Evaluating Graph-Regularized Methods In Data-Free Model Merging

by

Aditya Paikrao

May 21, 2026

A project report submitted to the
Faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfillment of the requirements for the
degree of

Master of Computer Science

Department of Computer Science and Engineering

To my family and friends

Acknowledgments

I would like to express my sincere gratitude to my advisor, Professor Kaiyi Ji, for his continued support, guidance, and encouragement throughout this work.

I would also like to thank my parents and my sister for their constant support and belief in me. This would not have been possible without them.

Lastly, I would also like to thank my friends for their support and for all the encouragement they have given me throughout my entire master's journey.

Table of Contents

List of Tables

List of Figures

Abstract

Deploying a separate fine-tuned language model for every task can increase checkpoint storage, serving complexity, and inference cost. In many practical settings, the original task data may also be unavailable because of privacy, licensing, or data-retention constraints. Data-free model merging addresses this setting by combining task-specific checkpoints into a single multitask model using only model parameters, without returning to the original training data or performing additional fine-tuning.

This project studies data-free merging through the geometry of task vectors, where a task vector is the parameter update from a shared pretrained RoBERTa-Base model to a task-specific expert. The central difficulty is task interference: updates that preserve one task may conflict with updates needed for another. The project evaluates this problem on five GLUE tasks and compares weight averaging, Task Arithmetic, TIES-Merging, WUDI, WUDI with Subspace Boosting, and two graph-based extensions.

The graph-based methods test whether relationships between tasks can provide useful information during merging. Graph Laplacian merging represents task vectors in a low-rank subspace, a compact coordinate system for the main task-update directions, and smooths scalar merge weights over a task-similarity graph. Graph-Regularized Adaptive WUDI keeps WUDI's free merged vector while learning graph-regularized weights on normalized task constraints that compare tasks after accounting for task-vector scale. In the recorded experiments, WUDI and WUDI with Subspace Boosting reach about 88.7–88.8% average normalized score, while the best Graph Laplacian merge reaches 63.48%. Across the tested setting, the results support a conservative conclusion: graph structure is useful for diagnosing and regularizing interference, but performance is strongest when WUDI's task-preservation constraints remain close to uniform.

Chapter1

Introduction

Modern neural language models are often fine-tuned as task-specific experts, each optimized for a target distribution: the inputs, labels, and objective it is expected to handle at deployment time. Maintaining one checkpoint per task is effective but costly: it increases storage requirements, complicates model serving and routing, and can raise inference cost when multiple experts must be loaded or evaluated. In addition, the data used to train the experts may not be available later because of privacy, licensing, ownership, or retention constraints. These deployment pressures motivate data-free model merging, which aims to combine several fine-tuned checkpoints into one multitask model without reusing the original training data or performing additional fine-tuning.

Figure 1.1 summarizes this setting. A shared pretrained model is fine-tuned into several task experts, and model merging combines those experts into a single checkpoint that should preserve their task abilities without accessing the original task datasets.

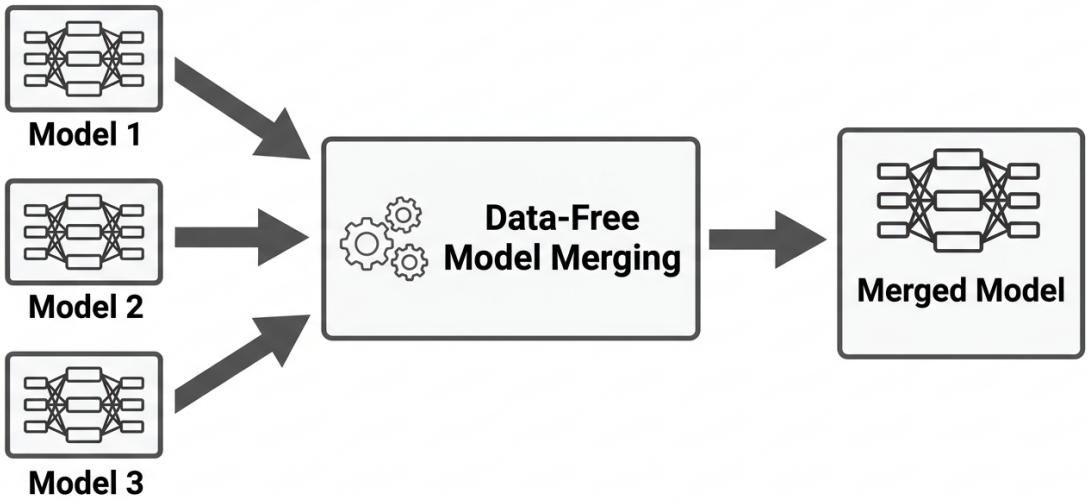

Figure 1.1: Data-free model merging. Multiple task-specific experts are derived from the same pretrained model, then merged into one multitask model using only the trained checkpoints rather than the original training data.

This project studies data-free merging for RoBERTa-Base experts fine-tuned on five GLUE tasks: CoLA, SST-2, MNLI, QQP, and STS-B [1, 2]. The central challenge is task interference. Each expert can be represented by a task vector, the parameter difference between the fine-tuned model and the shared pretrained initialization. If task vectors encode compatible changes, then a simple combined update may preserve several tasks at once. If they encode conflicting changes, however, merging can erase task-specific behavior, overemphasize high-resource or geometrically dominant tasks, and weaken tasks whose useful updates are smaller or more specialized.

