Efficient Model Editing with Task-localized Sparse Fine-tuning

Leonardo Iurada[1], Marco Ciccone[2], Tatiana Tommasi[1]

[1]Politecnico di Torino, Italy [2]Vector Institute, Toronto, Canada

<u>Poster Session 2</u>

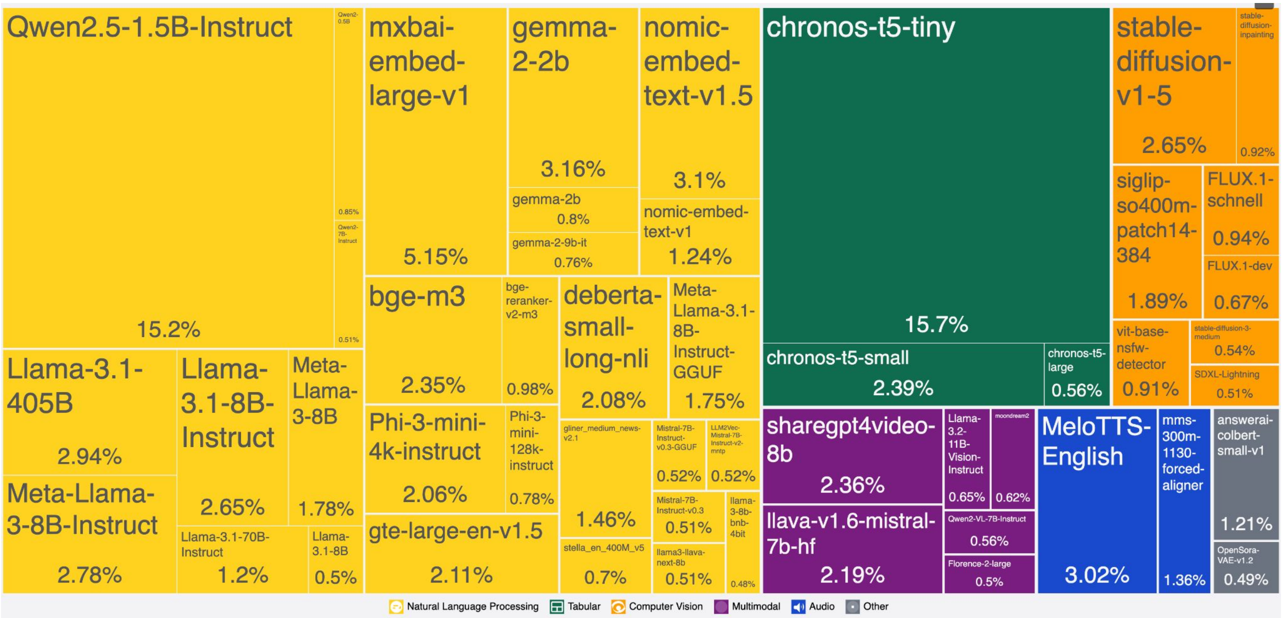
Thursday, April 24th, 2025

3:00pm - 5:30pm

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
FAIR
elsa European Lighthouse on Secure and Safe AI
ellis unit TURIN
Politecnico di Torino
VECTOR INSTITUTE
ICLR International Conference On Learning Representations

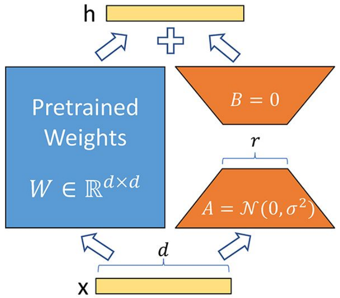
Context: The Democratization of AI

Top Pre-trained Models Downloads from Hugging Face (until 2024)

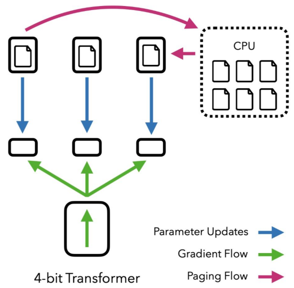


Adapted with:

- PEFT (eg. LoRA, Hu et al., 2022)



- Quantization (eg. QLoRA, Dettmers et al., 2023)

