DATALESS WEIGHT DISENTANGLEMENT IN TASK ARITHMETIC VIA KRONECKER-FACTORED APPROXIMATE CURVATURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Task Arithmetic (TA) provides a modular and scalable way to adapt foundation models. Combining multiple task vectors, however, can lead to cross-task interference, causing representation drift and degraded performance. Representation drift regularization provides a natural remedy to disentangle task vectors, but existing approaches typically require external task data, which conflicts with TA's modularity and availability constraints like privacy concerns. We propose a data-free approach by framing representation drift regularization as a curvature matrix approximation problem. This allows us leverage well-established techniques; in particular, we adopt Kronecker-Factored Approximate Curvature (KFAC) to obtain practical regularizers. Our method is data-free, has constant complexity with respect to the number of tasks, and improves performance on TA benchmarks.

1 INTRODUCTION

Task arithmetic (TA). TA (Ilharco et al., 2022) promises a modular and scalable approach for adapting foundation models. By fine-tuning, it produces task-specific parameter updates – so-called *task vectors* – which can be added or subtracted to edit model behavior. This enables reusing of task-specific knowledge across different domains without retraining. In practice, however, composing multiple task vectors often degrades performance due to cross-task interference: introducing a new task vector shifts representations relied on by other tasks. To prevent such interference, task-specific components must be decoupled, ensuring that other tasks' representations remain stable. This property, whereby distinct directions in parameter space lead to changes confined to non-overlapping regions of the input space, is known as *weight disentanglement* (Ortiz-Jimenez et al., 2023).

Encouraging weight disentanglement. To favor weight disentanglement, one might regularize the fine-tuning procedure to explicitly preserve other tasks' representations (Yoshida et al., 2025) or, in other words, prevent *representation drift* — i.e., change in a task's activations when new task vectors are added. Nonetheless, such regularizers often require access to other tasks' training data, which is impractical under privacy or regulatory constraints and contradicts modularity and reusability.

> *Therefore, our goal is to design a computationally efficient regularizer for weight disentanglement that can be used without requiring access to the training data.*

This task relates to approximating neural network function space distances (Dhawan et al., 2023), which measure how much a model's behavior changes without requiring access to the original data. Building on this perspective, we incorporate an additional insight specific to TA: fine-tuning the first-order Taylor approximation of the model around its pre-trained parameters empirically enhances weight disentanglement Ortiz-Jimenez et al. (2023). This linearization simplifies the representation drift into a quadratic form of the network Jacobian's *Gramian*, which can be pre-computed on, and shared instead of, the data. This regularizer enhances weight disentanglement (Fig. 1). However, the Gramian is intractably large, as its size grows quadratically with the number of parameters.

Link to curvature approximation. The Jacobian Gram matrix is an instance of the generalized Gauss-Newton (GGN) matrix (Schraudolph, 2003), an extensively studied object in the context of second-order optimization (Martens, 2010; 2020). This link allows us to leverage prior research on efficient curvature approximations. Specifically, we adopt Kronecker-factored approximate curvature (KFAC, Martens & Grosse, 2015), a block-diagonal approximation of the GGN where blocks correspond to layers and each block is a *Kronecker product* of two small matrices. KFAC drastically reduces storage and computation while still capturing most intra-layer correlations, bridging the gap between oversimplified diagonal approximations and the intractable full GGN of interest.

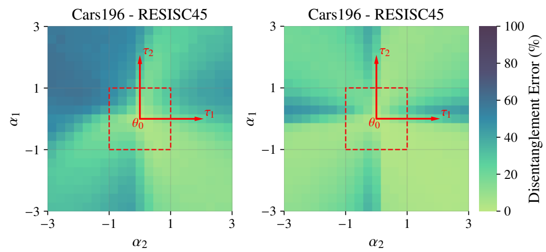


Figure 1: Weight disentanglement *(left)* without and *(right)* with Jacobian Gram regularization.

Adapting KFAC for TA. KFAC–based regularization faces a key limitation when applied to multi-task arithmetic: its associated regularizer cannot be accumulated exactly across tasks. The per-task regularizers induce memory and computational costs that grow linearly in the number of tasks. Going beyond the existing approximation, we propose an aggregation scheme that merges per-task curvature factors into a single surrogate, yielding *constant* complexity in the number of tasks during regularization. In summary, our contributions are the following:

- We derive a regularizer for task arithmetic that improves weight disentanglement without using external data, achieving state-of-the-art performance on vision and language benchmarks.
- We scale representation drift regularization by aggregating per-task regularizers into a single surrogate, ensuring *constant* complexity and storage regardless of the number of tasks.

2 Background: Task Arithmetic and Linearized Fine-Tuning

Setup. Let $f : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}^C$ denote a neural network that processes a datum $\boldsymbol{x} \in \mathbb{R}^D$ via parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ into a prediction $f(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^C$. During training, these predictions are compared to a target $\boldsymbol{y} \in \mathbb{R}^Y$ via a criterion function $c : \mathbb{R}^C \times \mathbb{R}^Y \to \mathbb{R}$ with the goal to minimize the empirical risk over a training data set $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_n$. We start from a model pre-trained on a large source dataset $\mathcal{D}_0$, yielding pre-trained weights $\boldsymbol{\theta}_0$. Our goal is to fine-tune this model on a specific downstream task t with data set $\mathcal{D}_t$, to obtain the task-specific fine-tuned weights $\boldsymbol{\theta}_t^\star$.

Task Arithmetic. The above fine-tuning procedure is typically repeated for multiple (T) tasks, yielding *task vectors* $\{\boldsymbol{\tau}_t := \boldsymbol{\theta}_t^\star - \boldsymbol{\theta}_0\}_{t=1}^T$. Such vectors form the core of TA, which posits that simple linear operations in weight space can induce targeted transformations in function space. This enables combining the capabilities of multiple task vectors to build a multi-task model without additional training, through simple linear combination (*task addition*): Given the individual task vectors $\{\boldsymbol{\tau}_t\}_{t=1}^T$, the composed model has parameters $\boldsymbol{\theta}_0 + \sum_{t=1}^T \alpha_t \boldsymbol{\tau}_t$ with $\alpha_t \in \mathbb{R}$ (in the simplest case, $\alpha_t = 1$). TA also addresses the removal of task-specific knowledge (*task negation*) by subtracting, rather than adding, a task vector. However, naïve linear composition is prone to interference, as overlapping task-vector updates often conflict and degrade the composed model's performance.

Linearized fine-tuning. Ortiz-Jimenez et al. (2023) empirically show that TA benefits from model linearization, particularly when applied during both training and inference. This approach replaces the network with its linear approximation around the pre-trained weights, $(f, \boldsymbol{\theta}_0) \leftrightarrow f_{\text{lin}}$ as

$$f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0), \tag{1}$$

with $\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \in \mathbb{R}^{C \times P}$ the Jacobian of the model's prediction on datum $\boldsymbol{x}$ w.r.t. its parameters, evaluated at $\boldsymbol{\theta}_0$. This encourages weight disentanglement in TA, a property whereby task vectors influence the model only on their own tasks, leaving its behavior unchanged elsewhere. Our goal is to construct a regularizer to encourage this property during linearized fine-tuning.

3 Making Representation Drift Regularization Data-Free

Simplified setup with two tasks. Model linearization simplifies the learning dynamics, allowing us to analyze how editing affects the model. We conduct this analysis in feature space through the lens

Algorithm 1 Idealized and practical representation drift regularizer for task t'

Require: Network $f(\cdot, \boldsymbol{\theta}_0)$, tasks $\{\mathcal{D}_t\}_{t=1, t\neq t'}^T$

Compute per-task GGNs $\{\boldsymbol{G}_{t\neq t'}\}$ (Eq. (3)) (approximate via KFAC, Sec. 3.3)

Merge over tasks: $\boldsymbol{G}_{-t'} = \sum_{t\neq t'} \lambda_t \boldsymbol{G}_t$ (optional: merge KFACs, Eq. (8))

return Quadratic form: $\boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\top \boldsymbol{G}_{-t'} \boldsymbol{\tau}$

Algorithm 2 Linearized FT on task vector $\boldsymbol{\tau}_{t'}$

Require: Initial weights $\boldsymbol{\theta}_0$, dataset $\mathcal{D}_{t'}$, task vector $\boldsymbol{\tau}_{t'}$ merged curvature matrix $\mathbf{G}_{-t'}$

Linearize the net: $(f, \boldsymbol{\theta}_0) \to f_{\text{lin}}(\bullet, \boldsymbol{\tau}_{t'} - \boldsymbol{\theta}_0)$

while not converged **do**

  Draw a mini-batch $\mathcal{B} \sim \mathcal{D}_{t'}$

  Minimize objective Eq. (7) on $\mathcal{B}$ w.r.t. $\boldsymbol{\tau}_{t'}$

end while

return Task vector $\boldsymbol{\tau}_{t'}$

of *representation drift*, the change in the last-layer activations of a task t when adding a new task t':

$$\left(\substack{\text{Pre-edit}\\ \text{representation}}\right)\ \boldsymbol{z}_t(\boldsymbol{x}) = f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t) \overset{\text{edit}}{\to} \boldsymbol{z}_{t,t'}(\boldsymbol{x}) = f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t + \alpha_{t'} \boldsymbol{\tau}_{t'})\ \left(\substack{\text{Post-edit}\\ \text{representation}}\right)$$
$$\implies \text{(Representation drift)}\ \ \Delta_{t\to t,t'}(\boldsymbol{x}) := \|\boldsymbol{z}_{t,t'}(\boldsymbol{x}) - \boldsymbol{z}_t(\boldsymbol{x})\|_2^2 \tag{2}$$

If the drift $\Delta_{t\to t,t'}(\boldsymbol{x})$ vanishes for all $\boldsymbol{x} \in \mathcal{D}_t$, the newly added task vector $\boldsymbol{\tau}_{t'}$ will not interfere as it does not change the model's behavior for inputs from task t. Interference between the two tasks can be reduced by penalizing representation drift Yoshida et al. (2025) via the neural network function space distance (Dhawan et al., 2023) $\mathcal{L}^{\text{drift}}_{t\to t,t'}(\boldsymbol{\tau}_{t'}) := {}^1\!/\!_{|\mathcal{D}_t|} \sum_{\boldsymbol{x}\in\mathcal{D}_t} \Delta_{t\to t,t'}(\boldsymbol{x})$. The regularizer for $\boldsymbol{\tau}_{t'}$ requires accessing data of the external task t. This may violate segregation policies, impose significant storage demands, and prevent independent training, ultimately reducing flexibility for decentralized training. These issues make direct optimization of this objective impractical in many real-world settings, such as decentralized (McMahan et al., 2017; Kairouz et al., 2021) or privacy-preserving learning scenarios (Abadi et al., 2016; Bonawitz et al., 2017).

3.1 Connecting Representation Drift Regularization to Curvature Matrices

Now, we reformulate the regularization objective to eliminate its dependence on external task data. Thanks to the linearization, the representation drift from Eq. (2) simplifies into $\Delta_{t\to t,t'}(\boldsymbol{x}) = \|\mathrm{J}_{\boldsymbol{\theta}} f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0)(\alpha_t \boldsymbol{\tau}_t - (\alpha_t \boldsymbol{\tau}_t + \alpha_{t'} \boldsymbol{\tau}_{t'}))\|_2^2 = \alpha_{t'}^2 \|\mathrm{J}_{\boldsymbol{\theta}} f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0)\, \boldsymbol{\tau}_{t'}\|_2^2$. The associated regularizer is[1]

$$\mathcal{L}^{\text{drift}}_{t\to t,t'}(\boldsymbol{\tau}_{t'}) = \alpha_{t'}^2 \boldsymbol{\tau}_{t'}^\top \boldsymbol{G}_t(\boldsymbol{\theta}_0) \boldsymbol{\tau}_{t'} \ \text{ with } \ \boldsymbol{G}_t(\boldsymbol{\theta}_0) = \tfrac{1}{|\mathcal{D}_t|} \textstyle\sum_{\boldsymbol{x}\in\mathcal{D}_t} \mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)^\top \mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \tag{3}$$

Note that the network Jacobian's Gramian $\boldsymbol{G}_t(\boldsymbol{\theta}_0) \in \mathbb{R}^{P\times P}$ – after initial pre-computation – does not require further data access. This idealized training loop is shown in Alg. 1 (black font).