CoLA and STS-B are especially informative stress tests for this setting. CoLA is a small grammatical-acceptability task, so its useful update can be fragile when merged with larger sentence-pair or sentiment tasks. STS-B is a semantic similarity regression task rather than a categorical classification task, so its objective and prediction geometry differ from most of the other selected GLUE tasks. These differences make both tasks vulnerable to interference and make worst-task normalized performance an important complement to average performance.

The methods in this report can be organized by how they construct or constrain the

merged update. Weight averaging and Task Arithmetic are linear baselines that directly combine model parameters or task vectors [3, 4]. TIES-Merging reduces sign-level conflicts before combining task vectors [5]. WUDI-Merging instead optimizes a free merged vector, meaning an update that is not restricted to a scalar-weighted sum of the input task vectors, while using normalized task constraints that scale each task's preservation penalty by the size of that task's vector [6]. WUDI with Subspace Boosting adds a low-rank boosting step to this baseline [7]. The two graph-based methods tested here ask whether task relationships can improve either scalar merge weights or WUDI's task-constraint weights.

The graph motivation is that task vectors are not necessarily independent objects. If two tasks have similar update directions, preserving one may also help preserve the other; if two tasks differ strongly, forcing them to share the same merge behavior may increase interference. A task-similarity graph gives a compact representation of these relationships. Graph Laplacian merging builds such a graph in a low-rank task-vector subspace, a small coordinate system capturing the main directions spanned by the task vectors, and then smooths merge weights so that connected tasks receive more compatible treatment. Graph-Regularized Adaptive WUDI keeps WUDI's free merged vector but learns task-constraint weights regularized by a graph prior, so that task relationships influence the balance of constraints without forcing the final update to be a weighted average of the experts.

The report develops the evidence in three phases. First, it establishes the interference problem by evaluating naive weight averaging and Task Arithmetic. Second, it compares stronger interference-aware baselines, including TIES-Merging, WUDI, and WUDI with Subspace Boosting. Third, it evaluates Graph Laplacian merging and Graph-Regularized Adaptive WUDI. The quantitative claims are substantiated in Chapter 5: baseline comparisons are reported in Table 5.1 and Figure 5.2, while the graph-based and adaptive results are reported in Table 5.2, Table 5.3, and Figure 5.5.

At a high level, the results support a conservative interpretation. Graph structure helps analyze task relationships and can regularize harmful movement in task weights, but

the tested graph-based variants do not replace the balanced task preservation provided by WUDI's normalized constraints. The detailed numerical comparison and interpretation are left to the Results and Discussion chapters, where this conclusion is examined through normalized GLUE scores and worst-task behavior.

1.1 Contributions

This project makes four main contributions:

1. It motivates data-free model merging as a practical deployment problem involving storage, serving cost, and restricted access to original training data.

2. It quantifies task interference among five RoBERTa–GLUE task experts using normalized average and worst-task scores.

3. It positions WUDI and WUDI with Subspace Boosting against weight averaging, Task Arithmetic, TIES-Merging, and Graph Laplacian merging under a common evaluation protocol.

4. It evaluates Graph-Regularized Adaptive WUDI and analyzes when graph-regularized task-constraint weighting preserves or weakens WUDI's balanced constraints.

1.2 Report Organization

Chapter 2 introduces task vectors, normalized evaluation, and model-merging baselines. Chapter 3 describes Graph Laplacian merging and Graph-Regularized Adaptive WUDI. Chapter 4 summarizes the experimental setup. Chapter 5 presents the main quantitative results and figures. Chapter 6 interprets the findings and limitations. Chapter 7 concludes with recommendations for future work.

Chapter2

Background

2.1 Task Vectors

Let θ_0 denote the pretrained RoBERTa-Base parameters and let θ_i denote the model fine-tuned on task i. The task vector for task i is

$$\tau_i = \theta_i - \theta_0. \tag{2.1}$$

The task vector isolates the update induced by fine-tuning. A merged model is usually built by combining task vectors and applying the result to the base model:

$$\theta_{\text{merge}} = \theta_0 + \tau_{\text{merge}}. \tag{2.2}$$

Because task heads differ across the selected GLUE tasks, the classifier parameters and biases are excluded in the merging experiments.

2.2 Naive Merging Baselines

The simplest baseline is weight averaging, which averages model parameters directly [3]. Although easy to implement, direct averaging can mix incompatible updates and produce severe task interference.

Task Arithmetic instead averages task vectors and applies a scaling factor γ [4]:

$$\theta_{\text{merge}} = \theta_0 + \gamma \cdot \frac{1}{T} \sum_{i=1}^{T} \tau_i. \tag{2.3}$$

The scaling coefficient controls the update magnitude. If γ is too small, useful task-specific information is under-applied. If it is too large, conflicting directions can dominate.

2.3 Interference-Aware Baselines

TIES-Merging reduces interference by trimming small-magnitude deltas, resolving sign conflicts, and merging only the elected signs [5]. It directly addresses the observation that task vectors can disagree in sign across parameters.

WUDI-Merging takes a different approach. Instead of simply averaging task vectors, WUDI optimizes a merged vector that satisfies normalized task constraints and reduces interference in the task-vector geometry [6]. A key property is that WUDI keeps the merged vector free: it is not restricted to a convex combination or scalar-weighted sum of the input task vectors.

WUDI with Subspace Boosting adds an additional low-rank denoising or boosting step [7]. In the recorded experiments, WUDI and WUDI with Subspace Boosting are very close, with Subspace Boosting giving the strongest average normalized score by a small margin.

