https://neurips2023-enlsp.github.io/papers/paper_61.pdf.

[17] B. Yu, Z. Yang, X. Yi, MoKA:parameter efficiency fine-tuning via mixture of Kronecker product adaption, in: O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B.D. Eugenio, S. Schockaert (Eds.), Proceedings of the 31st International Conference on Computational Linguistics, Association for Computational Linguistics, Abu Dhabi, UAE, 2025, pp. 10172–10182. https://aclanthology.org/2025.coling-main.679/.

[18] M. Sadeghi, M.G. Nejad, M.J. Asl, Y. Gu, Y. Yu, M. Asgharian, V.P. Nia, MoKA: mixture of Kronecker adapters, arXiv preprint arXiv:2508.03527 (2025).

[19] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, C. Raffel, Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., https://dl.acm.org/doi/10.5555/3600270.3600412.

[20] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, S. Yih, M. Khabsa, UniPELT: a unified framework for parameter-efficient language model tuning, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 6253–6264. https://doi.org/10.18653/v1/2022.acl-long.433

[21] S. Zuo, X. Liu, J. Jiao, Y.J. Kim, H. Hassan, R. Zhang, J. Gao, T. Zhao, Taming sparsely activated transformer with stochastic experts, in: International Conference on Learning Representations, 2022. https://openreview.net/forum?id=B72HXs80q4.

[22] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. Bowman, GLUE: a multi-task benchmark and analysis platform for natural language understanding [dataset], in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 353–355. https://doi.org/10.18653/v1/W18-5446

[23] S. Kullback, R.A. Leibler, On information and sufficiency, The Ann. Math. Stat. 22 (1) (1951) 79–86. https://api.semanticscholar.org/CorpusID:120349231.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: a robustly optimized BERT pretraining approach, ArXiv abs/1907.11692 (2019). https://api.semanticscholar.org/CorpusID:198953378.

[25] X.L. Li, P. Liang, Prefix-tuning: optimizing continuous prompts for generation, 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), Association for Computational Linguistics, Online, 2021, pp. 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

[26] M. Tahaei, E. Charlaix, V. Nia, A. Ghodsi, M. Rezagholizadeh, KroneckerBERT: significant compression of pre-trained language models through Kronecker decomposition and knowledge distillation, in: M. Carpuat, M.-C. de Marneffe, I.V. Meza Ruiz (Eds.), Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Seattle, United States, 2022, pp. 2116–2127. https://doi.org/10.18653/v1/2022.naacl-main.154

[27] A. Edalati, M. Tahaei, A. Rashid, V. Nia, J. Clark, M. Rezagholizadeh, Kronecker decomposition for GPT compression, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 219–226. https://doi.org/10.18653/v1/2022.acl-short.24

[28] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, S.R. Bowman, SuperGLUE: a stickier benchmark for general-purpose language understanding systems [dataset], in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019. https://dl.acm.org/doi/10.5555/3454287.3454581.

[29] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, SQuAD: 100,000+ questions for machine comprehension of text [dataset], in: J. Su, K. Duh, X. Carreras (Eds.), Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 2383–2392. https://doi.org/10.18653/v1/D16-1264

[30] P. Rajpurkar, R. Jia, P. Liang, Know what you don't know: unanswerable questions for SQuAD [dataset], in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 784–789. https://doi.org/10.18653/v1/P18-2124

[31] J. Novikova, O. Dušek, V. Rieser, The E2E dataset: new challenges for end-to-end generation [dataset], in: K. Jokinen, M. Stede, D. DeVault, A. Louis (Eds.), Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Association for Computational Linguistics, Saarbrücken, Germany, 2017, pp. 201–206. https://doi.org/10.18653/v1/W17-5525

[32] C. Gardent, A. Shimorina, S. Narayan, L. Perez-Beltrachini, The webNLG challenge: generating text from RDF data [dataset], in: J.M. Alonso, A. Bugarín, E. Reiter (Eds.), Proceedings of the 10th International Conference on Natural Language Generation, Association for Computational Linguistics, Santiago de Compostela, Spain, 2017, pp. 124–133. https://doi.org/10.18653/v1/W17-3518

[33] L. Nan, D. Radev, R. Zhang, A. Rau, A. Sivaprasad, C. Hsieh, X. Tang, A. Vyas, N. Verma, P. Krishna, Y. Liu, N. Irwanto, J. Pan, F. Rahman, A. Zaidi, M. Mutuma, Y. Tarabar, A. Gupta, T. Yu, Y.C. Tan, X.V. Lin, C. Xiong, R. Socher, N.F. Rajani, DART: open-domain structured data record to text generation [dataset], in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 432–447. https://doi.org/10.18653/v1/2021.naacl-main.37

[34] P. He, J. Gao, W. Chen, DeBERTaV3: improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing, in: The Eleventh International Conference on Learning Representations, 2023. https://openreview.net/forum?id=sE7-XhLxHA.