In exchange for eliminating the data dependency, however, we now face the challenge of computing the $P \times P$ Gramian. This is infeasible even for small neural networks. Thankfully, we can interpret $\boldsymbol{G}_t$ as a curvature matrix that is well-known in the optimization literature: the *generalized Gauss-Newton* (GGN) matrix (Schraudolph, 2003; Martens, 2020). This connection allows us to build on well-established approaches from the optimization literature to efficiently compute structural parametric approximations of $\boldsymbol{G}_t$, ultimately allowing us to make Alg. 1 practical (red font).

3.2 The Generalized Gauss-Newton (GGN) Matrix

The GGN is a curvature matrix related to the Hessian and arises from partial linearization: The Hessian of a function composition $\ell = c \circ f$ is $\nabla^2 \ell = \nabla^2 (c \circ f)$, while the GGN is $\nabla^2 (c \circ f_{\text{lin}})$. The standard setting in the second-order optimization literature sets f to be the neural network, and c the criterion function used for training. We now introduce the GGN in this context, showing that the Jacobian Gram matrix from Eq. (3) is an instance of the GGN that results from replacing the training criterion with the squared loss. We can then easily transfer existing GGN approximations.

GGN in the training setting. Let us consider the neural network f with criterion function c (e.g. cross-entropy) and training data $\mathcal{D}$ from Sec. 2. Next, we define the prediction and criterion functions for a datum n, i.e. $f_n := f(\bullet, \boldsymbol{x}_n)$ and $c_n := c(\bullet, \boldsymbol{y}_n)$. The loss function for the example $\boldsymbol{x}_n$

[1] In the following, we suppress $_{\text{lin}}$ since the Jacobians of f and f_{lin} coincide at $\boldsymbol{\theta}_0$.

is then given by the composition $\ell_n = c_n \circ f_n$, and training seeks to minimize the empirical risk

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_n c(f(\boldsymbol{x}_n, \boldsymbol{\theta}), \boldsymbol{y}_n) := \frac{1}{|\mathcal{D}|} \sum_n \ell_n(\boldsymbol{\theta}) := \frac{1}{|\mathcal{D}|} \sum_n (c_n \circ f_n)(\boldsymbol{\theta}). \tag{4}$$

For brevity, we use c_n to denote the value $c_n(f_n(\boldsymbol{\theta}))$, and $[\bullet]_i$ for slicing (e.g. $[\boldsymbol{a}]_i$ is the i^{th} entry of $\boldsymbol{a}$). Differentiating the empirical risk twice and applying the chain rule yields the Hessian and its Gauss–Newton decomposition (Schraudolph, 2003; Martens, 2020), containing the GGN $\boldsymbol{G}(\boldsymbol{\theta})$:

$$\nabla^2 \mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{G}(\boldsymbol{\theta}) + R(\theta) := \frac{1}{|\mathcal{D}|} \sum_n (\mathrm{J}_{\boldsymbol{\theta}} f_n)^\top \nabla^2 c_n (\mathrm{J}_{\boldsymbol{\theta}} f_n) + \frac{1}{|\mathcal{D}|} \sum_n \sum_{m=1}^{C} [\nabla c_n]_m \nabla^2 [f_n]_m \,. \tag{5}$$

For a linear network, the residual term $\boldsymbol{R}(\boldsymbol{\theta})$ vanishes as it depends on second derivatives, which are zero in the linear case. Therefore, the GGN is the Hessian of the empirical risk under a linearized network ($f \leftrightarrow f_{\text{lin}}$), and is equivalent to the Fisher information matrix (Amari, 2000) for many common criterion functions (Martens, 2020).

The Jacobian's Gram matrix as GGN. The GGN in Eq. (5) generalizes the Jacobian Gram matrix from Eq. (3), used for representation drift regularization, by additionally weighting the Jacobians with the criterion function's Hessian $\nabla^2 c$. If we choose squared error $c_n(\boldsymbol{f}) = 1/2 \|\boldsymbol{f} - \boldsymbol{y}_n\|_2^2$ rather than the training criterion, the GGN becomes the Jacobian Gram matrix exactly, since $\nabla^2 c_n = \boldsymbol{I}_C$. Therefore, the coefficient matrix $\boldsymbol{G}_t(\boldsymbol{\theta}_0)$ of the quadratic form in Eq. (3) corresponds to a curvature matrix: the GGN of the loss $\mathcal{L}(\boldsymbol{\theta})$ (Eq. (4)) with the training criterion replaced by the squared loss.

While the GGN is impractically large to compute or store for neural networks, the literature has developed scalable structured approximations for it. In the following, we build on these approximations (specifically, KFAC) and study how to adapt and extend them in the context of task arithmetic.

3.3 Kronecker-Factored Approximation of the Generalized Gauss-Newton

We rely on a structured GGN approximation called *Kronecker-Factored Approximate Curvature* (KFAC) introduced by Martens & Grosse (2015) for fully-connected, then generalized to convolutional (Grosse & Martens, 2016), recurrent (Martens et al., 2018), and transformer architectures (Eschenhagen et al., 2023). KFAC has been successfully applied to optimization (Osawa et al., 2019), pruning (Wang et al., 2019), Laplace approximations (Daxberger et al., 2021; Ritter et al., 2018) and influence functions (Grosse et al., 2023). For an in-depth tutorial, see Dangel et al. (2025).

Parametric form. For a net with L layers and parameters $\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^L$, KFAC approximates the GGN as block-diagonal. Each block corresponds to a layer, $\boldsymbol{G}(\boldsymbol{\theta}) = \operatorname{blockdiag}(\boldsymbol{G}(\boldsymbol{\theta}^1), \dots, \boldsymbol{G}(\boldsymbol{\theta}^L))$, and is further approximated as a Kronecker product, $\boldsymbol{G}(\boldsymbol{\theta}^l) \approx \boldsymbol{B}^l \otimes \boldsymbol{A}^l$. To evaluate the approximation's quadratic form for representation drift regularization, we simply store the Kronecker factors $\{(\boldsymbol{B}_t^l, \boldsymbol{A}_t^l)\}_l$ from task t, then evaluate (without instantiating the Kronecker product (Loan, 2000))

$$\mathcal{L}^{\text{drift}}_{t \to t, t'}(\boldsymbol{\tau}_{t'}) = \alpha_{t'}^2 \boldsymbol{\tau}_{t'}^\top \boldsymbol{G}_t(\boldsymbol{\theta}_0) \boldsymbol{\tau}_{t'} \overset{\text{KFAC}}{\approx} \alpha_{t'}^2 \sum_{l=1}^{L} \boldsymbol{\tau}_{t'}^{l\top} (\boldsymbol{B}_t^l \otimes \boldsymbol{A}_t^l) \boldsymbol{\tau}_{t'}^l \,, \tag{6}$$

with $\boldsymbol{\tau}^l$ denoting the part of $\boldsymbol{\tau}$ corresponding to the parameters in layer l.

KFAC for a single layer. To illustrate the approximation, consider a single fully-connected layer l in a neural network, with associated weights $\boldsymbol{W}^l \in \mathbb{R}^{D_1 \times D_2}$ (we omit biases for simplicity). The layer processes an intermediate input representation $\boldsymbol{a}_n^l \in \mathbb{R}^{D_2}$ for datum $\boldsymbol{x}_n$ into an intermediate output representation $\boldsymbol{z}_n^l = \boldsymbol{W} \boldsymbol{a}_n^l \in \mathbb{R}^{D_1}$. Further, let $\boldsymbol{\theta}^l := \operatorname{vec} \boldsymbol{W}^l \in \mathbb{R}^{D_1 D_2}$ denote the row-flattened weights. The layer's GGN block is $\boldsymbol{G}(\operatorname{vec} \boldsymbol{\theta}^l) = 1/|\mathcal{D}| \sum_n (\mathrm{J}_{\boldsymbol{\theta}^l} f_n)^\top \nabla^2 c_n (\mathrm{J}_{\boldsymbol{\theta}^l} f_n)$ and simplifies into a sum of Kronecker products by using the chain rule $\mathrm{J}_{\operatorname{vec} \boldsymbol{W}^l} f_n = (\mathrm{J}_{\boldsymbol{z}_n^l} f_n)(\mathrm{J}_{\operatorname{vec} \boldsymbol{W}^l} \boldsymbol{z}_n^l)$ where $\mathrm{J}_{\operatorname{vec} \boldsymbol{W}^l} \boldsymbol{z}_n^l = \boldsymbol{I}_{D_1} \otimes \boldsymbol{a}_n^{l\top}$ (e.g. Dangel et al., 2020) to obtain

$$\boldsymbol{G}(\operatorname{vec} \boldsymbol{W}^l) = \frac{1}{|\mathcal{D}|} \sum_n (\mathrm{J}_{\boldsymbol{z}_n^l} f_n)^\top \nabla^2 c_n (\mathrm{J}_{\boldsymbol{z}_n^l} f_n) \otimes \boldsymbol{a}_n^l \boldsymbol{a}_n^{l\top} := \mathbb{E}_n[\boldsymbol{B}_n^l \otimes \boldsymbol{A}_n^l].$$

For the last equality, we use $\mathbb{E}_n[\bullet] = 1/|\mathcal{D}| \sum_n \bullet_n$ for averaging over the data set. KFAC assumes $\mathbb{E}_n[\bullet_n \otimes \star_n] \approx \mathbb{E}_n[\bullet_n] \otimes \mathbb{E}[\star_n]$, yielding a single Kronecker product involving the small factors $\boldsymbol{A}^l \in \mathbb{R}^{D_2 \times D_2}$, $\boldsymbol{B}^l \in \mathbb{R}^{D_1 \times D_1}$ to approximate the intractable block $\boldsymbol{G}(\operatorname{vec} \boldsymbol{W}^l) \in \mathbb{R}^{D_1 D_2 \times D_1 D_2}$:

$$\boldsymbol{G}(\operatorname{vec} \boldsymbol{W}^l) \overset{\text{KFAC}}{\approx} \left(\frac{1}{|\mathcal{D}|} \sum_n (\mathrm{J}_{\boldsymbol{z}_n^l} f_n)^\top \nabla^2 c_n (\mathrm{J}_{\boldsymbol{z}_n^l} f_n) \right) \otimes \left(\frac{1}{|\mathcal{D}|} \sum_n \boldsymbol{a}_n^l \boldsymbol{a}_n^{l\top} \right) := \boldsymbol{B}^l \otimes \boldsymbol{A}^l \,.$$

Variations. KFAC computes two covariances per layer: (i) the input covariance $\boldsymbol{A}^l = \mathbb{E}_n[\boldsymbol{a}_n^l \boldsymbol{a}_n^{l\top}]$, and (ii) the output gradient covariance $\boldsymbol{B}^l = \mathbb{E}_{n,m}[\boldsymbol{g}_{n,m}^l \boldsymbol{g}_{n,m}^{l\top}]$ of pseudo-gradients $\boldsymbol{g}_{n,m}^l := (\mathrm{J}_{\boldsymbol{z}_n^l} f_n)^\top \boldsymbol{s}_{n,m}$ obtained by backpropagating vectors $\boldsymbol{s}_{n,m} \in \mathbb{R}^C$ related to the Hessian $\nabla^2 c_n$. There exist different variations to compute $\boldsymbol{B}^l$ and – since it is a priori unclear which approach works best in the context of TA – we consider two variants that differ in cost (details in (Dangel et al., 2025)): *(i)* **Exact** (Botev et al., 2017) uses C backpropagations per datum and exactly computes $\boldsymbol{B}^l$; *(ii)* **Monte-Carlo** (MC, Martens & Grosse, 2015) randomizes the exact approach and computes an unbiased MC estimate of $\boldsymbol{B}^l$ using $M < C$ backpropagations per datum (typically, $M = 1$).

