Comparative Analysis of Model Merging Techniques: Spherical Linear Interpolation versus TIES-Merging

Jayraj Pamnani
Electrical and Computer Engineering
New York University
jmp10051@nyu.edu

Abstract

The rapid proliferation of Large Language Models (LLMs) has necessitated efficient methods for combining the capabilities of distinct fine-tuned checkpoints without the computational cost of additional training. This report analyzes two prominent model merging strategies: Spherical Linear Interpolation (SLERP) and Trim, Elect, Sign & Merge (TIES). We detail the algorithmic steps, theoretical underpinnings, and comparative limitations of each method. While SLERP offers a geometric approach to preserving feature distributions between two models, TIES addresses parameter interference in multi-model settings through sparsity and sign consensus. Through empirical analysis of Python implementations, we provide a comprehensive understanding of the mechanisms governing these techniques and their practical applications in the open-source AI community.

1 Introduction

Model merging has emerged as a critical capability in the open-source artificial intelligence community, allowing practitioners to combine the strengths of multiple specialized models into a single unified architecture. Unlike traditional ensemble methods that require running multiple models at inference time with proportionally increased computational costs, merged models retain the original parameter count and inference complexity while potentially capturing the complementary capabilities of their constituent models.

The fundamental premise of model merging rests on the observation that fine-tuned variants of a common base model often learn complementary features. For instance, one model might excel at mathematical reasoning while another demonstrates superior creative writing capabilities. By intelligently combining their parameters, we can potentially create a unified model that exhibits both strengths without the overhead of maintaining separate inference pipelines.

In this work, we examine two distinct merging methodologies that represent different philosophical approaches to the parameter combination problem:

1. **SLERP (Spherical Linear Interpolation)**: A geometric interpolation method borrowed from computer graphics that preserves the magnitude and directional properties of weight vectors during interpolation.
2. **TIES-Merging**: A spectral method explicitly designed to resolve parameter interference through selective sparsification and democratic sign resolution.

This report dissects these methods to elucidate their distinct operational mechanics, theoretical foundations, and practical requirements for successful deployment.

Preprint. Under review.

1.1 Motivation and Background

Traditional approaches to combining model capabilities typically involve either ensemble methods or continued joint training. Ensemble methods, while effective, incur multiplicative increases in memory and computational requirements. Continued training requires access to the original training data and substantial computational resources, making it impractical for many practitioners in the open-source community.

Model merging offers an attractive alternative by operating directly in parameter space, requiring only the model checkpoints themselves. This democratizes access to specialized model capabilities and enables rapid experimentation with different combinations of expert models.

2 Methodology

2.1 Spherical Linear Interpolation (SLERP)

SLERP serves as an alternative to standard linear averaging (arithmetic mean) for weight interpolation. In high-dimensional parameter spaces characteristic of modern neural networks, linear interpolation can lead to a decrease in the magnitude of the interpolated vector, often resulting in a "washed-out" model with reduced feature sharpness. SLERP addresses this limitation by interpolating along the geodesic path—the shortest arc—on the surface of a hypersphere.

2.1.1 Mathematical Formulation

Given two weight vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ representing parameters from two distinct models, and an interpolation factor $t \in [0, 1]$, SLERP computes the merged weights as follows:

First, we compute the angle Ω between the normalized vectors:

$$\cos \Omega = \frac{\boldsymbol{\theta}_1 \cdot \boldsymbol{\theta}_2}{\|\boldsymbol{\theta}_1\| \|\boldsymbol{\theta}_2\|} \tag{1}$$

The interpolated vector is then given by:

$$\boldsymbol{\theta}_{\text{new}} = \frac{\sin((1-t)\Omega)}{\sin \Omega} \boldsymbol{\theta}_1 + \frac{\sin(t\Omega)}{\sin \Omega} \boldsymbol{\theta}_2 \tag{2}$$

where $t = 0$ yields $\boldsymbol{\theta}_1$, $t = 1$ yields $\boldsymbol{\theta}_2$, and intermediate values produce smooth transitions along the hyperspherical surface.