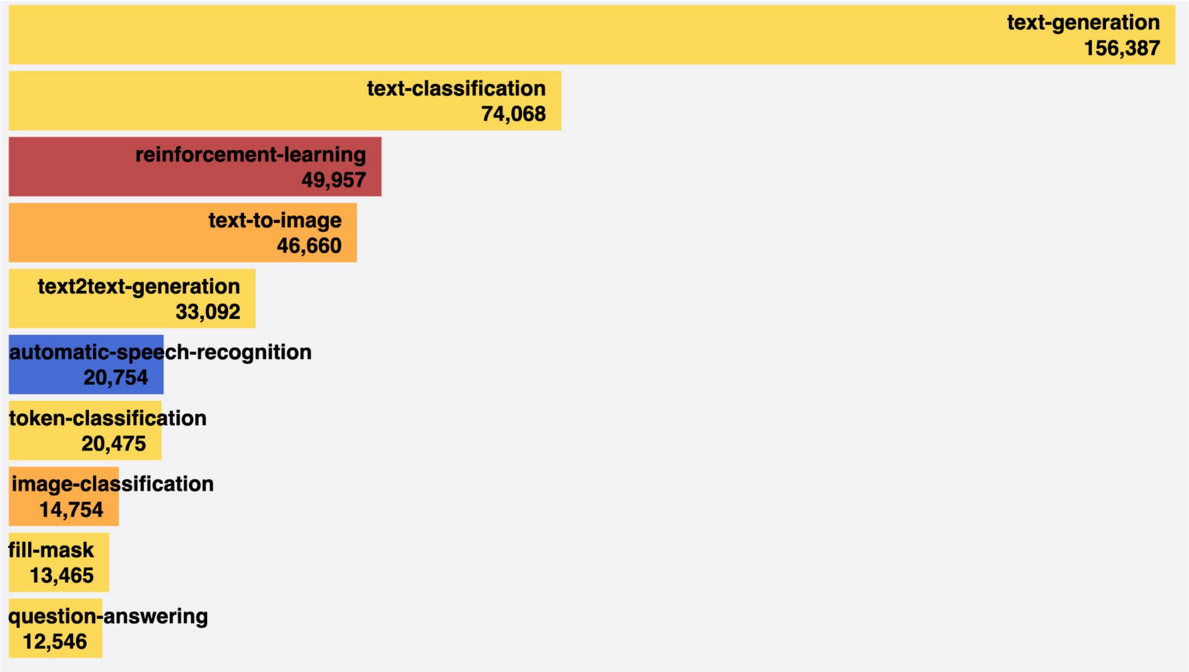


[AI WORLD & Hugging Face, 2024]

Context: The Democratization of AI

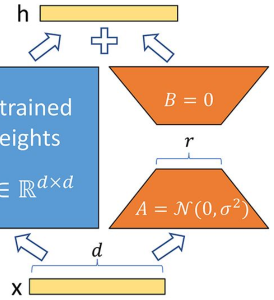
Top Tasks on which Pre-trained Models are fine-tuned and openly shared

Over the last 33 months, more than 1.1M models have been created for many specialized tasks.

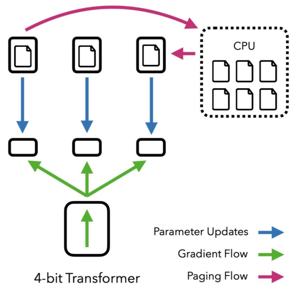


Adapted with:

- PEFT (eg. LoRA, Hu et al., 2022)



- Quantization (eg. QLoRA, Dettmers et al., 2023)