3.4 Multi-task Training Procedure & Regularization Merging

Naïve multi-task regularization. So far, we only considered two tasks. In practice, we want to add a new task vector given multiple existing task vectors, which introduces new challenges. To promote disentanglement when training the task vector $\boldsymbol{\tau}_{t'}$, we introduce a penalty against representation drift w.r.t. other tasks $t \neq t'$. Starting with the standard training loss $\mathcal{L}_{\mathcal{D}_{t'}}(\boldsymbol{\tau}_{t'}) = {}^1/_{|\mathcal{D}_{t'}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{t'}} c(f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\tau}_{t'} + \boldsymbol{\theta}_0), \boldsymbol{y})$, the overall fine-tuning objective becomes

$$\mathcal{L}_{\mathcal{D}_t}(\boldsymbol{\tau}_{t'}) + \beta \textstyle\sum_{t\neq t'} \lambda_t \mathcal{L}^{\text{drift}}_{t\to t,t'}(\boldsymbol{\tau}_{t'}) \overset{\text{KFAC}}{\approx} \mathcal{L}_{\mathcal{D}_{t'}}(\boldsymbol{\theta}) + \beta \sum_{t\neq t'} \lambda_t \sum_{l=1}^{L} \boldsymbol{\tau}_{t'}^{l\top} (\boldsymbol{B}_t^l \otimes \boldsymbol{A}_t^l) \boldsymbol{\tau}_{t'}^l\,, \tag{7}$$

where β and λ_t control the overall and task-specific regularization strengths, respectively. We weight tasks by data set size, $\lambda_t = {}^{|\mathcal{D}_t|}/_{\sum_{t\neq t'}|\mathcal{D}_t|}$. Given a pre-computed KFAC of each task $t \neq t'$, this formulation enables regularization without requiring direct access to data sets of external tasks.

Accumulated regularizer. A key limitation of the objective in Eq. (7) is that we must store the Kronecker factors individually for each task, incurring $\mathcal{O}(T)$ memory and run time cost. To address this, we introduce an approximation of the accumulated regularizer $\boldsymbol{G}_{-t'} = \sum_{t\neq t'} \lambda_t \boldsymbol{G}_t$, which accounts for all other tasks simultaneously using a single Kronecker product, via the further approximation

$$\boldsymbol{G}_{-t'} \overset{\text{KFAC}}{\approx} \textstyle\sum_{l=1}^{L} \sum_{t\neq t'} \lambda_t \boldsymbol{B}_t^l \otimes \boldsymbol{A}_t^l \overset{\text{merge}}{\approx} \sum_{l=1}^{L} \left(\sum_{t\neq t'} \boldsymbol{B}_t^l\right) \otimes \left(\sum_{t\neq t'} \lambda_t \boldsymbol{A}_t^l\right). \tag{8}$$

Empirically, this heuristic (Eq. (8)) matches the un-merged formulation's performance (Eq. (7)).

4 Experiments

Vision Tasks. We evaluate performance on the "8 Vision" benchmark (Ilharco et al., 2022), which covers eight classification data sets: Stanford Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), MNIST (LeCun et al., 2002), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), and SVHN (Netzer et al., 2011). We leverage CLIP (Radford et al., 2021) as foundational backbone and compare against non-linear fine-tuning (Non-Linear FT, Ilharco et al., 2022), linearized fine-tuning (Linear FT, Ortiz-Jimenez et al., 2023), and τJp (Yoshida et al., 2025), which uses external task data to mitigate task-vector interference. For each method, we collect eight checkpoints during training and subsequently merge them into a single unified model (see the supplementary materials for additional details). Following the original setup (Ortiz-Jimenez et al., 2023), we report both absolute and normalized accuracy. We further analyze the role of the rescaling coefficient α: *(i)* setting $\alpha_t = \alpha = 1$ for all tasks, corresponding to a plain addition of task vectors, and *(ii)* tuning α on a cross-task validation set.

Comparison with related works. We present a comparative analysis of our regularizer in two distinct regimes. On one hand, we evaluate it in the *linearized regime*, for which it was originally designed; on the other, we examine whether its benefits also extend to the *non-linear regime*. If so, this would broaden the applicability of our approach to most state-of-the-art learning frameworks.

Linearized fine-tuning regime. Tab. 1 reports the results on the 8Vision benchmark. We also refer to Fig. 2 (left) for a visual depiction of the per-task absolute accuracy of the merged model in the linearized regime. The results indicate that our KFAC-regularized approach yields substantial improvements against the baseline, achieving performance on par with τJp (Yoshida et al., 2025) while avoiding any reliance on external data from other tasks. This makes our method not only more flexible but also inherently privacy-preserving, without sacrificing accuracy. Furthermore, whereas competing methods often require coefficient grid search, our approach proves highly robust: even

Table 1: Task addition results on the eight vision datasets. The "α" column specifies how task vector coefficients are chosen. "1.0" denotes that all coefficients are fixed to 1.0, with no tuning.

Method	Dataless	α	ViT-B/32 Abs.	ViT-B/32 Norm.	ViT-B/16 Abs.	ViT-B/16 Norm.	ViT-L/14 Abs.	ViT-L/14 Norm.
Pre-trained	–	–	48.4	–	55.4	–	65.0	–
Individual	–	–	90.9	–	92.4	–	93.8	–
MTL	–	–	87.8	–	90.8	–	92.6	–
Linearized Fine-Tuning								
Linear FT	–	1.0	77.4	88.0	81.2	90.0	88.0	94.8
	–	Best	78.9	89.8	81.9	90.8	88.0	94.8
τJp Yoshida et al. (2025)	×	1.0	85.0	97.4	88.2	98.3	90.9	98.3
		Best	85.6	**98.2**	**88.6**	**98.7**	91.1	98.5
Diag. GGN Porrello et al. (2025)	✓	1.0	80.1	92.3	82.9	93.2	87.9	96.3
		Best	80.2	92.5	83.0	93.3	88.0	96.4
KFAC, **Ours**	✓	1.0	86.0	97.7	88.4	98.0	91.6	99.3
		Best	**86.1**	97.8	88.4	98.0	**91.6**	**99.3**

Table 2: Task addition results on the eight vision datasets under the **non-linear fine-tuning regime**.

Method	Dataless	α	ViT-B/32 Abs.	ViT-B/32 Norm.	ViT-B/16 Abs.	ViT-B/16 Norm.	ViT-L/14 Abs.	ViT-L/14 Norm.
Non Linear Fine-Tuning								
Non-linear FT	–	1.0	32.0	32.9	27.4	28.2	45.3	47.5
	–	Best	73.5	80.4	77.0	82.9	84.5	89.7
TaLoS Iurada et al. (2025)	✓	1.0	53.3	59.7	68.2	77.2	46.1	50.8
		Best	77.9	87.7	79.9	90.1	84.7	91.1
Attn. Only FT Jin et al. (2025)	–	1.0	22.5	23.3	22.8	23.4	66.2	69.7
	–	Best	78.2	86.3	80.4	87.1	88.2	93.8
Attn. Only FT + KFAC, **Ours**	✓	1.0	60.3	64.5	59.0	62.3	82.1	87.2
		Best	83.1	91.3	84.3	91.0	89.9	95.9

a simple addition of task vectors ($\alpha = 1$) performs competitively, suggesting that post-hoc tuning can be safely omitted. As a side note, the evidence on ViT-B/32 suggests that the smaller the model scale, the more crucial curvature regularization becomes for achieving strong final performance.

In this setup, we also compare against Porrello et al. (2025), which applies curvature regularization using a coarse estimate based on the **diagonal** of the Fisher Information Matrix. Both approaches exploit curvature information of the pre-trained model; however, our method relies on the KFAC approximation, which provides a more refined estimate that captures intra-layer weight dependencies. The results clearly show that the more accurate the curvature approximation, the larger the gains in Task Arithmetic. Notably, even the diagonal-based regularization improves over naïve linear fine-tuning, highlighting the central role of regularization in enabling weight disentanglement.

Non-linear fine-tuning regime. We now consider the non-linear fine-tuning regime (Tab. 2 and Fig. 2, *Right*). In this setting, alternative approaches attempt to approximate linear behavior without fully linearizing the model. For example, TaLoS Iurada et al. (2025) follows a different route and identifies a subset of parameters that consistently exhibit low gradient sensitivity across tasks and updates only these sparse components. This promotes weight disentanglement during fine-tuning while avoiding the computational bottlenecks of full linearization, enabling efficient task addition and negation. Instead, the authors of Attention-Only Fine-Tuning Jin et al. (2025) fine-tune only the attention layers of Transformers, showing that this strategy implicitly induces *kernel-like* behavior.

In this regard, although our regularization is not theoretically exact in the non-linear regime, its applicability can still be justified whenever linearized behavior is implicitly enforced. For this reason, in the non-linear setting we pair our regularizer with Attention-Only Fine-Tuning, which has been shown to induce approximately linear dynamics in Transformers, thereby providing a practical and

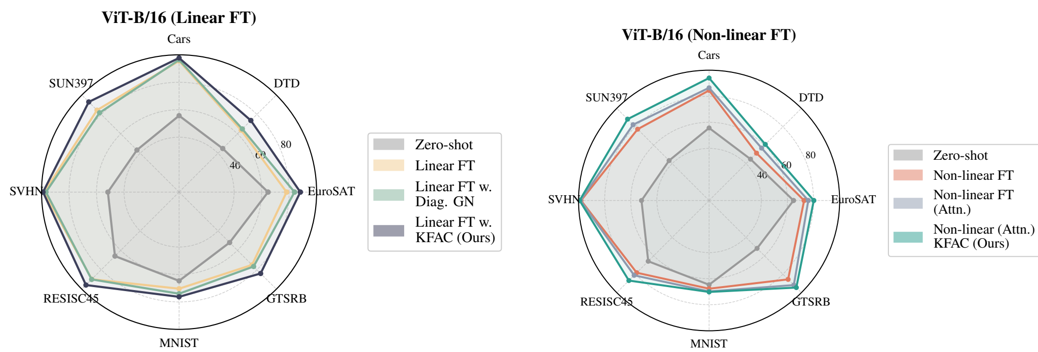


Figure 2: Impact of training and regularization choices on 8 Vision (abs. accuracy). *Left*: linearized regime, compared against the diagonal approximation (Porrello et al., 2025). *Right*: non-linear regime, compared against attention-only fine-tuning (Jin et al., 2025). See supplementary materials for full results (tables/plots) for **CLIP ViT-B/32** and **-L/14** – the finding remain consistent.

Table 3: Task negation on 8 Vision. As Ortiz-Jimenez et al. (2023), we report the minimum accuracy on the target tasks while preserving at least 95% of the pretrained model's accuracy on control tasks.