2.1.2 Algorithmic Implementation

The SLERP procedure proceeds through the following steps:

Algorithm 1 SLERP for Model Merging

Require: Models M_1, M_2 with parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, interpolation factor t
Ensure: Merged model with parameters $\boldsymbol{\theta}_{\text{new}}$

for each parameter tensor pair $(\boldsymbol{\theta}_1^{(i)}, \boldsymbol{\theta}_2^{(i)})$ **do**
  Flatten tensors to vectors
  Compute dot product: $d = \boldsymbol{\theta}_1^{(i)} \cdot \boldsymbol{\theta}_2^{(i)}$
  Compute angle: $\Omega = \arccos(d / (\|\boldsymbol{\theta}_1^{(i)}\| \|\boldsymbol{\theta}_2^{(i)}\|))$
  if $\sin \Omega \approx 0$ **then**
    Use linear interpolation (vectors nearly parallel)
  else
    Compute scale factors: $s_1 = \sin((1-t)\Omega)/\sin \Omega$, $s_2 = \sin(t\Omega)/\sin \Omega$
    $\boldsymbol{\theta}_{\text{new}}^{(i)} = s_1 \boldsymbol{\theta}_1^{(i)} + s_2 \boldsymbol{\theta}_2^{(i)}$
  end if
  Reshape back to original tensor shape
end for

2.1.3 Theoretical Justification

SLERP is preferred for merging two models because it preserves the variance and geometric properties of the weights more effectively than linear averaging. In deep neural networks, the direction of a weight vector often encodes the learned features, while the magnitude relates to the confidence or intensity of those features. SLERP ensures that the merged model effectively transitions between the two parent models' feature spaces without traversing the low-magnitude "dead zones" that linear averaging might cross.

The preservation of magnitude is particularly important in transformer-based architectures where attention mechanisms and layer normalization make the scale of weights meaningful for model behavior. By maintaining constant magnitude during interpolation, SLERP avoids the attenuation effects that can diminish model expressiveness.

2.2 TIES-Merging (Trim, Elect, Sign & Merge)

TIES-Merging is designed to address a specific phenomenon known as "parameter interference." When multiple models are fine-tuned from a common base model, they may update the same parameters in conflicting directions (positive versus negative gradients) or with varying magnitudes. Naive averaging can lead to these updates canceling each other out, resulting in a merged model that performs worse than its constituents.

2.2.1 Task Vector Formulation

TIES operates on the concept of *task vectors*, defined as the parameter-space difference between a fine-tuned model and its base model:

$$\boldsymbol{\tau}_i = \boldsymbol{\theta}_{\text{ft}}^{(i)} - \boldsymbol{\theta}_{\text{base}} \tag{3}$$

These task vectors represent the specific adaptations learned during fine-tuning for a particular task or capability. The goal of TIES is to combine multiple task vectors while minimizing destructive interference.

2.2.2 Three-Phase Algorithm

The TIES method proceeds through three distinct phases:

Phase 1: Trim (Sparsification)

The trim phase isolates the most significant parameter changes by applying a density threshold. For each task vector $\boldsymbol{\tau}_i$, we retain only the top-k values by magnitude, where $k = \lfloor \delta \cdot |\boldsymbol{\tau}_i| \rfloor$ and δ is the density hyperparameter (typically 0.2):

$$\tilde{\boldsymbol{\tau}}_i[j] = \begin{cases} \boldsymbol{\tau}_i[j] & \text{if } |\boldsymbol{\tau}_i[j]| \geq \text{threshold}_\delta(\boldsymbol{\tau}_i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This step operates under the assumption that small-magnitude updates represent noise or less critical adaptations that can be safely discarded.

Phase 2: Elect (Sign Resolution)

The elect phase resolves conflicts where models disagree on the direction of parameter updates. For each parameter position j, we compute the sign consensus:

$$s_j = \text{sign}\left(\sum_{i=1}^{N} \tilde{\boldsymbol{\tau}}_i[j]\right) \tag{5}$$

This creates a "democratic" decision about whether each parameter should increase or decrease in the merged model.