2.4 Normalized Evaluation

The report emphasizes normalized score rather than raw average metric, following model-merging evaluations that report performance relative to the corresponding task expert [8]. GLUE tasks use different metrics and scales, so raw averages can be misleading. For each task, the merged model score is divided by the corresponding expert score:

$$s_i^{\text{norm}} = \frac{s_i^{\text{merge}}}{s_i^{\text{expert}}}. \tag{2.4}$$

The main aggregate metric is the mean normalized score:

$$\bar{s}^{\text{norm}} = \frac{1}{T} \sum_{i=1}^{T} s_i^{\text{norm}}. \tag{2.5}$$

The Results chapter reports normalized scores as percentages. The worst normalized score is also reported because a high average can hide catastrophic degradation on a sensitive task such as CoLA.

Chapter3

Graph-Based Merging Methods

3.1 Graph Laplacian Merging

The graph Laplacian method treats task vectors as graph signals. It first projects task vectors into a low-rank subspace, builds a task-similarity graph, and then smooths task merge weights over that graph using Laplacian regularization [9]. This project uses the method as a data-free merge heuristic rather than as a validation-tuned graph-learning procedure.

Algorithm 1 gives the full procedure used in the experiments. The main implementation detail is that the low-rank task subspace is computed from the small task-by-task Gram matrix rather than by materializing a large parameter-by-task matrix.

Algorithm 1 Laplacian-Regularized Task Vector Merging

Require: Base parameters θ_0, task-specific parameters $\{\theta_i \mid i = 1, \ldots, T\}$, subspace rank r, graph degree q, regularization strength λ, initial task weights $a^{(0)}$

Ensure: Merged parameters θ_{merge}

Construct task vectors
for $i = 1$ to T **do**
  $\tau_i \leftarrow \theta_i - \theta_0$
end for
$M \leftarrow [\tau_1, \tau_2, \ldots, \tau_T] \in \mathbb{R}^{P \times T}$
Compute low-dimensional task subspace
$G \leftarrow M^T M \in \mathbb{R}^{T \times T}$
Compute $G = V \Gamma V^T$
Select the top-r eigenvectors V_r and eigenvalues $\gamma_1, \ldots, \gamma_r$
$\sigma_j \leftarrow \sqrt{\gamma_j}$ for $j = 1, \ldots, r$
$\Sigma_r \leftarrow \text{diag}(\sigma_1, \ldots, \sigma_r)$
$U_r \leftarrow M V_r \Sigma_r^{-1}$
Project task vectors into the subspace
$C \leftarrow U_r^T M$
for $i = 1$ to T **do**
  $c_i \leftarrow C[:, i]$
end for
Build task-similarity graph
for $i = 1$ to T **do**
  for $j = 1$ to T **do**
    $W_{ij} \leftarrow \dfrac{c_i \cdot c_j}{\|c_i\| \|c_j\|}$
  end for
end for
Sparsify W by keeping the q nearest neighbors for each task
Symmetrize W
$D_{ii} \leftarrow \sum_j W_{ij}$ for $i = 1, \ldots, T$
$L \leftarrow D - W$
Solve Laplacian-regularized weight optimization
$a^* \leftarrow \arg\min_a \|a - a^{(0)}\|_2^2 + \lambda a^T L a$
Equivalently solve $(I + \lambda L) a^* = a^{(0)}$
Merge task coordinates and project back
$c_{\text{merge}} \leftarrow \sum_{i=1}^{T} a_i^* c_i$
$\tau_{\text{merge}} \leftarrow U_r c_{\text{merge}}$
$\theta_{\text{merge}} \leftarrow \theta_0 + \tau_{\text{merge}}$
return θ_{merge}

3.2 Graph-Regularized Adaptive WUDI

The second graph-based extension keeps WUDI's free merged vector but learns per-task constraint weights inside the WUDI objective. This is important because preweighting task vectors before WUDI would change the geometry seen by WUDI, and constraining the merged vector to a weighted sum would remove one of WUDI's main advantages.

For a layer or parameter block l, define each task's normalized WUDI constraint as

$$e_i(\tau_{m,l}) = \frac{\|(\tau_{m,l} - \tau_{i,l})\tau_{i,l}^T\|_F^2}{\|\tau_{i,l}\|_F^2}. \tag{3.1}$$

The adaptive method learns a positive mean-one task-constraint vector. To distinguish these constraint weights from Task Arithmetic scaling and Laplacian merge weights, this chapter denotes them by ω. The softmax parameterization is

$$\omega_l = T \cdot \mathrm{softmax}(\beta_l), \tag{3.2}$$

where T is the number of tasks. The graph-regularized adaptive WUDI objective is

$$\mathcal{L}_l(\tau_{m,l}, \beta_l) = \sum_i \omega_{i,l} e_i(\tau_{m,l}) + \lambda \omega_l^T L_{\mathrm{graph},l} \omega_l + \rho \|\omega_l - \mathbf{1}\|_2^2. \tag{3.3}$$

Here ω_l is not a merge coefficient. It controls the importance of each task's WUDI constraint while $\tau_{m,l}$ remains a free optimized variable. Original WUDI is recovered when $\omega_l = \mathbf{1}$.

3.3 Graph Sources for Adaptive WUDI

The adaptive experiments tested several graph sources:

- **Layer-coordinate cosine**: build a graph from low-rank coordinates for each layer.

- **Global cosine**: accumulate a shared task Gram matrix across filtered parameters.
- **Averaged-layer cosine**: average dense cosine graphs computed layer by layer.
- **Subspace alignment ratio (SAR)**: compare task update subspaces using memory-safe per-tensor SVDs [10].