Method	**Dataless**	**ViT-B/32** Targ. ↓	Cont. ↑	**ViT-B/16** Targ. ↓	Cont. ↑	**ViT-L/14** Targ. ↓	Cont. ↑
Pre-trained	–	48.4	63.3	55.4	68.3	65.0	75.5
Non-linear FT	–	20.4	60.5	20.4	65.3	18.1	72.4
Linear FT	–	9.3	60.5	8.3	65.5	7.5	72.1
TaLoS Iurada et al. (2025)	✓	11.0	60.7	10.6	66.1	10.7	73.6
τJp Yoshida et al. (2025)	×	6.7	60.8	4.7	66.0	3.7	**73.0**
KFAC, **Ours**	✓	**3.4**	**62.4**	**3.4**	**66.4**	**3.5**	72.6

well-motivated way to extend our method beyond the strictly linearized regime. The results in Fig. 2 (*Right*) show that this is the case: when fine-tuning only attention layers, our approach proves beneficial even in the non-linear regime. Moreover, in this setting, the choice of the α coefficient has a stronger impact on the final accuracy. However, our approach still appears to be the most robust on average, a trend further corroborated by the experiment reported in App. F.2.

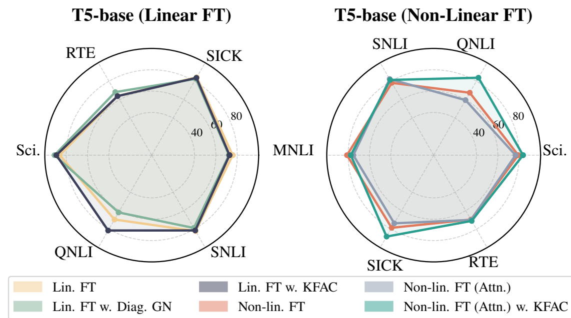
Unlearning. In Tab. 3 we investigate a setting where each task vector is subtracted from the pre-trained model. In doing so, we use ImageNet as a control task to verify whether subtraction selectively removes the corresponding task without erasing general knowledge. Our model achieves stronger forgetting of target tasks while better preserving control task, surpassing that of the main competitor, τJp (Yoshida et al., 2025). Notably, since our regularizer is dataless, it avoids the challenges associated with transferring and storing a "large" data set such as ImageNet to perform regularization. This property is particularly promising in the context of the massive data sets used today to train conversational models, where the cost of data access and management is critical.

Language tasks. Following Stoica et al. (2025), we apply our method to the T5-base model (Raffel et al., 2020) across six natural language tasks: SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), SICK (Marelli et al., 2014), SciTail (Khot et al., 2018), RTE (Wang et al., 2018), and QNLI (Wang et al., 2018). As shown in Fig. 3, in the text domain our approach consistently outperforms the baselines, particularly under non-linear fine-tuning, thus corroborating the findings observed in vision. However, leveraging data from other tasks (τJp) yields additional gains, suggesting that textual domains may still benefit from even more accurate curvature estimation.

Comparison of model merging strategies. Fig. 4 compares several existing approaches for merging task vectors, including **TIES** (Yadav et al., 2023), **DARE** (Yu et al., 2024), and the more recent state-of-the-art methods **TSV** (Gargiulo et al., 2025) and **ISO** (Marczak et al., 2025). These methods operate post-hoc, i.e., after training, and are therefore complementary to our approach, which instead acts during training. The results indicate that with naïve linear fine-tuning (yellow bars), non-trivial merging strategies such as TSV and ISO are essential to achieve good performance. In contrast, under KFAC regularization (green bars), the simple summation of task vectors (TA) already yields the

Method	Dataless	Abs.	Norm.
Individual	–	85.9	–
MTL	–	83.6	–
Non-lin. FT	–	75.7	87.7
Linear FT	–	76.9	92.8
Attn. Only FT	–	72.9	85.2
TaLoS	✓	76.3	93.4
τJp	×	**81.3**	**100**
KFAC, **Ours**	✓	78.7	98.9

(a) Task addition results for **T5-base**. All reported scores correspond to the best-performing α values; the results obtained with $\alpha = 1$ are provided in the appendix.



(b) Impact of training and regularization choices on language (abs. accuracy). *Left*: linearized regime with no regularization and with the diagonal approximation. *Right* non-linear regime, with attention-only fine-tuning with and without regularization.

Figure 3: Results for language tasks. *Left*: impact of different training strategies and sensitivity to α hyperparameter. *Right*: effects of different regularizations on linear and non-linear fine-tuning.

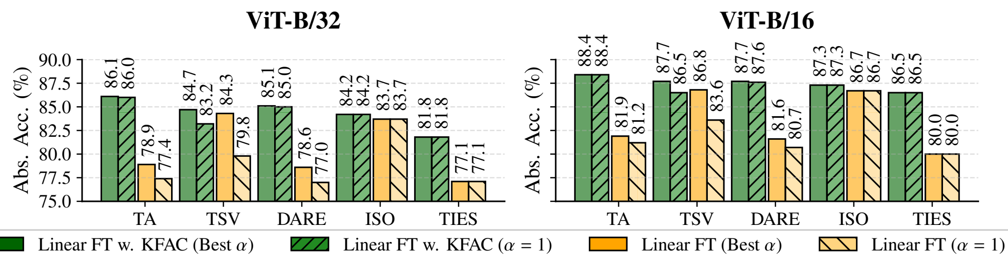


Figure 4: Effect of varying merging strategies and of grid search over the coefficient α, with and without regularization. All models are tested on checkpoints obtained through linearized fine-tuning.

best results and is robust to the choice of the merging coefficient α. This makes the approach suitable for scenarios where model merging must be performed on-the-fly and adaptively (Crisostomi et al., 2025), with negligible overhead, in order to personalize the model for specific user requests.

Naïve multi-task training *vs.* accumulated regularizer. We herein investigate the impact of the heuristic used in our approach, which accumulates the Kronecker matrices (see Eq. (8)) and thereby avoids a linear cost in the number of tasks. To this end, we run experiments using the idealized naïve multi-task training described in Eq. (7). Our findings, reported in Tab. 4, show that the gap between the idealized and the actual approach is marginal for medium-sized architectures such as ViT-B/16 in vision and T5-base in text. For ViT-B/32, we instead observe a small but consistent gap in favor of the idealized training objective, which aligns with our experience that smaller architectures tend to be more sensitive to curvature regularization and hence to the quality of the approximation.

Curvature regularization enables Task Localization. We show that our approach enables a clear separation between training and out-of-distribution examples. Indeed, given an input $\boldsymbol{x}$ and a task vector $\boldsymbol{\tau}_t$, we measure $\|\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\boldsymbol{\tau}_t\|_2^2$, which we interpret as a *normalcy score* for task t. With our regularization (Eq. (3)), these scores are indeed forced to remain low for examples outside the t-th training distribution. As illustrated in Fig. 5, this is exactly what we observe in practice: the distribution of $\|\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\boldsymbol{\tau}_t\|_2^2$ is pushed toward zero whenever the input does not belong to task t. With the naïve linear fine-tuning, this behavior is instead not as much clear. This indicates that, under KFAC curvature regularization, each task vector influences the network output only for inputs drawn from its own training distribution. Moreover, this property suggests a natural use of our method for out-of-distribution detection, as it provides a principled mechanism to assess whether an input lies within the model training distribution. A complementary analysis in the nonlinear fine-tuning regime is provided in the supplementary materials, where we compare our method against TaLoS and attention-only fine-tuning and observe that the same task-localization behavior persists.

Table 4: Our Kronecker-accumulation heuristic *vs.* the idealized multi-task formulation.

Method	**Complexity**	α	**ViT-B/32** Abs.	Norm.	**ViT-B/16** Abs.	Norm.	**T5-base** Abs.	Norm.
Naïve Multi-Task FT	$\mathcal{O}(T)$	1.0	86.5	98.4	88.0	97.5	78.5	97.0
		Best	86.6	98.5	88.1	97.6	78.5	97.0
Accumulated reg.	$\mathcal{O}(1)$	1.0	86.0	97.7	88.4	98.0	78.6	98.7
		Best	86.1	97.8	88.4	98.0	78.7	98.9

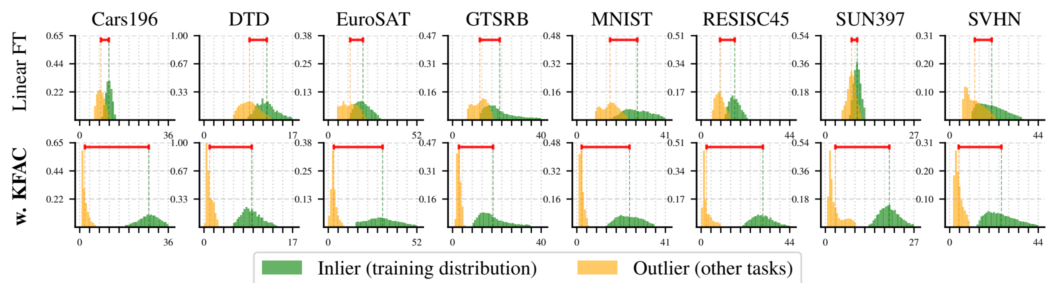
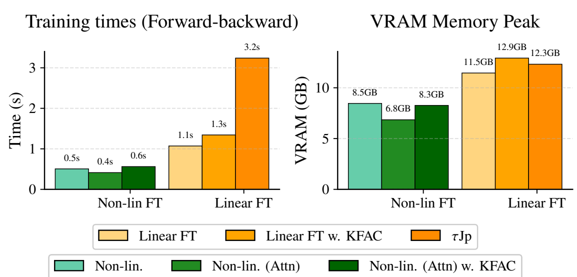


Figure 5: Distribution of $\|\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\boldsymbol{\tau}_t\|_2^2$ for inputs originating from the training distribution of task t (inliers) versus from other tasks (outliers), under both regularized and non-regularized FT.



(a) Computational overhead: training times and GPU peak.

	Exact	MC=1 (ours)
$\boldsymbol{A}$ [s]	1.4	1.4
$\boldsymbol{B}$ [s]	91.5	0.2
Total [min]	198.7	3.9

(b) Computation time for the KFAC approximation. Reported times for A and G correspond to the *average* over a batch of 8 examples, while the last row shows the total time (in minutes) required to compute the KFAC approximation for all tasks of 8Vision.

Figure 6: Analysis of the overhead of KFAC regularization during training and pre-computation.

Training costs. Fig. 6 analyzes the overhead introduced by our approach, which is twofold: estimating the KFAC matrices (before training) and computing the regularizer (during training). No overhead is introduced at inference time. With a single Monte Carlo sample, estimating all KFAC matrices for the 8 Vision tasks (128 examples per task) takes only 4 minutes, a very limited amount of time compared to the exact approach from Botev et al. (2017). During training, the overhead mainly depends on the chosen regime, with linearized fine-tuning having the largest computational footprint. Nonetheless, KFAC regularization requires only a negligible amount of additional resources, amounting to roughly one third of the training time of τJp (Yoshida et al., 2025). This efficiency arises because the τJp penalty requires a second forward–backward pass through the (slower) linearized model. Moreover, since our method does not rely on data for regularization, it avoids the repeated cost of loading new batches into GPU memory, another factor that slows down τJp.

Memory footprint. Fig. 6 (right) reports the peak VRAM usage across training regimes. KFAC introduces small increase relative to unregularized baselines: in the linearized regime, it shows a $+12\%$ overhead ($11.5 \rightarrow 2.9$ GB) w.r.t. linear fine-tuning, while in the non-linear attention-only training it shows a $+22\%$ increase ($6.8 \rightarrow 8.3$ GB). For reference, τJp peaks at 12.3 GB ($+7\%$ vs. linear FT), and standard non-linear fine-tuning reaches 8.5 GB. No memory overhead incurs at inference since regularization is inactive. Notably, aggregating all per-task KFAC factors into a single surrogate keeps the training footprint of our method at $\mathcal{O}(1)$ w.r.t. the number of tasks.