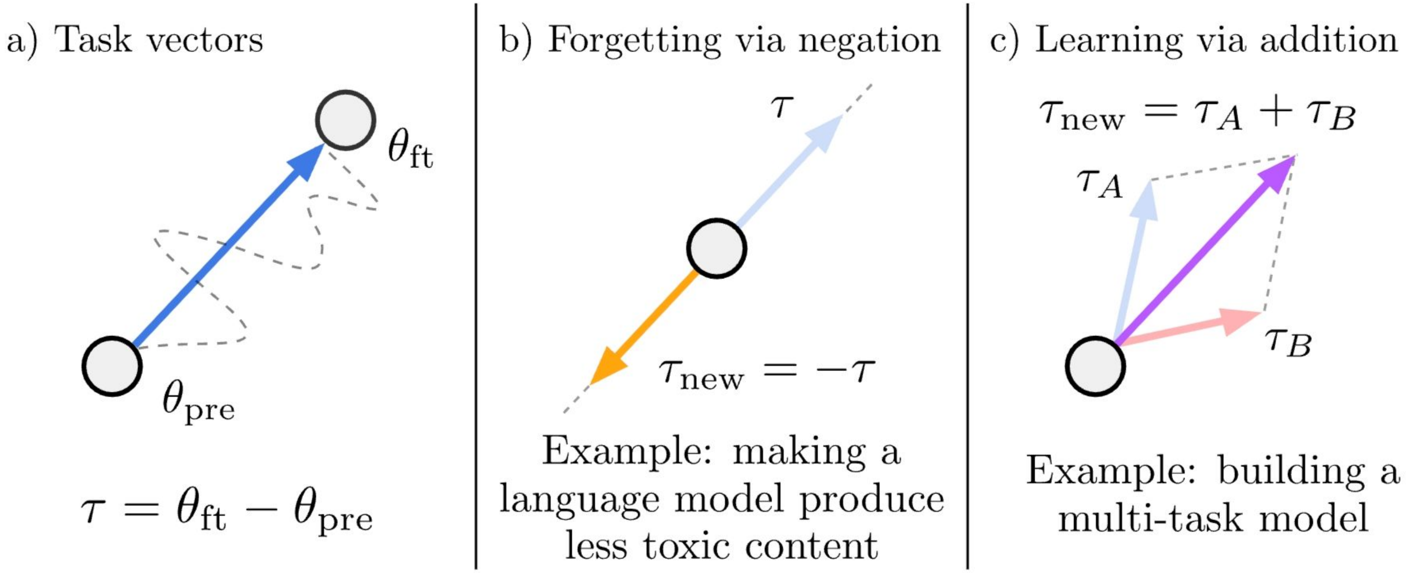


[AI WORLD & Hugging Face, 2024]

Adapting Models via Task Arithmetic

Task arithmetic desiderata:

- **Task-specific knowledge (eg. Math, Coding…) are encoded in “Task Vectors”**
- **Simple arithmetic operations (+, -) with task vectors steer the model’s behavior**

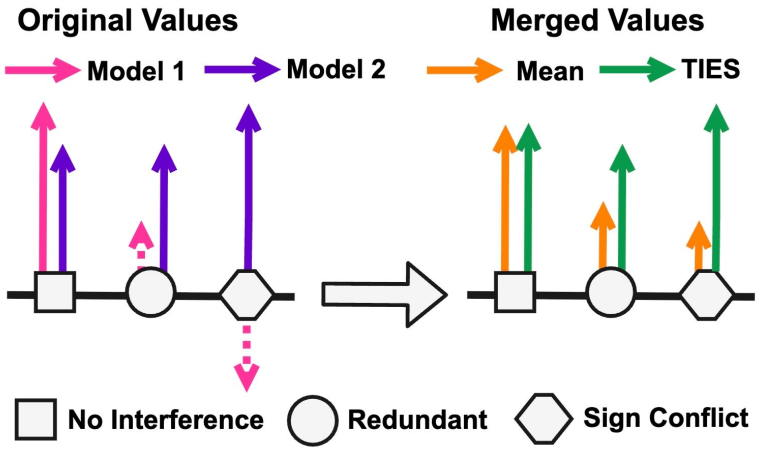


[G.Ilharco et al., 2023]

Adapting Models via Task Arithmetic

CHALLENGE: Collaboration in decentralized settings is hard…

- **Interference** between task vectors when combined ⇒ unintended model behavior!



[P.Yadav et al., 2023]

Adapting Models via Task Arithmetic

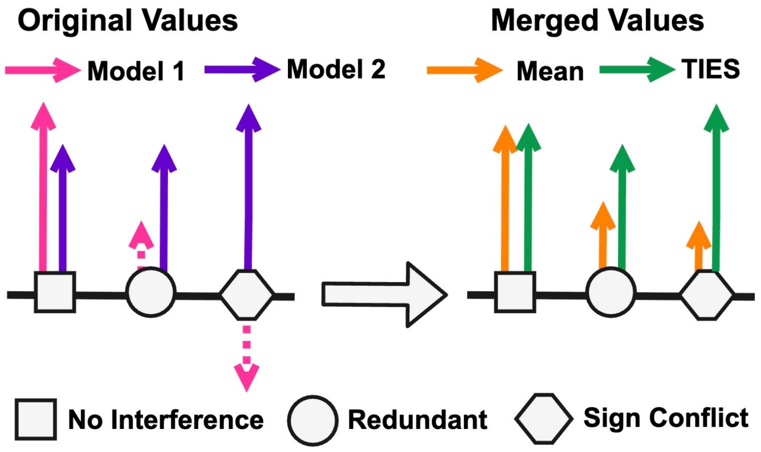
CHALLENGE: Collaboration in decentralized settings is hard…

- **Interference** between task vectors when combined ⇒ unintended model behavior!