The SAR graph was included because raw vector angles may not capture subspace compatibility. For each two-dimensional parameter tensor, the method computes truncated left and right singular subspaces and measures overlap between tasks. These overlaps are averaged into a shared graph without forming a full parameter-by-task matrix.

3.4 Bounded Constraint-Weight Variant

Because free softmax constraint weights can move too far from uniform, a bounded trust-region parameterization was also tested:

$$\tilde{\omega}_i = 1 + \epsilon \tanh(\beta_i), \tag{3.4}$$

$$\omega_i = T \frac{\tilde{\omega}_i}{\sum_j \tilde{\omega}_j}. \tag{3.5}$$

This ensures positive weights with mean one and keeps WUDI at the center of the feasible region. The parameter ϵ directly controls the allowed movement away from uniform WUDI constraints.

Chapter4

Experimental Setup

4.1 Tasks and Metrics

The experiments use five RoBERTa-Base GLUE experts. Table 4.1 summarizes the task set.

Table 4.1: Selected RoBERTa–GLUE tasks.

Task	Type	Metric	Labels	Role in the experiment
CoLA	Grammaticality	MCC	2	Sensitive low-data task and common interference victim.
SST-2	Sentiment	Accuracy	2	Clean single-sentence classification baseline.
MNLI	Natural language inference	Accuracy	3	Large three-way sentence-pair task.
QQP	Paraphrase detection	F1/Acc.	2	Large sentence-pair task with different features.
STS-B	Semantic similarity	Pearson/Spearman	Reg.	Regression task with a structurally different head.

The selected tasks intentionally combine single-sentence classification, sentence-pair classification, natural language inference, paraphrase detection, and regression. This diversity makes the merge difficult and exposes interference that may not appear in homogeneous task groups.

4.2 Experiment Phases

The project was organized into three main phases.

1. **Step 1: Interference measurement.** Individual experts were evaluated, then weight averaging and Task Arithmetic were tested. Task Arithmetic scaling was swept to identify the best naive task-vector baseline.

2. **Step 2: Interference-aware baselines.** TIES, WUDI, and WUDI with Subspace Boosting were evaluated using fixed settings from the project run scripts.

3. **Step 3: Graph methods.** Graph Laplacian merging was tested with a λ sweep. Graph-Regularized Adaptive WUDI was tested with softmax constraint weights, multiple graph sources, SAR graphs, and bounded weights.

4.3 Fixed Baseline Settings

The Step 2 baseline settings were:

- Task Arithmetic: scaling $\gamma = 0.8$ for the fixed comparison row, with the best Step 1 sweep value also reported separately.
- TIES: mask rate 0.9 and scaling 0.7.
- WUDI: effective scaling 1 and ℓ_1 coefficient 0.
- WUDI with Subspace Boosting: effective scaling 1, ℓ_1 coefficient 0.

4.4 Graph Laplacian Settings

The graph Laplacian implementation uses a memory-safe Gram-matrix construction. Rather than materializing the full dense matrix of flattened task vectors, it computes pairwise dot products to form a 5×5 Gram matrix. The main recorded setting uses rank 5 and a kNN task graph. λ values include 0.0, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, and 50.0.

4.5 Adaptive WUDI Settings

The main adaptive WUDI experiments use five source tasks, WUDI iterations set to 300, and a learning rate of 2×10^{-5}. Diagnostic adaptive runs use constraint-weight learning rate

0.01.

For softmax constraint weights, the main λ sweep uses $\rho = 0.01$ and the layer-coordinate graph source. Additional graph-source experiments compare global cosine, averaged-layer cosine, and SAR. The bounded-weight experiments use a bounded-tanh parameterization with ϵ values 0.01, 0.02, and 0.05.

Chapter5

Results

5.1 Step 1: Naive Merging Shows Strong Interference

The first analysis confirms that naive merging substantially degrades task performance. Weight averaging reaches only 36.70% average normalized score, with a worst-task normalized score of 6.03%. The best Task Arithmetic run uses scaling $\gamma = 0.7$ and improves the average normalized score to 62.51%, but its worst-task score remains only 12.12%. These results show that task-vector scaling helps, but does not solve cross-task interference.