KFAC estimation. In Fig. 7, we analyze the effect of varying the number of examples and MC samples used for curvature estimation. Our findings (Fig. 7, Left) indicate that using 128–256 examples is already sufficient to saturate performance, yielding results comparable to those obtained

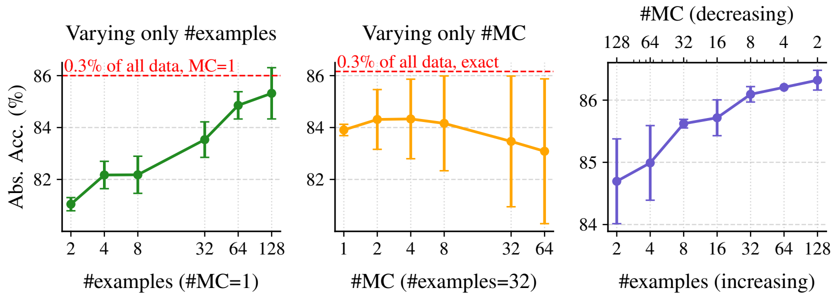
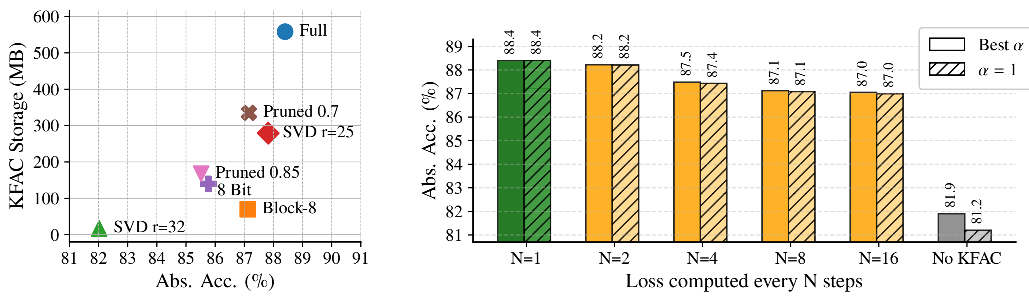


Figure 7: Impact of varying the number of examples and Monte Carlo samples for the KFAC.



(a) KFAC Storage *vs.* Accuracy. (b) Effect of KFAC Update Rate (ViT-B/16, 8Vision).

Figure 8: Memory–efficiency analysis of the proposed KFAC regularizer. (a) Accuracy under different KFAC compression strategies. (b) Effect of applying the regularization loss every N steps.

with 30% of each training set. Moreover, final performance is generally on par with that obtained with the exact approximation of Botev et al. (2017). With respect to Monte Carlo sampling, only a few samples per example (1–2) are sufficient. Surprisingly, performance deteriorates beyond this point, with variance across seeds increasing as the number of MC samples grows. Overall, increasing the number of MC samples is less effective than using more data with fewer MC samples.

Compressed KFAC. Unfortunately, the memory cost of storing KFAC matrices scales quadratically with the layer width, which may become challenging for very large models. To mitigate this cost, we evaluate how aggressively KFAC matrices can be compressed – via dynamic 8-bit quantization, structured pruning, block-diagonalization, and truncated SVD (see App. F.6) – without harming accuracy. On ViT-B/16 (8 Vision), these techniques yield substantial memory savings with only minor performance loss (Fig. 8a). The block-based strategy provides the best trade-off, decreasing memory from approximately 550 MB (full KFAC) to about 70 MB – 87% reduction – while incurring only $\sim$1-point drop in absolute accuracy (88.40 to 87.12).

We additionally analyze whether the KFAC matrices can be moved off-GPU during training without introducing prohibitive overhead. To do so, we evaluate a regime where the penalty loss is computed and backpropagated **only once** every N training steps. As illustrated in Fig. 8b, applying the loss every 16 steps leads to a modest degradation ($\sim$1.4 points) relative to applying it at every iteration. This demonstrates that scheduling curvature updates can effectively amortize memory transfers and enable GPU–CPU factor shuffling without compromising the usefulness of the regularizer.

5 Conclusions

We investigate curvature-based regularization as a means to enhance Weight Disentanglement in Task Arithmetic. Our approach is dataless, efficient, and effective, making the simple summation of task vectors competitive with state-of-the-art merging strategies, without the need for additional tuning. We demonstrate its applicability in both linearized and non-linear regimes, and show that it enables a clear separation between in- and out-of-distribution examples. Our work calls for releasing additional assets together with the pre-trained weights without having to open-source the training data. Such information, e.g. gradient accumulators of the adaptive optimizer used for training (Li et al., 2025), or in our case KFAC, enable further downstream applications with foundation models.

Reproducibility statement

The full codebase used to run our experiments is released along with the paper to facilitate future research.

Disclosure on the Use of Language Models

Large Language Models (LLMs) were used exclusively to improve the clarity and polish of the writing. All scientific ideas, methodological contributions, experimental designs, analyses, and conclusions presented in this paper originate entirely from the authors.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Alessandro Achille, Aditya Golatkar, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Lqf: Linear quadratic fine-tuning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 15729–15739, 2021.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 2000.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

Sanjeev Arora, Simon S Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. *International Conference on Learning Representations*, 2020.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*. PMLR, 2017.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 632–642, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014.

Donato Crisostomi, Alessandro Zirilli, Antonio Andrea Gargiulo, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, Iacopo Masi, and Emanuele Rodolà. Mass: Moerging through adaptive subspace selection. *arXiv preprint arXiv:2504.05342*, 2025.

Felix Dangel, Stefan Harmeling, and Philipp Hennig. Modular block-diagonal curvature approximations for feedforward architectures. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Felix Dangel, Runa Eschenhagen, Bálint Mucsányi, and Tobias Weber. Kfac from scratch. *arXiv*, 2025. URL https://github.com/f-dangel/kfac-tutorial.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless bayesian deep learning. In *Advances in Neural Information Processing Systems*, 2021.

Nikita Dhawan, Sicong Huang, Juhan Bae, and Roger Baker Grosse. Efficient parametric approximations of neural network function space distance. In *International Conference on Machine Learning*, 2023.

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems*, 2023.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 18695–18705, 2025.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 792–801, 2021.

Roger Grosse and James Martens. A kronecker-factored approximate Fisher matrix for convolution layers. In *International Conference on Machine Learning*, 2016.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *IEEE International Conference on Computer Vision*, 2021.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2022.

Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning. In *International Conference on Learning Representations*, 2025.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Ruochen Jin, Bojian Hou, Jiancong Xiao, Weijie Su, and Li Shen. Fine-tuning attention modules only: Enhancing weight disentanglement in task arithmetic. *International Conference on Learning Representations*, 2025.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Yu Xin Li, Felix Dangel, Tam Derek, and Colin Raffel. Fishers for free? approximating the fisher information matrix by recycling the squared gradient accumulator. In *International Conference on Machine Learning (ICML)*, 2025.

Wu Lin, Felix Dangel, Runa Eschenhagen, Kirill Neklyudov, Agustinus Kristiadi, Richard E. Turner, and Alireza Makhzani. Structured inverse-free natural gradient descent: Memory-efficient & numerically-stable KFAC. In *International Conference on Machine Learning (ICML)*, 2024.

Tian Yu Liu and Stefano Soatto. Tangent model composition for ensembling and continual fine-tuning. In *IEEE International Conference on Computer Vision*, pp. 18676–18686, 2023.

Tian Yu Liu, Aditya Golatkar, and Stefano Soatto. Tangent transformers for composition, privacy and removal. In *International Conference on Learning Representations*, 2024.

Charles F.Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 2000.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In *International Conference on Machine Learning*, 2025.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 1–8, 2014.

James Martens. Deep learning via Hessian-free optimization. In *International Conference on Machine Learning*, 2010.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21:1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*. PMLR, 2015.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2017.

Fangzhou Mu, Yingyu Liang, and Yin Li. Gradients as features for deep representation learning. In *International Conference on Learning Representations*, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Granada, Spain, 2011. URL `http://ufldl.stanford.edu/housenumbers/`.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019.

Angelo Porrello, Lorenzo Bonicelli, Pietro Buzzega, Monica Millunzi, Simone Calderara, and Rita Cucchiara. A second-order perspective on model compositionality and incremental learning. *International Conference on Learning Representations*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Yi Ren, Shangmin Guo, Wonho Bae, and Danica J Sutherland. How to prepare your task head for finetuning. In *International Conference on Learning Representations*, 2023.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 2018.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pp. 535–542, 2003.

Hyounguk Shon, Janghyeon Lee, Seung Hwan Kim, and Junmo Kim. Dlcft: Deep linear continual fine-tuning for general incremental learning. In *Proceedings of the European Conference on Computer Vision*, pp. 513–529. Springer, 2022.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*. IEEE, 2011.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. In *International Conference on Learning Representations*, 2025.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter efficient multi-task model fusion with partial linearization. In *International Conference on Learning Representations*, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International Conference on Machine Learning*, 2019.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. In *International Conference on Machine Learning*, pp. 23549–23588. PMLR, 2022.

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

Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119 (1):3–22, 2016.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2023.

Kotaro Yoshida, Yuji Naraki, Takafumi Horie, Ryosuke Yamaki, Ryotaro Shimizu, Yuki Saito, Julian McAuley, and Hiroki Naganuma. Mastering task arithmetic: τjp as a key indicator for weight disentanglement. In *International Conference on Learning Representations*, 2025.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*, 2024.

A Appendix / Supplementary Material

The appendix is organized as follows:

- App. B discusses the main limitations of our approach, including memory requirements and curvature-estimation challenges.
- App. C provides a derivation and a formal bound on the approximation error introduced when merging multiple K-FAC factors using the Kronecker heuristic.
- App. D presents additional plots illustrating the disentanglement error.
- App. E details the implementation of our methods, with separate discussions for the vision and text domains.
- App. F reports additional experiments. These include:
  - Core analyses:
    * per-task performance analysis,
    * alpha-sweep robustness study (App. F.2),
    * ablation on the regularization coefficient (App. F.3),
    * evaluation of a shared KFAC computed on a reference dataset (App. F.4),
    * task-localization analysis under nonlinear fine-tuning (App. F.5);
  - extended experiments:
    * analysis of task localization under memory-efficient KFAC approximations, including block-based, SVD-based, pruning, and 8-bit quantized variants (App. F.6),
    * additional results on more challenging vision domains using a class-incremental partitioning protocol (App. F.7).
- App. G provides a concise overview of prior work on linearized fine-tuning and its recent developments.

B Limitations

KFAC requires storing the Kronecker matrices in GPU memory – two per layer, each with quadratic complexity in the number of units. For large models this can become problematic, suggesting that alternative strategies based on matrix compression or structured Kronecker factors (Grosse et al., 2023; Lin et al., 2024) should be explored. While we combine the well-established KFAC with an accumulation strategy, designing curvature approximations that can easily be merged without sacrificing accuracy may be worth exploring in the future. Moreover, our experiments in the text domain indicate room for improvement, raising the question of whether more sophisticated techniques for curvature estimation could further enhance Task Arithmetic.