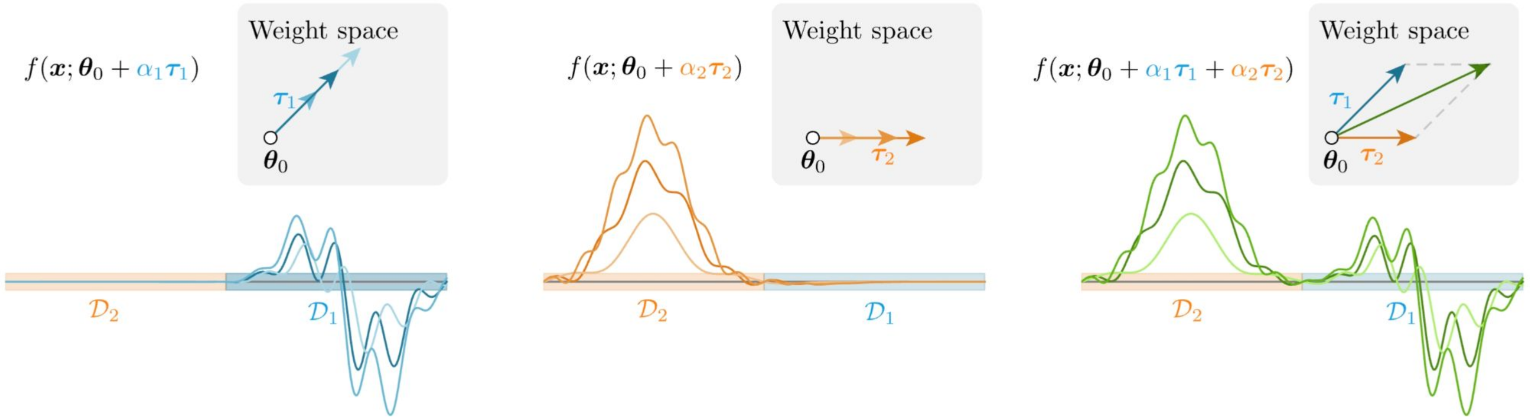
IDEA: when fine-tuning to derive task vectors…

"data from distinct regions in input space affect non-overlapping regions of the activation space"

- **Weight Disentanglement property (⇒ emerges from extensive pre-training)**



[P.Yadav et al., 2023]



[G.Ortiz-Jimenez et al., 2023]

Adapting Models via Task Arithmetic

CHALLENGE: Collaboration in decentralized settings is hard…

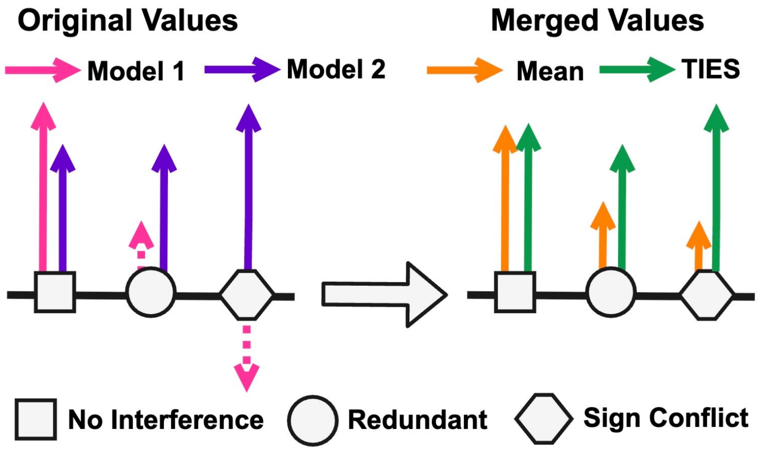
- **Interference** between task vectors when combined ⇒ unintended model behavior!

IDEA: when fine-tuning to derive task vectors…

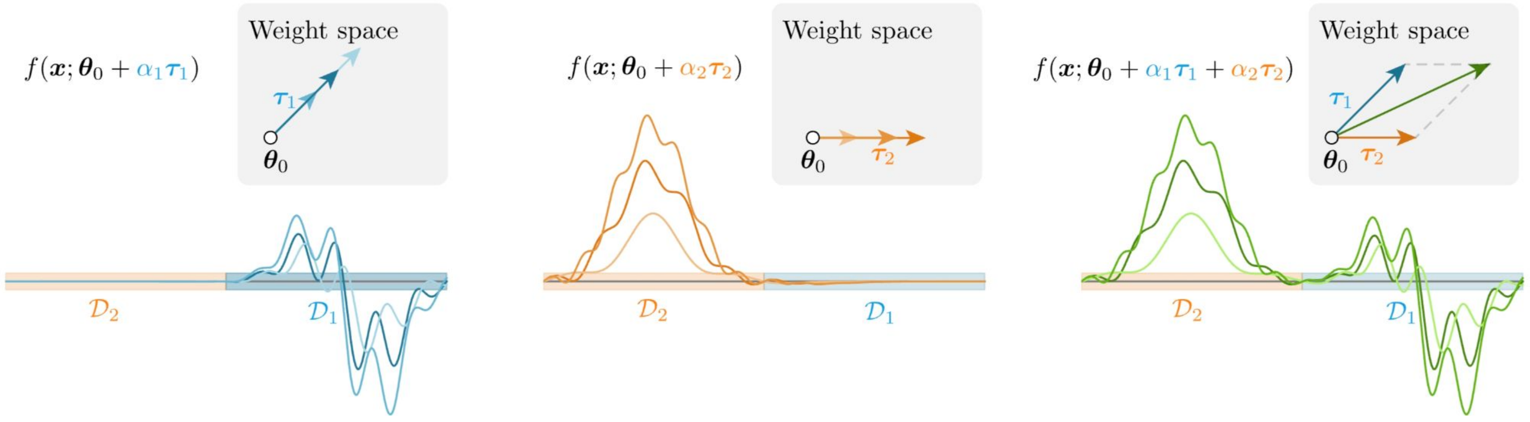
"data from distinct regions in input space affect non-overlapping regions of the activation space"