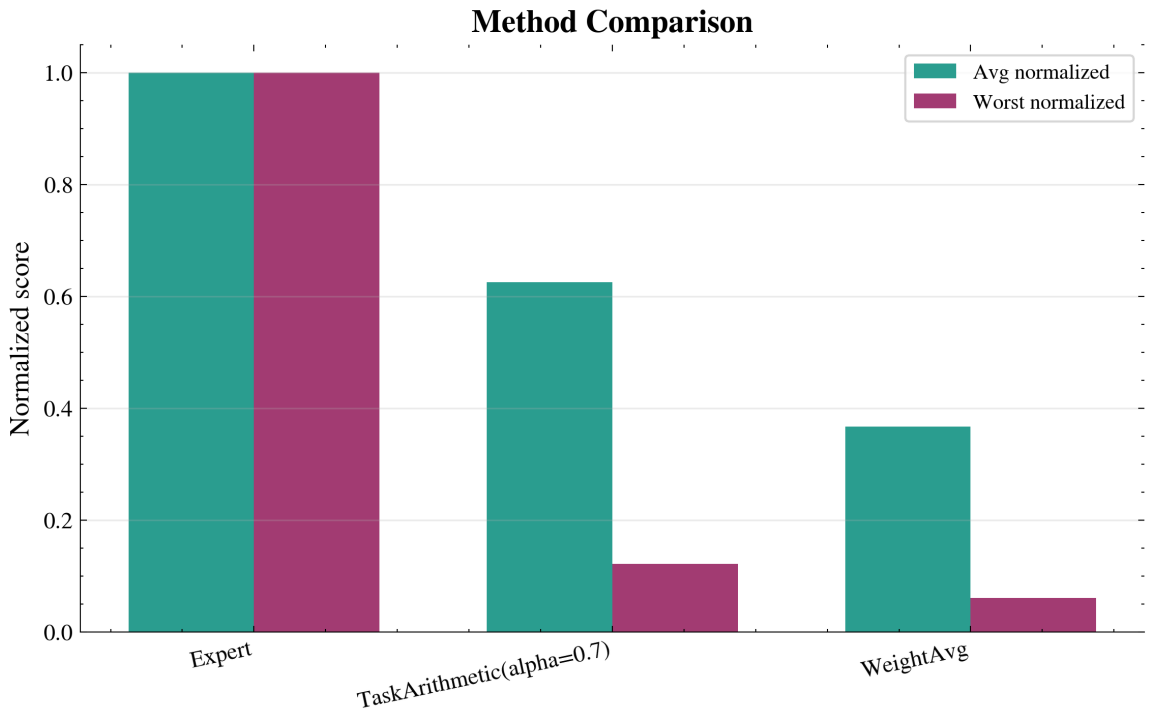

Figure 5.1: Step 1 comparison of naive merging methods. Weight averaging is highly unstable, while Task Arithmetic improves average normalized performance but leaves severe worst-task degradation.

5.2 Step 2: WUDI Is the Strongest Baseline

Table 5.1 summarizes the main fixed-baseline results. TIES under the selected setting performs worse than Task Arithmetic on average and has a negative worst normalized score, indicating severe degradation on at least one task. WUDI and WUDI with Subspace Boosting are far stronger than the other baselines.

Table 5.1: Aggregate normalized baseline results. Scores are percentages relative to the corresponding task experts.

Method	Avg. normalized score	Worst normalized score
Weight averaging	36.70	6.03
Task Arithmetic, best $\gamma = 0.7$	62.51	12.12
Task Arithmetic, fixed $\gamma = 0.8$	59.62	3.34
TIES, mask = 0.9, scale = 0.7	42.80	-34.63
WUDI	88.67	65.93
WUDI + Subspace Boosting	88.83	66.28

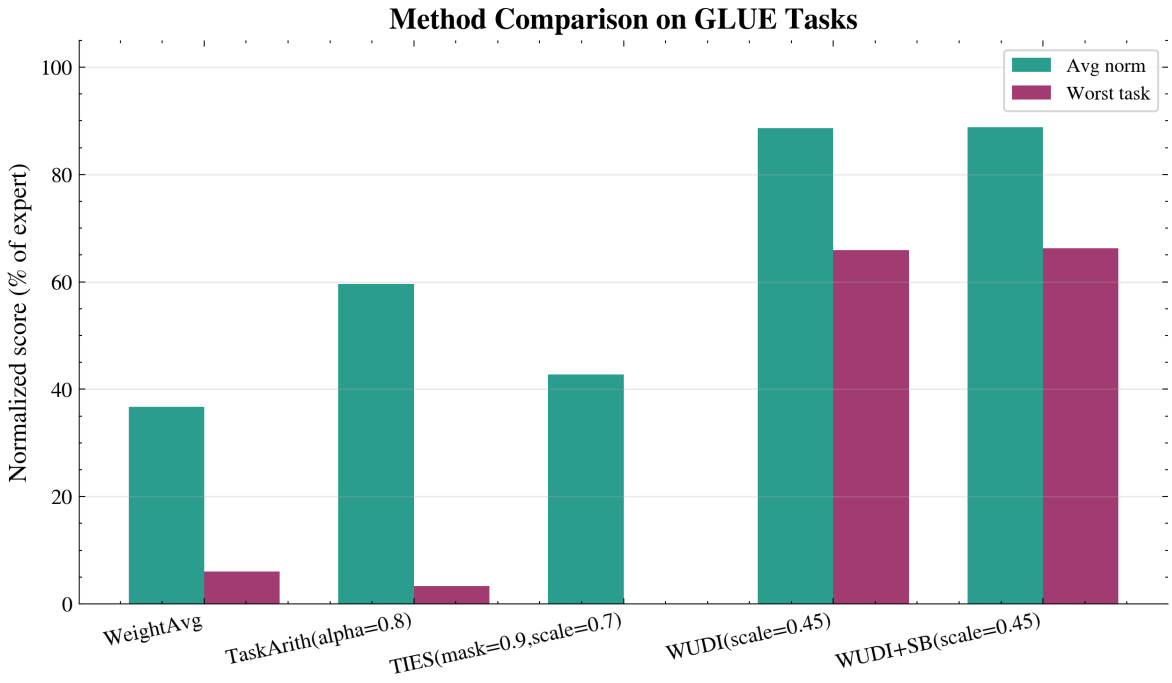

Figure 5.2: Step 2 method comparison. WUDI and WUDI with Subspace Boosting dominate the weaker baselines under normalized GLUE evaluation.

The main conclusion from Step 2 is that WUDI's free optimized merged vector and normalized constraints are highly effective for this five-task setup. Subspace Boosting provides only a small improvement over WUDI, suggesting that the primary benefit comes from the WUDI objective itself rather than the boosting extension.

5.3 Step 3: Graph Laplacian Merging

Graph Laplacian merging improves over weight averaging and some weak baselines, but it remains far below WUDI. The best recorded Laplacian merge uses $\lambda = 10$ and reaches 63.48% average normalized score with 8.22% worst normalized score. This is close to the best Task Arithmetic average but still has poor worst-task preservation.