C Approximation Error of the Merged K-FAC Factors

For clarity, we focus on a single layer and assume all layers contribute equally, omitting the task weights λ_t. Let $\{A_t\}_{t=1}^T$ and $\{B_t\}_{t=1}^T$ denote the K-FAC factors associated with the tasks involved in the merge. The heuristic used in Eq. 8 replaces the sum of Kronecker products with the Kronecker product between aggregated factors

$$\sum_{t=1}^{T} B_t \otimes A_t \approx \left(\sum_{t=1}^{T} B_t\right) \otimes \left(\frac{1}{T}\sum_{t=1}^{T} A_t\right). \tag{9}$$

We now provide a simple bound that quantifies the error introduced by this approximation. To do so, we define the empirical means and the deviations from the mean

$$\bar{A} = \frac{1}{T}\sum_{t=1}^{T} A_t, \qquad \bar{B} = \frac{1}{T}\sum_{t=1}^{T} B_t, \qquad \Delta A_t = A_t - \bar{A}, \qquad \Delta B_t = B_t - \bar{B}. \tag{10}$$

Note that, by construction, $\sum_t \Delta A_t = \sum_t \Delta B_t = 0$. Substituting $A_t = \bar{A} + \Delta A_t$ and $B_t = \bar{B} + \Delta B_t$ into the left-hand side of Eq. (9) yields

$$\sum_{t=1}^{T} B_t \otimes A_t = \sum_{t=1}^{T} (\bar{B} + \Delta B_t) \otimes (\bar{A} + \Delta A_t) \tag{11}$$

$$= \sum_{t=1}^{T} \Big(\bar{B} \otimes \bar{A} + \bar{B} \otimes \Delta A_t + \Delta B_t \otimes \bar{A} + \Delta B_t \otimes \Delta A_t \Big) \tag{12}$$

$$= \underbrace{\sum_{t=1}^{T} \bar{B} \otimes \bar{A}}_{T\,\bar{B}\otimes\bar{A}} \; + \; \underbrace{\bar{B} \otimes \sum_{t=1}^{T} \Delta A_t}_{=0} \; + \; \underbrace{\left(\sum_{t=1}^{T} \Delta B_t \right) \otimes \bar{A}}_{=0} \; + \; \sum_{t=1}^{T} \Delta B_t \otimes \Delta A_t \tag{13}$$

$$= T\,\bar{B} \otimes \bar{A} \; + \; \sum_{t=1}^{T} \Delta B_t \otimes \Delta A_t. \tag{14}$$

Substituting $A_t = \bar{A} + \Delta A_t$ and $B_t = \bar{B} + \Delta B_t$ into the right-hand side of Eq. (9), instead, yields

$$\left(\sum_{t=1}^{T} B_t \right) \otimes \left(\sum_{t=1}^{T} A_t \right) = T^2\,\bar{B} \otimes \bar{A}. \tag{15}$$

Hence the approximation error is

$$E := \sum_{t=1}^{T} B_t \otimes A_t \; - \; \frac{1}{T} \left(\sum_{t=1}^{T} B_t \right) \otimes \left(\sum_{t=1}^{T} A_t \right) = \sum_{t=1}^{T} \Delta B_t \otimes \Delta A_t.$$

Error bound. Using the Frobenius norm and the property $\|X \otimes Y\|_F = \|X\|_F \, \|Y\|_F$, we obtain

$$\|E\|_F \leq \sum_{t=1}^{T} \|\Delta B_t\|_F \, \|\Delta A_t\|_F \leq \sqrt{\sum_{t=1}^{T} \|\Delta B_t\|_F^2} \, \sqrt{\sum_{t=1}^{T} \|\Delta A_t\|_F^2}. \tag{16}$$

Defining the deviations (standard deviations in matrix space), we obtain:

$$\sigma_A := \sqrt{\frac{1}{T} \sum_{t=1}^{T} \|\Delta A_t\|_F^2}, \qquad \sigma_B := \sqrt{\frac{1}{T} \sum_{t=1}^{T} \|\Delta B_t\|_F^2}, \tag{17}$$

we finally obtain the compact bound

$$\|E\|_F \; \leq \; T\,\sigma_A\,\sigma_B. \tag{18}$$

Interpretation. The approximation error is proportional to the product of the variations of the K-FAC factors across tasks. When the task-specific factors (A_t, B_t) cluster tightly around their means, both σ_A and σ_B are small, yielding a small deviation between the true mixed K-FAC term and its merged approximation. This situation is particularly likely to occur when the matrices are estimated from a fixed pre-trained backbone such as CLIP: since the underlying feature extractor remains unchanged across tasks, the induced activation and gradient statistics tend to vary only mildly. As a result, the corresponding K-FAC factors exhibit limited task-to-task fluctuation, further justifying the accuracy of the merged approximation.

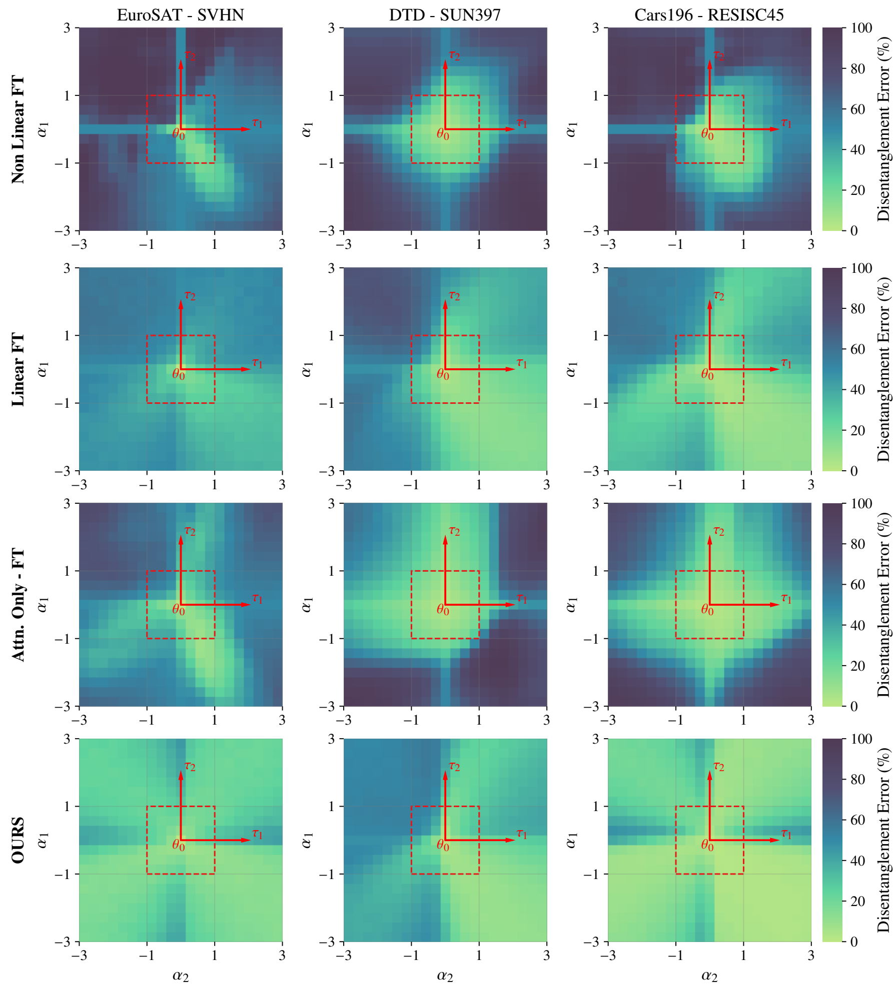


Figure 9: Visualization of weight disentanglement (Ortiz-Jimenez et al., 2023) in ViT-B/16. Non linear fine-tuning Ilharco et al. (2022), Linear fine-tuning Ortiz-Jimenez et al. (2023), Attention-Only fine-tuning Jin et al. (2025), Linear fine-tuning with KFAC regularization.

D Additional plots on Weight Disentanglement

In Fig. 9 we report the disentanglement error, a metric introduced by Ortiz-Jimenez et al. (2023):

$$\xi(\alpha_1, \alpha_2) = \sum_{t=1}^{2} \mathbb{E}_{\boldsymbol{x}\sim\mu_t}\left[\text{dist}\left(f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t), f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1 + \alpha_2 \boldsymbol{\tau}_2)\right)\right], \tag{19}$$

where $\text{dist}(y_1, y_2) = \mathbb{1}(y_1 \neq y_2)$. When $\xi(\alpha_1, \alpha_2) = 0$, tasks τ_1 and τ_2 merge without interference for the corresponding values of α_1 and α_2.

As shown in the plots, linearized fine-tuning substantially improves the disentanglement of task vectors. This property is further enhanced under our regularization regime, where only a few darker regions remain, mostly for $\alpha > 1$, a setting that is never used in practice. Notably, in our experiments the disentanglement error is consistently close to zero along the diagonals, which is the most relevant case, since in the literature the common choice is $\alpha_1 = \alpha_2 = \cdots = \alpha_n$.

E Implementation Details

The GGN information matrices were estimated using a single Monte Carlo sample and computed on 33% of the available training data. However, our empirical analysis showed that sampling only 250-300 training points is sufficient to obtain a reliable estimation of the curvature matrix.

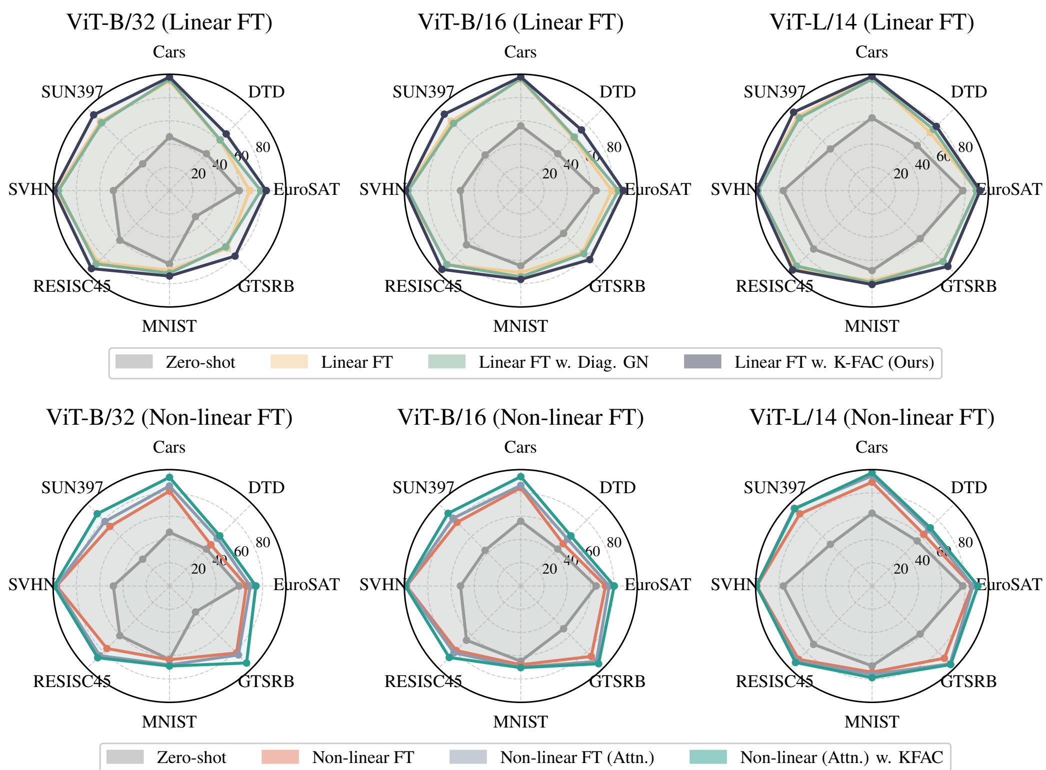


Figure 10: Impact of training and regularization choices on vision tasks (absolute accuracy). Top: linearized regime, compared against the diagonal approximation. Bottom: non-linear regime, compared against attention-only fine-tuning.