[35] P. He, X. Liu, J. Gao, W. Chen, DEBERTa: decoding-enhanced BERT with disentangled attention, in: ICLR 2021, https://openreview.net/forum?id=XPZIaotutsD.

[36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: state-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[37] N. Shazeer, M. Stern, AdaFactor: adaptive learning rates with sublinear memory cost, in: Proceedings of the 35th International Conference on Machine Learning,

Proceedings of Machine Learning Research, PMLR, 2018. https://proceedings.mlr.press/v80/shazeer18a.html.
[38] L. Xu, H. Xie, S.-Z.J. Qin, X. Tao, F.L. Wang, Parameter-efficient fine-tuning methods for pretrained language models: a critical review and assessment, 2023, . https://arxiv.org/abs/2312.12148. 2312.12148
[39] S.-K. Hong, J.-S. Jang, H.-Y. Kwon, Enhancing performance of transformer-based models in natural language understanding through word importance embedding, Knowl. Based Syst. 304 (2024) 112404. https://doi.org/10.1016/j.knosys.2024.112404
[40] S.H. Muhammad, N. Ousidhoum, I. Abdulmumin, S.M. Yimam, J.P. Wahle, T. Ruas, M. Beloucif, C. De Kock, T.D. Belay, I.S. Ahmad, N. Surange, D. Teodorescu, D.I. Adelani, A.F. Aji, F. Ali, V. Araujo, A.A. Ayele, O. Ignat, A. Panchenko, Y. Zhou, S.M. Mohammad, SemEval-2025 task 11: bridging the gap in text-based emotion detection, 2025, (2025). https://arxiv.org/abs/2503.07269. 2503.07269
[41] M. Valipour, M. Rezagholizadeh, I. Kobyzev, A. Ghodsi, DyLoRA: parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation, in: A. Vlachos, I. Augenstein (Eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 3274–3287. https://doi.org/10.18653/v1/2023.eacl-main.239
[42] V. Lialin, V. Deshpande, X. Yao, A. Rumshisky, Scaling down to scale up: a guide to parameter-efficient fine-tuning, 2024, (2034). https://arxiv.org/abs/2303.15647. 2303.15647
[43] J. Chen, S.-h. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, S.H.G. Chan, Run, don't walk: chasing higher FLOPS for faster neural networks, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 12021–12031. https://doi.org/10.1109/CVPR52729.2023.01157
[44] Y. Wang, H. Tang, Y. Xie, X. Chen, S. Ma, Z. Sun, Q. Sun, L. Chen, H. Zhu, J. Wan, et al., An in-memory computing architecture based on two-dimensional semiconductors for multiply-accumulate operations, Nat. Commun. 12 (1) (2021) 3347. https://doi.org/10.1038/s41467-021-23719-3
[45] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288 (2023).
[46] A. Warstadt, A. Singh, S.R. Bowman, Neural network acceptability judgments [dataset], Trans. Assoc. Comput. Ling. 7 (2019) 625–641. https://doi.org/10.1162/tacl_a_00290
[47] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, S. Bethard (Eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1631–1642. https://aclanthology.org/D13-1170/.
[48] W.B. Dolan, C. Brockett, Automatically constructing a corpus of sentential paraphrases [dataset], in: Proceedings of the Third International Workshop on Paraphrasing (IWP2005), 2005. https://aclanthology.org/I05-5002.
[49] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, L. Specia, SemEval-2017 task 1: semantic textual similarity multilingual and crosslingual focused evaluation [dataset], in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1–14. https://doi.org/10.18653/v1/S17-2001
[50] A. Williams, N. Nangia, S. Bowman, A broad-coverage challenge corpus for sentence understanding through inference [dataset], in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 1112–1122. https://doi.org/10.18653/v1/N18-1101
[51] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, SQuAD: 100,000+ questions for machine comprehension of text [dataset], in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 2383–2392. https://doi.org/10.18653/v1/D16-1264