Table 5.2: Best graph Laplacian result.

Method	Best λ	Avg. normalized score	Worst normalized score
Graph Laplacian merging	10	63.48	8.22

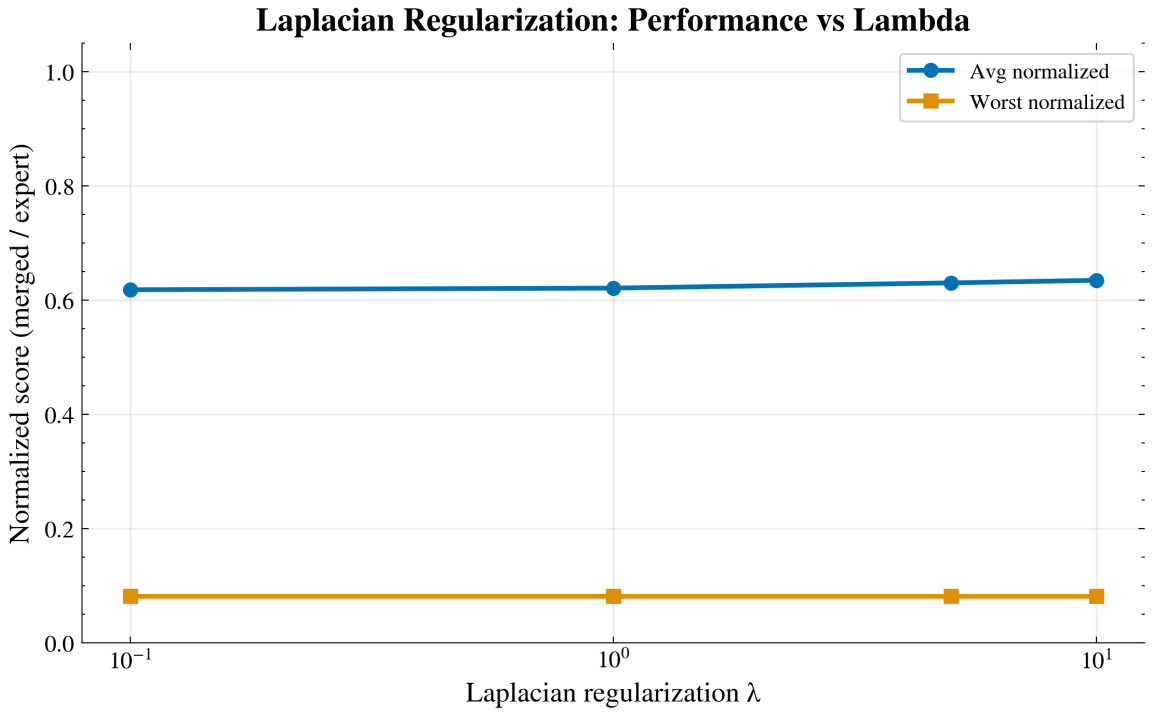

Figure 5.3: Graph Laplacian λ sweep. The curve identifies a controllable smoothing effect and a best recorded λ of 10, but the resulting score is still much lower than WUDI.

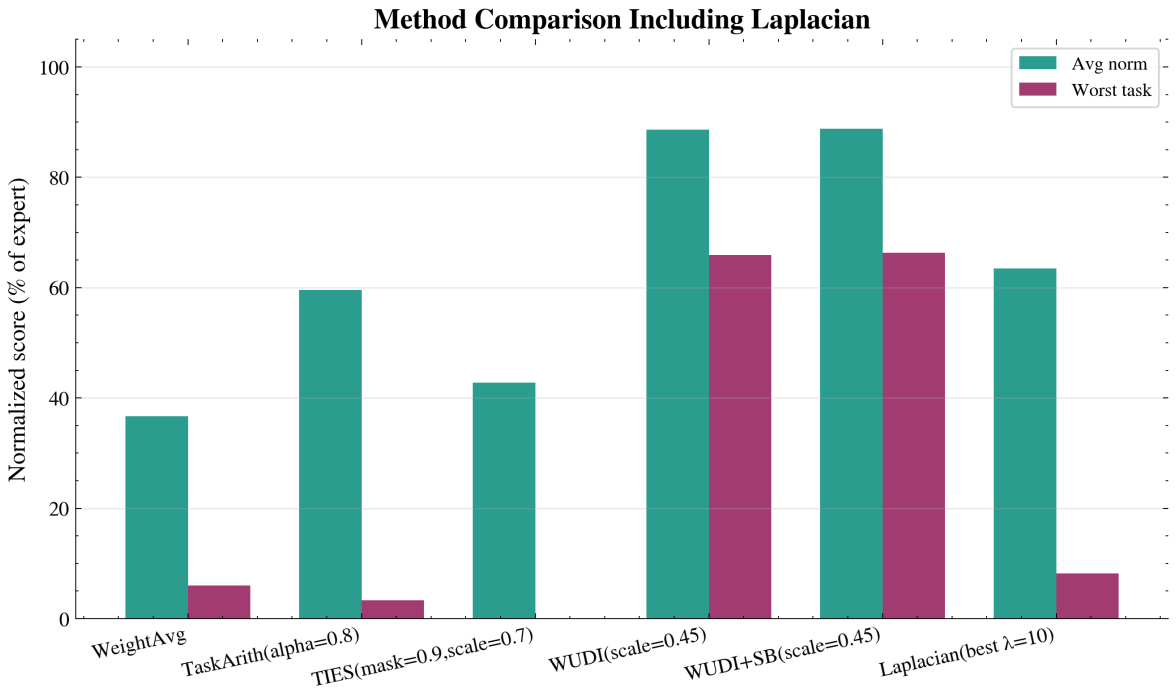

Figure 5.4: Method comparison including the best graph Laplacian merge. Laplacian smoothing improves over weak naive baselines but does not close the gap to WUDI.