E.1 Vision Domain

For training the task vectors, we followed the setup of previous works Ilharco et al. (2022); Ortiz-Jimenez et al. (2023); Yoshida et al. (2025), adopting a batch size of 128. We used the AdamW optimizer with a learning rate of 3×10^{-4}, weight decay of 0.1, and a cosine annealing learning rate scheduler. Unlike prior approaches, we did not apply gradient clipping during training. The regularization term in the loss was weighted by $\lambda = 100$ for ViT-B/32, $\lambda = 500$ for ViT-B/16, and $\lambda = 2000$ for ViT-L/14.

Compared to previous work, we employed a higher learning rate. Since our formulation includes an explicit regularization term in the loss, this allowed us to increase the learning rate without introducing interference across tasks.

E.2 Text Domain

SNLI, MultiNLI, and SICK are three-way classification tasks where the relation between a premise and a hypothesis must be identified as entailment, contradiction, or neutral. In contrast, SciTail, RTE, and QNLI are binary entailment tasks, and therefore fine-tuning and evaluation are restricted to two labels.

For training language task vectors, we adopted a batch size of 128, using an AdamW optimizer with a learning rate of 3×10^{-4} with an iteration-based cosine-annealing scheduler and a weight decay of 0.01. Like in vision tasks, we did not apply gradient clipping during training. The regularization term in the loss is set to $\lambda = 20$ for the KFAC regularization and to $\lambda = 0.1$ for the diagonal regularization.

Table 5: 8 Vision - Comparison of different merging strategies on task vectors obtained in the linear fine tuning regime Ortiz-Jimenez et al. (2023) with and without KFAC regularization.

Method	α	**ViT-B/32**		**ViT-B/16**	
		Abs.	Norm.	Abs.	Norm.
Linear FT + TA Ilharco et al. (2022)	1.0	77.4	88.0	81.2	90.0
	Best	78.9	89.8	81.9	90.8
Linear FT + TIES Yadav et al. (2023)	1.0	77.1	87.6	80.0	88.6
	Best	77.1	87.6	80.0	88.6
Linear FT + ISO Marczak et al. (2025)	1.0	83.7	95.5	86.7	96.4
	Best	83.7	95.5	86.7	96.4
Linear FT + TSV Gargiulo et al. (2025)	1.0	79.8	90.7	83.6	92.7
	Best	84.3	96.2	86.8	96.5
Linear FT + DARE Yu et al. (2024)	1.0	77.0	87.5	80.7	89.4
	Best	78.6	89.6	81.6	90.5
KFAC, **Ours** + TA Ilharco et al. (2022)	1.0	86.0	97.7	88.4	98.0
	Best	**86.1**	**97.8**	**88.4**	**98.0**
KFAC, **Ours** + TIES Yadav et al. (2023)	1.0	81.8	92.5	86.5	95.6
	Best	81.8	92.5	86.5	95.6
KFAC, **Ours** + ISO Marczak et al. (2025)	1.0	84.2	95.5	87.3	96.7
	Best	84.2	95.5	87.3	96.7
KFAC, **Ours** + TSV Gargiulo et al. (2025)	1.0	83.2	94.3	86.5	95.7
	Best	84.7	96.2	87.7	97.1
KFAC, **Ours** + DARE Yu et al. (2024)	1.0	85.0	96.5	87.6	97.0
	Best	85.1	96.6	87.7	97.1

F Additional Experiments

In this section we present the results of additional experiments on task addition conducted on the 8Vision dataset, complementing those already reported in the main paper.

F.1 Performance

Fig. 10 provides a per-task breakdown of the same experiment reported in Tab. 1. Interestingly, the larger ViT-L/14 backbone exhibits smaller relative gains from regularization, particularly in the non-linear regime, where its behavior closely resembles that of its linearized counterpart. Consistent with prior work Ortiz-Jimenez et al. (2023), this suggests that very large models may already display an implicit form of regularization. Conversely, the ViT-B/32 benefits the most from regularization, showing that smaller architectures require more careful fine-tuning to enable effective task arithmetic.

Finally, Tab. 5 reports both absolute and normalized accuracy for different merging strategies: **TIES** (Yadav et al., 2023), **DARE** (Yu et al., 2024), **TSV** (Gargiulo et al., 2025), and **ISO** (Marczak et al., 2025), when applied to task vectors obtained through linearized fine-tuning with and without regularization. As also shown in Fig. 4 and discussed in the main paper, our results indicate that without regularization, non-trivial merging strategies such as TSV and ISO are essential to achieve strong performance. In contrast, under KFAC regularization, simple task arithmetic (TA) already provides the best results and remains robust to the choice of the merging coefficient α.

F.2 Robustness Under Task Arithmetic: Alpha-Sweep Analysis

In this section, we evaluate how different fine-tuning strategies behave when performing task arithmetic, focusing on the stability of performance as the task-vector scaling coefficient α varies in the range $[0, 1]$. The evaluation follows the standard task-arithmetic setup, where multiple task vectors are combined through simple summation. A method is considered robust if its accuracy varies smoothly across the sweep and remains stable over a broad interval of α values.

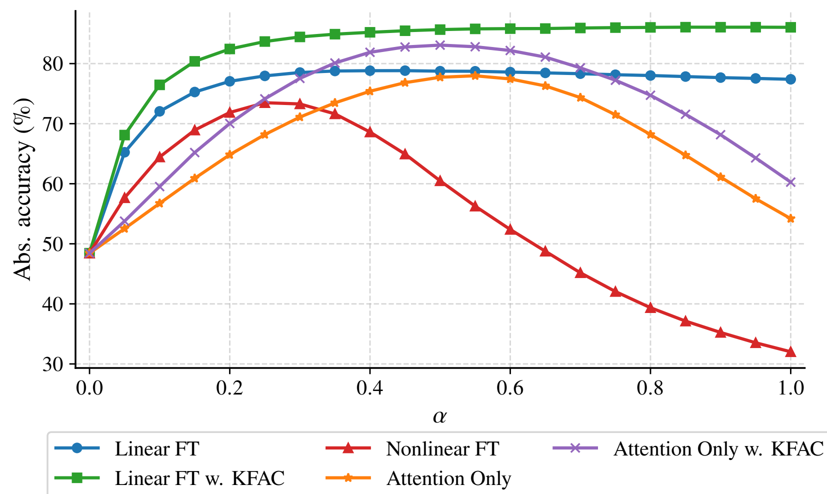


Figure 11: α-sweep analysis on ViT-B/32 (8Vision) under task arithmetic. Accuracy is reported as a function of the scaling coefficient $\alpha \in [0, 1]$. The linearized KFAC-regularized model shows the highest robustness across all α, while in the nonlinear regime it consistently outperforms attention-only fine-tuning (Jin et al., 2025).

Table 6: On 8Vision, ablation of λ on ViT-B/32 (left) and ViT-B/16 (right). All performances are reported in terms of absolute accuracy using $\alpha = 1$.

ViT-B/32				
λ	**Seed** 7	**Seed** 21	**Seed** 42	**AVG.**
0	75.0	75.4	75.1	75.2 ± 0.028
1	82.2	82.4	80.6	81.7 ± 0.648
10	85.2	85.1	85.1	85.1 ± 0.002
100	86.2	85.8	86.0	86.0 ± 0.026
1000	86.5	86.4	86.4	86.4 ± 0.002
10000	84.5	84.4	84.3	84.4 ± 0.006

ViT-B/16				
λ	**Seed** 7	**Seed** 21	**Seed** 42	**AVG.**
0	79.1	78.7	79.1	79.0 ± 0.188
1	83.2	83.4	83.8	83.5 ± 0.265
50	86.9	86.8	87.0	86.9 ± 0.059
500	88.0	87.9	88.2	88.0 ± 0.114
5000	88.3	88.4	88.4	88.4 ± 0.015
50000	86.7	86.6	86.6	86.6 ± 0.002

We compare several fine-tuning strategies: naive nonlinear fine-tuning, linear fine-tuning, attention-only fine-tuning (Jin et al., 2025), and our KFAC curvature regularization (under both linearized and nonlinear training regimes). The results, shown in Fig. 11, reveal that our linearized model with curvature regularization is consistently the most robust across the entire α sweep. It maintains high accuracy for all values of α and exhibits the smallest sensitivity to task-vector scaling, while other methods display markedly less stable behavior. In the nonlinear setting, our method continues to outperform attention-only fine-tuning across all α values, confirming that the benefits of our approach extend beyond the linear regime.

Overall, this analysis indicates that curvature regularization not only improves absolute performance but also enhances the reliability of task arithmetic, enabling stable and predictable model behavior even when combining multiple task vectors.

F.3 Ablation on the Regularization Coefficient

This section presents an ablation study investigating the impact of the scaling coefficient λ applied to the regularization term in the loss function. In Tab. 6 we evaluate the performance of ViT-B/32 and ViT-B/16 using six values of the regularization coefficient, ranging over five orders of magnitude from 0 to 10^4, and repeated each experiment with three random seeds. The case $\lambda = 0$ serves as the baseline, corresponding to non-regularized fine-tuning. It should be noted that these results differ from those reported in Tab. 1, as the linear fine-tuning therein follows the hyperparameter configuration of Ilharco et al. (2022), whereas the experiments presented here employ the hyperparameter setting described in App. E.

Table 7: Task addition results on the eight vision datasets when using either task-specific KFAC factors or a single shared KFAC computed on ImageNet-1k. Results show that a universal, task-agnostic KFAC (ImageNet-KFAC) retains most of the benefits of our regularizer while requiring no access to auxiliary task-specific data.

Method	**Dataless**	α	**ViT-B/32** Abs.	**ViT-B/32** Norm.	**ViT-B/16** Abs.	**ViT-B/16** Norm.
Linear FT	–	1.0	77.4	88.0	81.2	90.0
	–	Best	78.9	89.8	81.9	90.8
KFAC, **Ours**	✓	1.0	86.0	97.7	88.4	98.0
		Best	**86.1**	97.8	88.4	98.0
ImageNet-KFAC, **Ours**	✓	1.0	84.7	97.0	86.0	95.4
		Best	84.7	97.0	86.0	95.4

The results indicate that the proposed method is robust with respect to the choice of λ. Optimal performance is observed for values of λ between 10^2 and 10^3, while only minor degradation occurs for $\lambda = 10$ and $\lambda = 10^4$. This behavior confirms that successful model merging primarily depends on the presence of regularization based on information from the Generalized Gauss-Newton matrix, and that the magnitude of this term must be sufficiently emphasized. However, the results also show that no precise tuning of λ is required to achieve strong performance.

F.4 Eliminating Task Dependence with a Universal KFAC

Although our framework completely removes the need for raw auxiliary data, it still requires pre-computed input and gradient covariance factors from the tasks to be disentangled. This dependence may be limiting in scenarios where such factors cannot be shared due to practical difficulties in storing or distributing task-specific curvature statistics, or simply because the set of tasks to be composed is not known in advance at training time.

To assess whether this dependence can be relaxed, we test whether broad curvature statistics – extracted from a large, natural-image distribution – can serve as a proxy and effectively replace the per-task KFAC factors. In details, we build a variant, denoted *ImageNet-KFAC*, in which every layer uses a single pair of A/B matrices computed on ImageNet-1k. Ideally, these factors capture universal visual covariances, and hence they can remain fixed for all downstream tasks. During fine-tuning, these shared factors can entirely substitute the task-specific ones normally employed by our regularizer.