[52] D. Giampiccolo, B. Magnini, I. Dagan, B. Dolan, The third PASCAL recognizing textual entailment challenge, in: S. Sekine, K. Inui, I. Dagan, B. Dolan, D. Giampiccolo, B. Magnini (Eds.), Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, Association for Computational Linguistics, Prague, 2007, pp. 1–9. https://aclanthology.org/W07-1401/.
[53] L. Bentivogli, B. Magnini, I. Dagan, H.T. Dang, D. Giampiccolo, The fifth PASCAL recognizing textual entailment challenge [dataset], in: Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009, NIST, 2009. https://tac.nist.gov/publications/2009/additional.papers/RTE5_overview.proceedings.pdf.
[54] H.J. Levesque, E. Davis, L. Morgenstern, The winograd schema challenge [dataset], in: Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning, KR'12, AAAI Press, 2012, p. 552-561. https://cdn.aaai.org/ocs/4492/4492-21843-1-PB.pdf.
[55] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, K. Toutanova, BoolQ: exploring the surprising difficulty of natural yes-no questions [dataset], in: J. Burstein, C. Doran, T. Solorio (Eds.), 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 2924–2936. https://doi.org/10.18653/v1/N19-1300
[56] M.T. Pilehvar, J. Camacho-Collados, WiC: The word-in-context dataset for evaluating context-sensitive meaning representations [dataset], in: J. Burstein, C. Doran, T. Solorio (Eds.), 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 1267–1273. https://doi.org/10.18653/v1/N19-1128
[57] M.-C. de Marneffe, M. Simons, J. Tonhauser, The commitmentbank: investigating projection in naturally occurring discourse [dataset], Proc. Sinn Bedeutung 23 (2) (2019) 107-124. https://doi.org/10.18148/sub/2019.v23i2.601
[58] Y. Chen, Q. Fu, G. Fan, L. Du, J.-G. Lou, S. Han, D. Zhang, Z. Li, Y. Xiao, Hadamard adapter: an extreme parameter-efficient adapter tuning method for pre-trained language models, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM '23 Association for Computing Machinery New York, NY, USA (2023) 276–285. https://doi.org/10.1145/3583780.3614904.
[59] R. Li, G. Murray, G. Carenini, Mixture-of-linguistic-experts adapters for improving and interpreting pre-trained language models, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore,(2023) 9456–9469. https://doi.org/10.18653/v1/2023.findings-emnlp.634.
[60] R. Zhang, R. Qiang, S.A. Somayajula, P. Xie, AutoLoRA: automatically tuning matrix ranks in low-rank adaptation based on meta learning, in: K. Duh, H. Gomez, S. Bethard (Eds.), Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Mexico City, Mexico (2024) 5048–5060. https://aclanthology.org/2024.naacl-long.282/
[61] B. Liao, C. Monz, 3-in-1: 2D rotary adaptation for efficient finetuning, efficient batching and composability, Adv. Neural Inf. Process. Syst. 37 (2024) 35018–35048.
[62] Z. Wu, A. Arora, Z. Wang, A. Geiger, D. Jurafsky, C.D. Manning, C. Potts, ReFT: representation finetuning for language models, Adv. Neural Inf. Process. Syst. 37 (2024) 63908–63962.
[63] S. Li, X. Luo, H. Wang, X. Tang, Z. Cui, D. Liu, Y. Li, X. He, R. Li, Beyond higher rank: token-wise input-output projections for efficient low-rank adaptation, in: The Thirty-Ninth Annual Conference on Neural Information Processing Systems, (2025). https://openreview.net/forum?id=w2xl15ZbD3
[64] S. Li, X. Luo, H. Wang, X. Tang, Z. Cui, D. Liu, Y. Li, X. He, R. Li, BoRA: towards more expressive low-rank adaptation with block diversity, arXiv preprint arXiv:2508.06953 (2025).
[65] S. Azizi, S. Kundu, M. Pedram, LaMDA: large model fine-tuning via spectrally decomposed low-dimensional adaptation, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2024 Association for Computational Linguistics, Miami, Florida, USA (2024) 9635–9646. https://aclanthology.org/2024.findings-emnlp.563/
[66] Z. Gao, Q. Wang, A. Chen, Z. Liu, B. Wu, L. Chen, J. Li, Parameter-efficient fine-tuning with discrete Fourier transform, International Conference on Machine Learning, PMLR, (2024) 14884–14901.