The limitation is expressivity. The Laplacian method smooths scalar task weights in a low-rank subspace. WUDI, in contrast, optimizes a free merged vector for each parameter block. The graph prior can regularize task weights, but scalar smoothing cannot fully repair conflicts that require parameter-level geometric corrections.

5.4 Graph-Regularized Adaptive WUDI

Adaptive WUDI was designed to preserve WUDI's free merged vector while learning task-constraint weights. The implementation has a strong parity check: when the learned constraint weights remain uniform, it nearly exactly matches vanilla WUDI. Table 5.3 shows the main adaptive results.

Table 5.3: Representative Graph-Regularized Adaptive WUDI results. Normalized scores are percentages.

Setting	Weight movement	Normalized score	Avg. metric
Vanilla WUDI	0.0000	88.463	73.478
Near-uniform adaptive WUDI	0.0003	88.461	73.476
Softmax weights, no graph, $\rho = 0.01$	0.0443	87.283	72.666
Cosine graph, $\lambda = 0.5$	0.0072	88.260	73.328
Global cosine graph	0.0038	88.267	73.334
Averaged-layer cosine graph	0.0040	88.267	73.334
SAR graph, $\lambda = 0.5$	0.0009	88.372	73.398
Bounded weights, $\epsilon = 0.01$	0.0044	88.459	73.474
Bounded weights, $\epsilon = 0.05$	0.0220	88.168	73.280

The central pattern is monotonic in practice: when the learned weights remain close to uniform, performance stays near WUDI; when they move more, normalized score drops. Graph regularization helps primarily by suppressing weight movement rather than by finding a better non-uniform task weighting.

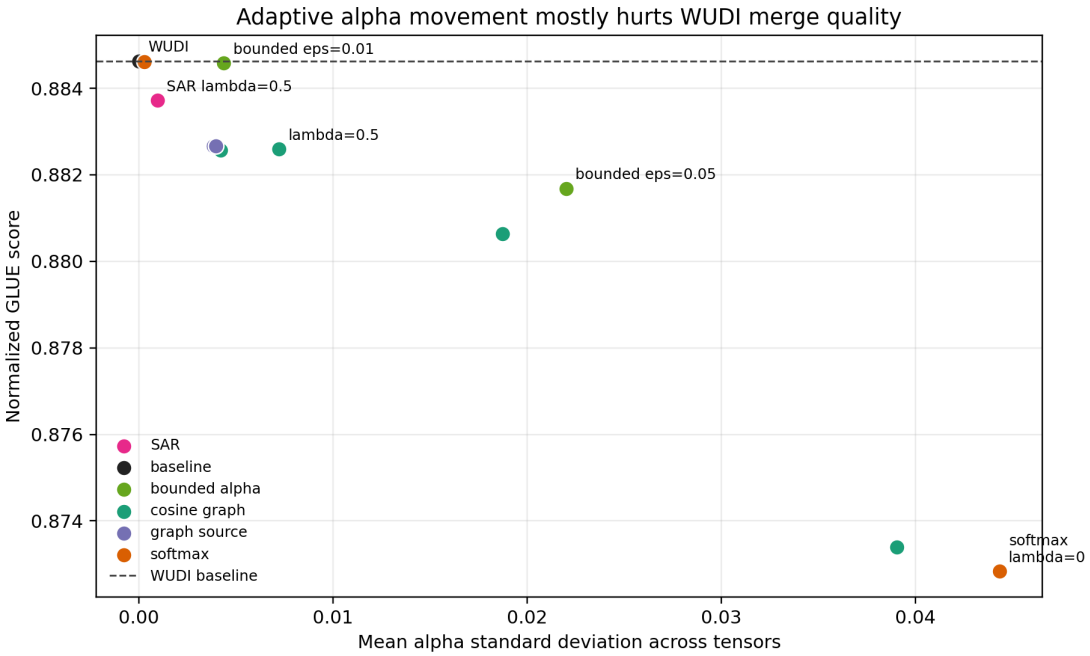

Figure 5.5: Adaptive WUDI constraint-weight movement versus normalized score. The best points are near uniform weights, while larger deviations are associated with lower downstream normalized performance.

5.5 Adaptive Lambda Sweep

For softmax constraint weights with $\rho = 0.01$, increasing graph strength reduces weight movement and recovers much of the performance lost by unregularized adaptive weighting. Table 5.4 summarizes the sweep.

Table 5.4: Softmax adaptive WUDI λ sweep with $\rho = 0.01$.

λ	Weight standard deviation	Normalized score
0.00	0.0443	87.283
0.01	0.0390	87.339
0.10	0.0187	88.063
0.50	0.0072	88.260
1.00	0.0042	88.257

This sweep does not show that the graph discovers a superior weighting. Instead, it shows that graph regularization prevents harmful deviations from WUDI's uniform task constraints. SAR is the best graph prior tested, but it still remains below vanilla WUDI parity.