- **Weight Disentanglement property (⇒ emerges from extensive pre-training)**

QUESTION: How to preserve Weight Disentanglement when deriving task vectors via fine-tuning?



[P.Yadav et al., 2023]



[G.Ortiz-Jimenez et al., 2023]

The Task Arithmetic Framework

Formally, <u>Weight Disentanglement (WD)</u> is defined as:

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0)\mathbb{1}\left(\boldsymbol{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t\right) + \sum_{t=1}^{T} f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)\mathbb{1}(\boldsymbol{x} \in \mathcal{D}_t)$$

$$= g_0(\boldsymbol{x}) + \sum_{t=1}^{T} g_t(\boldsymbol{x}, \alpha_t \boldsymbol{\tau}_t) \ .$$

The Task Arithmetic Framework

Formally, <u>Weight Disentanglement (WD)</u> is defined as:

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0)\mathbb{1}\left(\boldsymbol{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t\right) + \underline{\sum_{t=1}^{T} f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)\mathbb{1}(\boldsymbol{x} \in \mathcal{D}_t)}$$

$$= g_0(\boldsymbol{x}) + \sum_{t=1}^{T} g_t(\boldsymbol{x}, \alpha_t \boldsymbol{\tau}_t) \ .$$

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t), \quad \forall \boldsymbol{x} \in \mathcal{D}_t$$

… meaning that:

- **inference on $\boldsymbol{x} \in \mathcal{D}_t \Rightarrow$ only $f(\cdot, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)$ must activate**

The Task Arithmetic Framework

Formally, <u>Weight Disentanglement (WD)</u> is defined as:

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = \underline{f(\boldsymbol{x}, \boldsymbol{\theta}_0)\mathbb{1}\left(\boldsymbol{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t\right)} + \sum_{t=1}^{T} f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)\mathbb{1}(\boldsymbol{x} \in \mathcal{D}_t)$$

$$= g_0(\boldsymbol{x}) + \sum_{t=1}^{T} g_t(\boldsymbol{x}, \alpha_t \boldsymbol{\tau}_t) \ .$$

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0), \quad \forall \boldsymbol{x} \notin \mathcal{D}$$

… meaning that:

- **inference on $\boldsymbol{x} \in \mathcal{D}_t$ ⇒ only $f(\cdot, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)$ must activate**
- **inference on $\boldsymbol{x} \notin \mathcal{D}$ ⇒ must fall back to base pre-trained behavior**

The Task Arithmetic Framework

Formally, Weight Disentanglement (WD) is defined as:

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0)\mathbb{1}\left(\boldsymbol{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t\right) + \sum_{t=1}^{T} f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)\mathbb{1}(\boldsymbol{x} \in \mathcal{D}_t)$$

$$= g_0(\boldsymbol{x}) + \sum_{t=1}^{T} g_t(\boldsymbol{x}, \alpha_t \boldsymbol{\tau}_t) \, .$$

… meaning that:

- **inference on $\boldsymbol{x} \in \mathcal{D}_t \Rightarrow$ only $f(\cdot, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)$ must activate**
- **inference on $\boldsymbol{x} \notin \mathcal{D} \Rightarrow$ must fall back to base pre-trained behavior**

→ Previous works: "enforce" (hope) WD preservation via explicit network linearization, i.e. fine-tuning:

$$f_{\text{lin}}\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t^{\top} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$$

The Task Arithmetic Framework

Formally, <u>Weight Disentanglement (WD)</u> is defined as:

$$f\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0)\underline{\mathbb{1}\left(\boldsymbol{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t\right)} + \sum_{t=1}^{T} f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)\underline{\mathbb{1}(\boldsymbol{x} \in \mathcal{D}_t)}$$

$$= g_0(\boldsymbol{x}) + \sum_{t=1}^{T} g_t(\boldsymbol{x}, \alpha_t \boldsymbol{\tau}_t)\ .$$

… meaning that:

- **inference on $\boldsymbol{x} \in \mathcal{D}_t$ ⇒ only $f(\cdot, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t)$ must activate**
- **inference on $\boldsymbol{x} \notin \mathcal{D}$ ⇒ must fall back to base pre-trained behavior**

→ <u>Previous works</u>: "enforce" (hope) WD preservation via explicit network linearization, i.e. fine-tuning:

$$f_{\text{lin}}\left(\boldsymbol{x}, \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t\right) = f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \sum_{t=1}^{T} \underline{\alpha_t \boldsymbol{\tau}_t^{\top} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)}$$

Doesn't consider Localization!