Phase 3: Merge (Disjoint Aggregation)

Finally, we combine only those values that agree with the elected sign:

$$\boldsymbol{\tau}_{\text{merged}}[j] = \frac{\sum_{i:\text{sign}(\tilde{\boldsymbol{\tau}}_i[j])=s_j} \tilde{\boldsymbol{\tau}}_i[j]}{|\{i : \text{sign}(\tilde{\boldsymbol{\tau}}_i[j]) = s_j\}|} \tag{6}$$

The final merged model is then:

$$\boldsymbol{\theta}_{\text{merged}} = \boldsymbol{\theta}_{\text{base}} + \boldsymbol{\tau}_{\text{merged}} \tag{7}$$

Algorithm 2 TIES-Merging

Require: Base model M_{base} with parameters $\boldsymbol{\theta}_{\text{base}}$
Require: Fine-tuned models $\{M_1, \ldots, M_N\}$ with parameters $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$
Require: Density parameter $\delta \in [0, 1]$
Ensure: Merged model with parameters $\boldsymbol{\theta}_{\text{merged}}$

// Phase 1: Compute and trim task vectors
for $i = 1$ to N **do**
  $\boldsymbol{\tau}_i = \boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{base}}$
  threshold $= \text{Percentile}(|\boldsymbol{\tau}_i|, 100(1-\delta))$
  $\tilde{\boldsymbol{\tau}}_i = \boldsymbol{\tau}_i \odot \mathbb{1}_{|\boldsymbol{\tau}_i| \geq \text{threshold}}$
end for
// Phase 2: Elect sign for each parameter
for each parameter index j **do**
  $s_j = \text{sign}\left(\sum_{i=1}^{N} \tilde{\boldsymbol{\tau}}_i[j]\right)$
end for
// Phase 3: Merge with sign agreement
for each parameter index j **do**
  $\mathcal{A}_j = \{i : \text{sign}(\tilde{\boldsymbol{\tau}}_i[j]) = s_j\}$
  $\boldsymbol{\tau}_{\text{merged}}[j] = \frac{1}{|\mathcal{A}_j|} \sum_{i \in \mathcal{A}_j} \tilde{\boldsymbol{\tau}}_i[j]$
end for
$\boldsymbol{\theta}_{\text{merged}} = \boldsymbol{\theta}_{\text{base}} + \boldsymbol{\tau}_{\text{merged}}$

2.2.3 Theoretical Motivation

The TIES methodology is grounded in the observation that parameter interference is a primary cause of performance degradation in merged models. By explicitly filtering for consensus and agreement, TIES aims to preserve the coherent adaptations while eliminating conflicting updates. The sparsification step further reduces interference by removing potentially noisy low-magnitude updates that may not generalize well.

3 Comparative Analysis

3.1 Fundamental Differences

The two methods represent fundamentally different philosophies for model merging:

- **Geometric vs. Component-wise**: SLERP treats weights as high-dimensional vectors and focuses on the geometric path between them, preserving the structure of the parameter manifold. TIES treats weights as individual components subject to interference and explicitly filters them at the element level.
- **Model Capacity**: SLERP is mathematically constrained to *pairwise* merging—combining exactly two models at a time. Merging three or more models requires a hierarchical approach (merge $A + B$, then merge result with C). TIES is inherently designed for *multi-model* merging, capable of integrating dozens of task vectors simultaneously by finding global consensus.
- **Base Model Dependency**: SLERP operates directly on model parameters and does not require knowledge of a base model. TIES fundamentally requires a common base model to compute task vectors, making it inapplicable when such a base is unavailable or unknown.