As shown in Tab. 7, despite using non–task-specific information, this proxy KFAC recovers approximately 97–99$\%$ of the performance obtained with full task-specific factors on both ViT-B/16 and ViT-B/32 (8Vision). The absolute accuracy reached by the ImageNet-KFAC variant is 84.7% on ViT-B/32 and 86.0% on ViT-B/16, closely matching the performance of the original approach while substantially surpassing diagonal or no-regularization baselines as well as competitive alternatives such as TaLoS or attention-only fine-tuning.

These results indicate that a task-agnostic curvature prior, captured by a single shared factorization, delivers most of the benefits of our dataless regularizer without accessing any task-specific statistics. In practical scenarios, this makes the method fully data-agnostic with respect to the problem, effectively eliminating any residual coupling to external tasks.

F.5 Task Localization Under Non-Linear Fine-Tuning

In this section we extend the task-localization analysis presented in the main paper to the nonlinear fine-tuning regime. The goal is to assess whether the separation between in-task and out-of-task examples, induced by our curvature regularizer under linearized training, persists when full model parameters are updated. In details, we measure the same editing-localization metric used in the main paper, namely the difference between the Jacobian-projected output variation $\|\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\, \boldsymbol{\tau}_t\|_2^2$ for inputs belonging to task t versus those coming from other tasks.

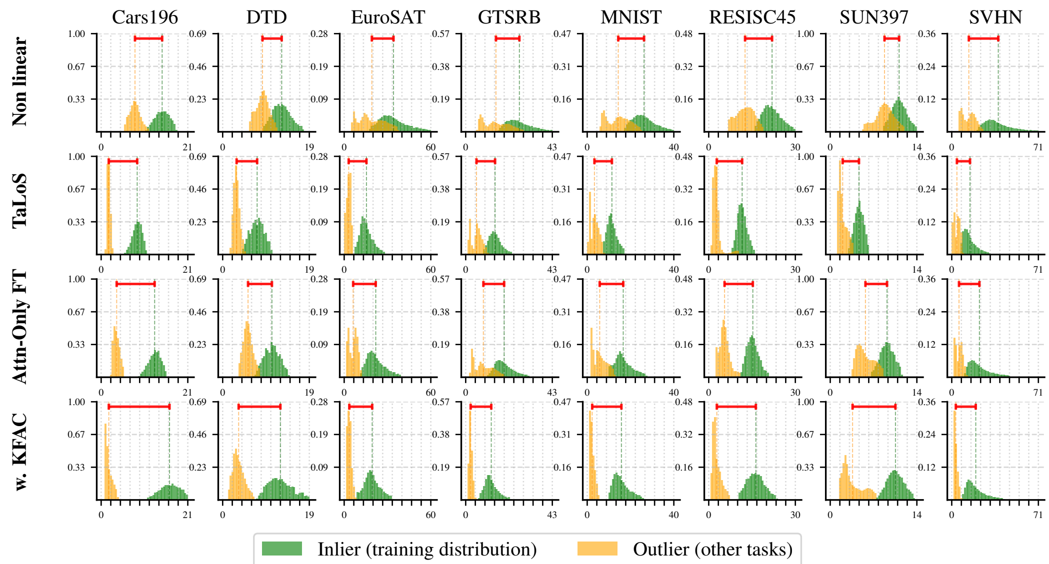


Figure 12: Task localization under **non-linear fine-tuning**. We report the distribution of the Jacobian-projected normalcy scores $\|\mathrm{J}_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\, \boldsymbol{\tau}_t\|_2^2$ for inputs belonging to task t (in-task) versus inputs from all other tasks (out-of-task).

As shown in Fig. 12, We evaluate four methods: the standard non-linear fine-tuning, TaLoS Iurada et al. (2025), attention-only fine-tuning Jin et al. (2025), and our proposed KFAC-based curvature regularizer. For each approach, we fine-tune the model in the fully nonlinear setting and compute the distribution of normalcy scores for in-task and out-of-task inputs.

The results show a consistent pattern across all datasets. Our method maintains a clear and sharp separation between in-distribution and out-of-distribution examples, closely mirroring the behavior observed under the linearized regime. TaLoS and attention-only fine-tuning preserve part of this effect but yields a weaker distinction. Overall, these findings confirm that curvature regularization continues to restrict the influence of each task vector to its corresponding training distribution even when the full network is fine-tuned.

F.6 KFAC Compression Strategies and Task Localization

To assess the robustness of our curvature regularizer under memory constraints, we evaluate several compression strategies applied directly to the KFAC factors. All strategies described below are applied independently to both A and B matrices for every layer.

The first strategy is a **block-diagonal approximation** ("Block 8"), in which each factor is partitioned into eight equally sized blocks along the main diagonal, with all off-diagonal blocks discarded. This yields a substantial reduction in memory while maintaining a structured representation and preserving dominant second-order interactions.

The second strategy relies on **truncated SVD**. Given the factorization $A = U\Sigma V^\top$, we keep only the top singular components, either by selecting a fixed rank (32 in our experiments) or by retaining a percentage of the original rank (25%). The truncated reconstruction $\tilde{A} = U_k \Sigma_k V_k^\top$ provides a low-rank surrogate that preserves the principal curvature directions.

A third strategy applies unstructured **magnitude pruning**. Each KFAC matrix is converted to COO sparse format, and only the largest-magnitude entries are preserved. We consider two keep ratios, 30% and 15%, corresponding to increasingly aggressive sparsification. All remaining entries are set to zero, effectively reducing memory and bandwidth requirements.

Finally, we evaluate **dynamic 8-bit quantization**. Each factor is quantized on-the-fly to an 8-bit integer representation, with per-row scaling ensuring that reconstruction errors remain controlled.

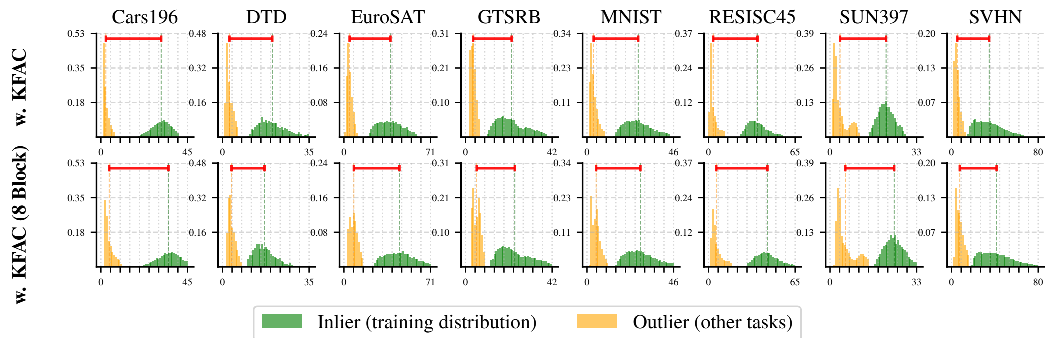


Figure 13: Task localization under linearized fine-tuning with block-compressed KFAC. The separation between the two distributions closely matches that of the full KFAC model, indicating that the block-based compression has negligible impact on task localization and that curvature-based task isolation remains robust even under aggressive memory reductions.

Task localization. We further investigate whether the task-localization behavior observed in the main paper remains stable when applying memory-efficient KFAC approximations. In particular, we focus on the block-based compression strategy, where each KFAC factor is decomposed into 8 diagonal blocks, substantially reducing storage while preserving the structure of the Kronecker approximation. This variant is the most promising among those we evaluated, as it consistently provides the best trade-off between memory savings and accuracy.

The results, shown in Fig. 13, reveal that the block-based KFAC approximation preserves the same localization behavior as the full KFAC model. Even with only eight diagonal blocks per factor, the model continues to sharply distinguish in-distribution from out-of-distribution samples. The compression therefore appears to have negligible impact on this diagnostic, suggesting that curvature-based task localization is robust to coarse, memory-friendly KFAC approximations.

TEXT DOMAIN: RESULTS FOR $\alpha = 1$

Results for $\alpha = 1$. Following the setup described in the main text for language tasks, where we evaluate T5-base using the fixed hyperparameter value $\alpha = 1$. As discussed in Fig. 3, our method exhibits consistently strong performance in the text domain, mirroring the trends observed in the vision setting.

Method	Dataless	Abs.	Norm.
Individual	–	85.9	–
MTL	–	83.6	–
Non-lin. FT	–	65.5	75.9
Linear FT	–	76.1	92.0
Attn-Only FT	–	67.0	78.3
TaLoS	✓	75.8	92.8
τJp	×	**81.0**	**99.5**
KFAC, **Ours**	✓	78.6	98.7

Figure 14: Task addition results for **T5-base** with $\alpha = 1$.

F.7 EXPERIMENT ON OTHER VISION DOMAINS

In Tab. 8 we present additional experiments on a different vision domain to further assess the effectiveness of KFAC regularization on less trivial tasks. Following (Porrello et al., 2025), each dataset is split into partitions containing distinct classes. This procedure ensures task diversity while keeping the domain consistent, since all partitions originate from the same dataset. The number of classes per partition depends on the dataset: ImageNet-R (Hendrycks et al., 2021) is divided into 10 tasks of 20 classes each, RESISC45 Krizhevsky et al. (2009) into 9 tasks of 5 classes each, and EuroSAT (Helber et al., 2019) into 5 tasks of 2 classes each. After fine-tuning the base model on each partition, the resulting models are merged and evaluated on the full test set, considering the union of all classes across tasks rather than restricting evaluation to the classes of the training task

Table 8: Performance comparison across different regularization strategies on ViT-B/16

Model	**ImageNet-R**	**EUROSAT**	**RESISC**
Zero-shot	77.72	49.48	66.02
Non-linear FT	82.32	71.21	73.85
Linear FT	81.66	70.40	72.28
Linear FT w. Diag. GN	81.64	73.94	74.04
τjP Yoshida et al. (2025)	81.28	84.36	84.83
KFAC, **Ours** (naive penalty)	82.64	79.64	78.91
KFAC, **Ours** (aggregated penalty)	82.63	79.64	78.30

only, as done in the 8 Vision benchmark. Accuracy is then reported on this joint classification problem, following the protocol of (Porrello et al., 2025). These experiments demonstrate that KFAC regularization achieves state-of-the-art performance even under this more challenging setting.

G Related works on Linearized Fine-Tuning

Linearized models offer a principled lens for analyzing fine-tuning by considering first-order expansions around a pre-trained initialization. Foundational work (Arora et al., 2019; Jacot et al., 2018) showed that infinitely wide networks trained with gradient descent follow kernel gradient flow under the Neural Tangent Kernel (NTK), yielding exact functional characterizations of training dynamics. This perspective has since been extended to more realistic settings, including representation learning (Mu et al., 2020), small-data regimes (Arora et al., 2020), and random-matrix studies of generalization (Wei et al., 2022). Building on these insights, several linearized fine-tuning approaches have been proposed to improve efficiency and stability, such as LQF (Achille et al., 2021), privacy-preserving updates (Golatkar et al., 2021), improved task-head initialization (Ren et al., 2023), continual learning (Shon et al., 2022), and language-model adaptation (Malladi et al., 2023). More recent work explores model composition and ensembling through tangent-space operations (Liu & Soatto, 2023; Tang et al., 2024).

The linearized regime has also become central to task arithmetic. Tangent-space representations have been linked to weight disentanglement and reliable task editing (Ortiz-Jimenez et al., 2023; Porrello et al., 2025; Yoshida et al., 2025; Liu et al., 2024). Within this framework, NTK-based approximations enhance task separability and make linear combinations of task vectors more predictable, further underscoring the versatility of model linearization for fine-tuning, composition, and editing.