Imposing Function Localization

To exactly have WD on linearized networks:

$$f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t) \approx f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \alpha_t {\boldsymbol{\tau}_t}^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$$

… we must ensure that each $\boldsymbol{\tau}_t \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \neq 0$ **only for** $\boldsymbol{x} \in \mathcal{D}_t$

⇒ Formally, (<u>Function Localization Constraints</u>):

$$\forall \boldsymbol{x} \in \mathcal{D}_{t' \neq t}, \;\; {\boldsymbol{\tau}_t}^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) = 0$$

Imposing Function Localization

To exactly have WD on linearized networks:

$$f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t) \approx f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \alpha_t \boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$$

… we must ensure that each $\boldsymbol{\tau}_t \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \neq 0$ **only for** $\boldsymbol{x} \in \mathcal{D}_t$

⇒ Formally, (Function Localization Constraints):

$$\underline{\forall \boldsymbol{x} \in \mathcal{D}_{t' \neq t},}\quad \boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) = 0$$

PROBLEM: We can't access $\mathcal{D}_{t' \neq t}$ **(we only have** $\boldsymbol{x} \in \mathcal{D}_t$**, isolated decentralized setting!)**

Deriving Weight-Disentangled Task Vectors

⇒ **EMPIRICALLY: <u>least sensitive</u> weights ($\nabla_{\boldsymbol{\theta}_{[j]}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \approx 0$) are *least sensitive* for all tasks!**

- **So, for the *least sensitive* weights, the Constraints are** $\underline{\forall \boldsymbol{x} \in \mathcal{D}_t}, \quad \boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$

Deriving Weight-Disentangled Task Vectors

⇒ **EMPIRICALLY: <u>least sensitive</u> weights ($\nabla_{\boldsymbol{\theta}_{[j]}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \approx 0$) are *least sensitive* for all tasks!**

- So, for the *least sensitive* weights, the Constraints are $\forall \boldsymbol{x} \in \mathcal{D}_t, \quad \boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$

⇒ We propose: ***<u>Task-Localized Sparse Fine-tuning (TaLoS)</u>*** → derive task vector $\boldsymbol{\tau}_t$ by:

- Calibrate Gradient Mask $\boldsymbol{c}$ → Update only the *least sensitive* weights

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \gamma[\boldsymbol{c} \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(\boldsymbol{x}, \boldsymbol{\theta}^{(i-1)}), y)] \qquad j = 1, \ldots, m \quad \boldsymbol{c}_{[j]} = \begin{cases} 1 & \text{if } \nabla_{\boldsymbol{\theta}_{[j]}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) < \text{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- Requiring access only to task data $\boldsymbol{x} \in \mathcal{D}_t$ (no information sharing needed)

Deriving Weight-Disentangled Task Vectors

⇒ **EMPIRICALLY: least sensitive weights ($\nabla_{\boldsymbol{\theta}_{[j]}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \approx 0$) are *least sensitive* for all tasks!**

- So, for the *least sensitive* weights, the Constraints are $\forall \boldsymbol{x} \in \mathcal{D}_t, \quad \boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$

⇒ We propose: *Task-Localized Sparse Fine-tuning (TaLoS)* → derive task vector $\boldsymbol{\tau}_t$ by:

- Calibrate Gradient Mask $\boldsymbol{c}$ → Update only the *least sensitive* weights

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \gamma[\boldsymbol{c} \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(\boldsymbol{x}, \boldsymbol{\theta}^{(i-1)}), y)] \qquad j = 1, \ldots, m \quad \boldsymbol{c}_{[j]} = \begin{cases} 1 & \text{if } \nabla_{\boldsymbol{\theta}_{[j]}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) < \text{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- Requiring access only to task data $\boldsymbol{x} \in \mathcal{D}_t$ (no information sharing needed)