Chapter6

Discussion

6.1 Why WUDI Wins

The strongest result is that WUDI consistently outperforms methods based on scalar task weights. Weight averaging and Task Arithmetic merge task vectors directly, so they are limited by the compatibility of the source updates. Graph Laplacian merging improves the scalar weights using a similarity graph, but it still constructs the final update from low-rank task-vector coordinates. WUDI instead optimizes a free merged vector under normalized task constraints, giving it more freedom to resolve interference.

The WUDI objective also normalizes each task constraint by the corresponding task-vector norm. This prevents large-norm task vectors from dominating merely because of scale. In the five-task RoBERTa–GLUE setting, this uniform normalized constraint structure appears to be an effective inductive bias.

6.2 What the Graph Laplacian Adds

The graph Laplacian method is still useful diagnostically. It provides a controllable way to smooth task weights, and the λ sweep shows that graph regularization changes performance in an interpretable way. The implementation is also memory safe: all graph operations are small because they operate over tasks, and the low-rank subspace can be built from a 5×5 Gram matrix.

However, the method does not solve the hardest part of the merge. CoLA and STS-B

remain difficult to preserve, and the worst normalized score remains low even at the best λ. This suggests that the main interference is not only a task-level weighting problem.

6.3 Why Adaptive Constraint Weights Do Not Improve WUDI

Graph-Regularized Adaptive WUDI was designed as a conservative WUDI extension. It does not force the merged vector to be a task-weighted sum; it only learns how much each task's WUDI constraint should matter. The parity result confirms that the implementation is valid: near-uniform adaptive weighting matches vanilla WUDI.

The negative result is therefore meaningful. The adaptive objective can learn non-uniform constraint weights, but those weights do not improve downstream normalized GLUE score. When the weights move far from uniform, performance drops. When graph regularization or bounded parameterization prevents movement, performance returns toward WUDI. This implies that the learned weighting objective is misaligned with downstream task balance. It may reduce the layerwise surrogate loss while weakening constraints needed for tasks such as CoLA.

6.4 Graph Source Interpretation

Changing the graph source does not materially change the conclusion. Global cosine and averaged-layer cosine behave almost identically. Denser kNN settings produce only tiny changes. SAR performs best among the tested graph priors, but its advantage mainly comes from stronger suppression of constraint-weight movement. Thus, graph construction quality is not the only bottleneck; the scalar adaptive-weight formulation itself appears limited.

6.5 Limitations

This report focuses on five RoBERTa–GLUE tasks and does not claim that adaptive task-constraint weighting fails universally. The selected tasks are diverse but still drawn from one benchmark family and one base architecture. The conclusions should be tested on larger task sets, other transformer families, and settings with validation-informed or data-assisted merging. The current results are strongest as evidence that scalar adaptive WUDI constraint weights are not beneficial for this specific data-free RoBERTa–GLUE merge.

Chapter7

Conclusion

This project evaluated graph Laplacian and adaptive WUDI methods for data-free model merging of five RoBERTa–GLUE task experts. The experiments confirm that naive merging creates strong task interference. Weight averaging performs poorly, and Task Arithmetic improves average performance but leaves severe worst-task degradation.

WUDI is the strongest tested method. It reaches roughly 88.67% average normalized score in the fixed-baseline analysis, while WUDI with Subspace Boosting reaches 88.83%. These results substantially exceed weight averaging, Task Arithmetic, TIES, and graph Laplacian merging.

Graph Laplacian merging provides an interpretable task-graph smoothing mechanism and improves over weak naive baselines, but its best recorded score of 63.48% remains far below WUDI. Its scalar task-weight formulation is not expressive enough to resolve the main interference patterns in these experiments.

Graph-Regularized Adaptive WUDI has a clean parity check: when the learned constraint weights stay uniform, it matches vanilla WUDI. Across softmax weights, cosine graphs, SAR graphs, and bounded-weight variants, however, the method only approaches WUDI when the weights remain close to uniform. Larger weight movement consistently reduces normalized downstream performance.

The main conclusion is that WUDI's uniform normalized task constraints are already a strong inductive bias for this five-task RoBERTa–GLUE setting. Learning scalar task-constraint weights inside WUDI tends to disturb that balance rather than improve it.

Bibliography

[1] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[2] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2018, pp. 353–355.

[3] Mitchell Wortsman et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *International conference on machine learning*. PMLR. 2022, pp. 23965–23998.

[4] Gabriel Ilharco et al. "Editing Models with Task Arithmetic". In: *International Conference on Learning Representations*. 2023.

[5] Prateek Yadav et al. "TIES-Merging: Resolving Interference When Merging Models". In: *Advances in Neural Information Processing Systems*. 2023.

[6] Runxi Cheng et al. "Whoever Started the Interference Should End It: Guiding Data-Free Model Merging via Task Vectors". In: *arXiv preprint arXiv:2503.08099* (2025).

[7] Ronald Skorobogat, Karsten Roth, and Mariana-Iuliana Georgescu. *Subspace-Boosted Model Merging*. 2025. arXiv: 2506.16506 [cs.LG]. URL: https://arxiv.org/abs/2506.16506.

[8] Zhenyi Lu et al. *Twin-Merging: Dynamic Integration of Modular Expertise in Model Merging*. 2024. arXiv: 2406.15479 [cs.CL]. URL: https://arxiv.org/abs/2406.15479.

[9] Rie Ando and Tong Zhang. "Learning on Graph with Laplacian Regularization". In: *Advances in Neural Information Processing Systems* 19 (2006).

[10] Daniel Marczak et al. *No Task Left Behind: Isotropic Model Merging with Common and Task-Specific Subspaces*. 2025. arXiv: 2502.04959 [cs.LG]. URL: https://arxiv.org/abs/2502.04959.