3.2 Limitations and Trade-offs

3.2.1 SLERP Limitations

1. **Pairwise Constraint**: The method is inherently limited to merging two models at a time. For N models, this requires $N - 1$ sequential merge operations, each potentially introducing approximation errors.
2. **Ignorance of Interference**: SLERP assumes that a smooth geometric path exists between models and that this path represents meaningful intermediate models. It does not account for the possibility that parameters might be functionally incompatible—for instance, safety-aligned weights conflicting with uncensored model weights.
3. **Sensitivity to Magnitude Differences**: When models have significantly different parameter magnitudes, the geometric interpolation may not produce meaningful results, as the angle-based weighting can be dominated by scale differences.

3.2.2 TIES Limitations

1. **Information Loss**: The trim phase intentionally discards 70-90% of fine-tuning updates in typical configurations. While this reduces noise and interference, it risks discarding subtle but important adaptations, particularly for nuanced tasks requiring fine-grained parameter adjustments.
2. **Hyperparameter Sensitivity**: TIES introduces the density hyperparameter δ, which must be carefully tuned for each merging scenario. An incorrect density can lead to either excessive sparsity (underperforming merged model) or insufficient filtering (interference-heavy model).
3. **Base Model Requirement**: The method fundamentally requires access to the common base model to compute task vectors. This makes TIES inapplicable in scenarios where only fine-tuned checkpoints are available, or where models have been fine-tuned from different base models.
4. **Democratic Assumption**: The sign election mechanism assumes that majority agreement indicates correctness. However, in cases where a minority update represents a crucial capability, the democratic approach may suppress important adaptations.

3.3 Computational Complexity

Both methods scale linearly with the number of parameters, but TIES incurs additional overhead:

- **SLERP**: $O(P)$ where P is the total number of parameters. Requires one forward pass through all parameters.
- **TIES**: $O(N \cdot P)$ where N is the number of models being merged. Requires computing task vectors for all models, performing percentile calculations for thresholding, and aggregating across models.

For merging large language models with billions of parameters, both methods remain tractable, but TIES requires proportionally more memory to hold all task vectors simultaneously.

4 Requirements for Successful Model Merging

For both SLERP and TIES methods to produce valid merged models, several strict requirements must be satisfied:

4.1 Architectural Requirements

1. **Identical Architecture**: Models must share the exact same neural network architecture, including layer counts, hidden dimensions, attention heads, and activation functions. Weight matrices must have identical shapes for element-wise operations to be well-defined.

2. **Tokenization Compatibility**: Models must use identical tokenizers and vocabulary sizes. Mismatches in special token IDs (e.g., `pad_token_id`, `eos_token_id`) can lead to generation failures or undefined behavior.
3. **Precision Consistency**: All models should use the same numerical precision (e.g., FP16, BF16, FP32) to avoid numerical instabilities during interpolation or aggregation.

4.2 Training Lineage Requirements

1. **Common Ancestry (Strongly Recommended)**: While not strictly mathematically required for SLERP, merging works best when models share a common pre-trained base. This ensures that the feature representations are anchored in a shared embedding space.
2. **Base Model Access (Required for TIES)**: TIES explicitly requires access to the common base model to compute task vectors. The base model must be the exact checkpoint from which all fine-tuned models were derived.
3. **Compatible Fine-tuning Procedures**: Models fine-tuned with drastically different procedures (e.g., full fine-tuning versus LoRA) may not merge effectively, as their parameter updates may operate in fundamentally different subspaces.

4.3 Practical Considerations

1. **Complementary Capabilities**: Merging is most effective when models possess complementary rather than conflicting capabilities. Merging two models fine-tuned for similar tasks may yield marginal improvements.
2. **Quality Control**: Not all merges produce functional models. Post-merge evaluation on diverse benchmarks is essential to verify that the merged model maintains acceptable performance.
3. **Iterative Refinement**: Hyperparameters (SLERP interpolation factor t, TIES density δ) often require iterative tuning based on downstream task performance.