⇒ TaLoS promotes *Weight Disentanglement* during fine-tuning

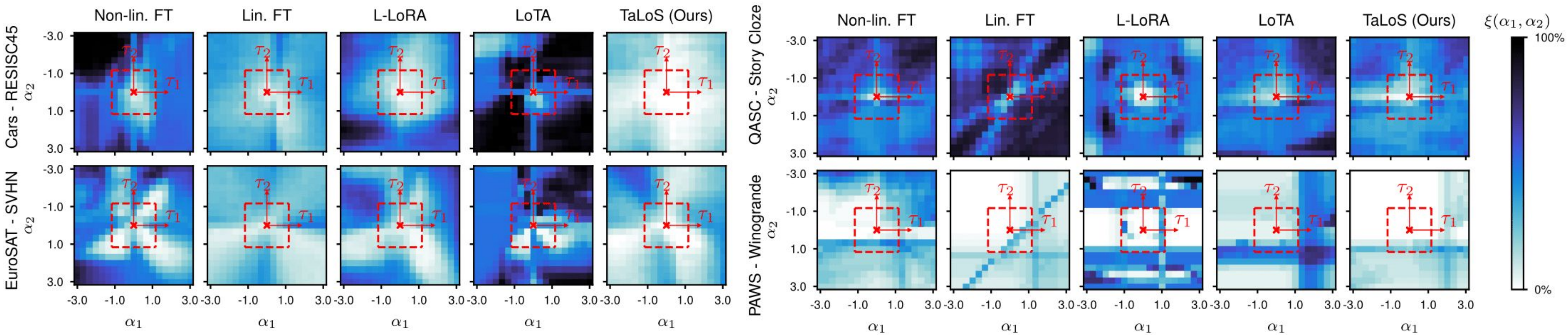
- As it minimally increases the Constraints' dot product $\boldsymbol{\tau}_t^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$ $(\forall \boldsymbol{x} \in \mathcal{D}_t)$

Assessing Weight Disentanglement

Plotting the Weight Disentanglement Error [G.Ortiz-Jimenez, 2023]:

$$\xi(\alpha_1, \alpha_2) = \sum_{t=1}^{2} \mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_t} [\text{dist}(f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1), f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1 + \alpha_2 \boldsymbol{\tau}_2))]$$

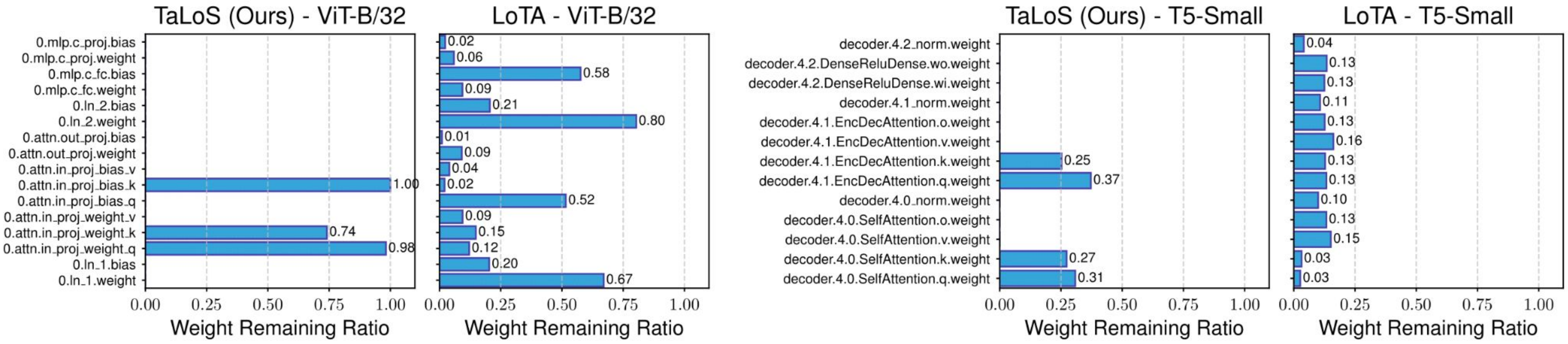
$$\text{dist}(y_1, y_2) = \mathbb{1}(y_1 \neq y_2)$$



Which Weights does TaLoS Select?

Looking at gradient mask c of one transformer block at 90% sparsity:

- **Very structured pattern ⇒ can freeze most of the parameters (<u>high efficiency gains!</u>)**
- **NOTE: this pattern is repeated in all blocks**
    - **<u>So, the "special" weights are in Q, K projections of multi-head self-attention layers</u>**



Task Arithmetic Experiments

Efficiency vs. Task Arithmetic Results

TaLoS improves on Task Addition & Negation while being the most efficient fine-tuning strategy