5 Experimental Insights

Based on the implementation analysis, several practical insights emerge:

5.1 SLERP in Practice

The SLERP implementation demonstrates high fidelity when merging models with similar training procedures and compatible objectives. Key observations include:

- Interpolation factors $t \in [0.3, 0.7]$ tend to produce more balanced merges than extreme values.
- The method exhibits numerical stability even with large language models containing billions of parameters.
- Merging time is dominated by I/O operations for loading and saving checkpoints rather than the interpolation computation itself.

5.2 TIES in Practice

The TIES implementation reveals interesting trade-offs:

- Density values around $\delta = 0.2$ (keeping 20% of updates) provide a good balance between noise reduction and information preservation for typical language model merges.
- The sign election mechanism proves particularly effective when merging instruction-tuned models with different formatting conventions.
- Memory requirements can become substantial when merging many models, as all task vectors must be computed and held in memory during the election phase.

6 Discussion and Future Directions

6.1 When to Use Each Method

The choice between SLERP and TIES depends on the specific merging scenario:

Use SLERP when:

- Merging exactly two models
- High-fidelity preservation of both models' capabilities is critical
- Base model is unavailable
- Models are closely related (e.g., different checkpoints from the same training run)

Use TIES when:

- Merging three or more models
- Parameter interference is a known concern
- Base model is available
- Models have potentially conflicting updates that need resolution

6.2 Emerging Techniques

Recent developments in model merging have introduced additional methods that build upon or combine these approaches:

- **DARE-TIES**: Combines TIES with dropout-based sparsification for improved robustness.
- **Task Arithmetic**: Uses scaled task vectors for more flexible combination of capabilities.
- **RegMean**: Incorporates regularization to prevent extreme values in merged models.
- **Evolutionary Merging**: Uses genetic algorithms to optimize merging parameters and model selection.

6.3 Open Questions

Several fundamental questions remain in the model merging literature:

1. Can we develop theoretical guarantees about when merging will succeed or fail?
2. How can we predict the optimal hyperparameters without extensive post-merge evaluation?
3. Can merging methods be extended to work across different architectures or model families?
4. What is the theoretical limit on the number of capabilities that can be merged into a single model?

7 Conclusion

This report has provided a comprehensive analysis of two prominent model merging techniques: SLERP and TIES-Merging. SLERP offers a geometrically principled approach to pairwise model interpolation that preserves feature magnitude and direction, making it ideal for high-fidelity merging of two closely related models. TIES-Merging provides a sophisticated multi-model merging framework that explicitly addresses parameter interference through sparsification and democratic sign resolution.

The choice between these methods depends critically on the specific requirements of the merging scenario, including the number of models to be merged, the availability of base models, and the expected degree of parameter interference. Neither method dominates the other across all scenarios; rather, they represent complementary tools in the model merging toolkit.

As the open-source AI community continues to develop increasingly specialized models, efficient and effective merging techniques will become increasingly important. Future work should focus

on developing more sophisticated merging strategies that can handle diverse model families, provide theoretical performance guarantees, and automatically adapt hyperparameters based on model characteristics.

The implementations analyzed in this report demonstrate that model merging has matured from a theoretical curiosity to a practical technique for capability combination. With careful consideration of the requirements and limitations outlined here, practitioners can successfully leverage these methods to create unified models that capture the strengths of multiple expert systems.

Acknowledgments

This analysis was based on implementations from the MergeKit toolkit and related open-source projects in the model merging community.

References

[1] Arcee AI. MergeKit: A Toolkit for Merging Large Language Models. `https://github.com/arcee-ai/mergekit`, 2024.

[2] 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*, 2023.

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

[4] Yu Lu, Xingyu Chen, Shengcao Cao, Ke Zhang, Zhiqi Bu, and Meng Cao. DARE: Drop and rescale for efficient fine-tuning. In *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[5] Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH Computer Graphics*, volume 19, pages 245–254, 1985.

[6] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022.

[7] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.

[8] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020.

[9] Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems*, 2021.

[10] Hugging Face. Model merging documentation. `https://huggingface.co/docs/transformers/model_merging`, 2024.