Method	Effective Cost of Fine-tuning				Task Addition		Task Negation	
	Forward-Backward Pass Time (s)	Optim. Step Time (s)	Tot. Iteration Time (s)	Peak Memory Usage (GiB)	Abs. (↑)	Norm. (↑)	Targ. (↓)	Cont. (↑)
ViT-B/32								
Non-linear FT (Ilharco et al., 2023)	0.3608 ± 0.0036	0.0114 ± 0.0010	0.3722 ± 0.0037	6.5	71.25	76.94	24.04	60.36
Linearized FT (Ortiz-Jimenez et al., 2023)	0.6858 ± 0.0042	0.0103 ± 0.0020	0.6961 ± 0.0047	10.2	76.70	85.86	11.20	60.74
L-LoRA (Tang et al., 2024)	0.3270 ± 0.0076	**0.0036** ± 0.0032	0.3306 ± 0.0082	5.3	78.00	86.08	17.29	60.75
LoTA (Panda et al., 2024)	0.3289 ± 0.0041	0.1269 ± 0.0050	0.4558 ± 0.0065	6.8	64.94	74.37	21.09	**61.01**
TaLoS (Ours)	**0.1256** ± 0.0045	0.0388 ± 0.0040	**0.1644** ± 0.0060	**4.7**	**79.67**	**90.73**	**11.03**	60.69
ViT-L/14								
Non-linear FT (Ilharco et al., 2023)	1.2174 ± 0.0097	0.0156 ± 0.0055	1.2330 ± 0.0112	18.6	86.09	90.14	20.61	72.72
Linearized FT (Ortiz-Jimenez et al., 2023)	1.6200 ± 0.0067	0.0262 ± 0.0082	1.6462 ± 0.0106	21.3	88.29	93.01	10.86	72.43
L-LoRA (Tang et al., 2024)	0.5153 ± 0.0077	**0.0082** ± 0.0015	0.5235 ± 0.0078	9.7	87.77	91.87	19.39	73.14
LoTA (Panda et al., 2024)	0.8438 ± 0.0052	0.4449 ± 0.0074	1.2887 ± 0.0090	15.4	87.66	91.69	22.11	73.21
TaLoS (Ours)	**0.1891** ± 0.0039	0.1372 ± 0.0036	**0.3263** ± 0.0053	**7.8**	**88.37**	**95.20**	**10.68**	**73.63**
T5-Large								
Non-linear FT (Ilharco et al., 2023)	0.9047 ± 0.0068	0.0894 ± 0.0034	0.9941 ± 0.0076	30.0	75.37	85.25	41.54	45.49
Linearized FT (Ortiz-Jimenez et al., 2023)	1.7683 ± 0.0084	0.1170 ± 0.0060	1.8853 ± 0.0103	35.1	69.38	78.95	41.37	**45.70**
L-LoRA (Tang et al., 2024)	0.7452 ± 0.0084	**0.0136** ± 0.0029	0.7588 ± 0.0089	18.2	72.10	87.78	48.37	45.51
LoTA (Panda et al., 2024)	0.8526 ± 0.0043	0.3842 ± 0.0019	1.2368 ± 0.0047	32.1	75.84	88.14	44.33	45.47
TaLoS (Ours)	**0.4358** ± 0.0075	0.0509 ± 0.0046	**0.4867** ± 0.0088	**12.1**	**79.07**	**90.61**	**37.20**	**45.70**

Conclusions & Next Steps

- We advanced the field of task arithmetic by deriving **a novel set of function localization constraints** that provide **exact guarantees of weight disentanglement** on linearized networks.

- We empirically observed that **the least sensitive parameters** in transformer-based architectures pre-trained on large-scale datasets **can be consistently identified regardless of the task**. We exploit this regularity to **satisfy the localization constraints** under strict **individual training** assumptions.

- We introduced **Task-Localized Sparse Fine-Tuning (TaLoS)** that enables task arithmetic by jointly implementing the **localization constraints** and inducing a **linear regime** during fine-tuning, without incurring in the overheads of explicit network linearization.

Next Steps:

- **TaLoS** works because **we trust the regularities of the pre-trained model**
  ⇒ What if we **explicitly impose the localization constraint during fine-tuning?**

Efficient Model Editing with Task-localized Sparse Fine-tuning

Leonardo Iurada[1], Marco Ciccone[2], Tatiana Tommasi[1]

[1]Politecnico di Torino, Italy [2]Vector Institute, Toronto, Canada

- ⁉️ **When: Poster Session 2** - Thursday, April 24th, 2025 (3:00pm - 5:30pm)
- 📚 **Read our Paper:** https://openreview.net/forum?id=TDyE2iuvyc
- 💻 **Code & Project Page:** https://github.com/iurada/talos-task-arithmetic
- ✉️ Correspondence to: leonardo.iurada@polito.it

Finanziato dall'Unione europea NextGenerationEU · Ministero dell'Università e della Ricerca · Italiadomani · FAIR · elsa · ellis unit TURIN · Politecnico di Torino · Vector Institute · ICLR