# ML2025 HW9
# Model Merging

TAs: 黃筱穎、陳又華、謝翔

Email: ntu-ml-2025-spring-ta@googlegroups.com

**Deadline: 2025/6/6 23:59:59 (UTC+8)**

# Links

- Course Website
- NTU COOL
- Colab Sample Code
- Kaggle Sample Code
- Dataset
- Judgeboi
- PEFT ckpts: GSM8K, ARC
- (2025/05/21 Update) TA version peft package: link1, link2, link3, link4
- (Deprecated to avoid plagiarism)How to build a private customized peft package

# Outline

- Task Description
- Dataset
- Eval Metric and Answer Extraction
- Merging Algorithms, TODO and Hints
- Submission and Grading
- Reference

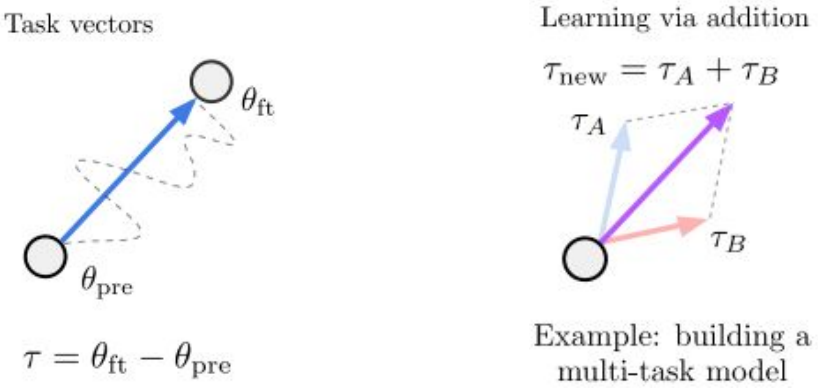
# Task Description

# Task Description

- Goal: Learn to merge models with distinct capabilities at the **parameter level** to build a unified, multi-task model without additional training
- Explore various model merging algorithms to develop a unified model that preserves or improves performance across two tasks

# Model Merging

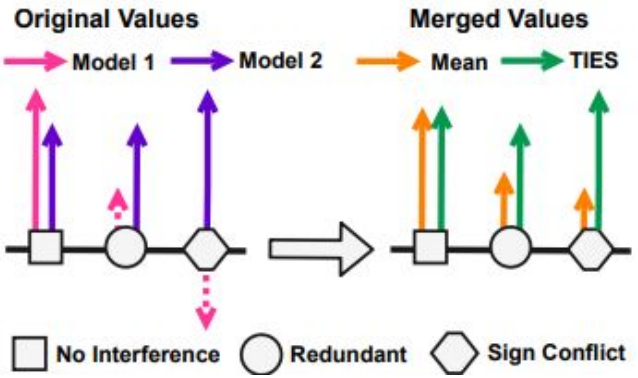
- Def. refer to the process of merging models with **simple arithmetic** on parameters **without retraining** from scratch or accessing original training data, to preserve or integrate capabilities from each source model (e.g., tasks, domains) into a unified model




Ilharco, Gabriel, et al. "Editing models with task arithmetic." *arXiv preprint arXiv:2212.04089* (2022).

# Model Merging

- recent emergence of large foundation models and pretraining-finetuning paradigm have motivated more merging techniques → *Multi-Task Learning*
- redundant parameter or sign conflicts in different task vectors ⇒ ***parameter interference*** → degraded performance




Lee, Yeoreum, Jinwook Jung, and Sungyong Baik. "Mitigating Parameter Interference in Model Merging via Sharpness-Aware Fine-Tuning." *The Thirteenth International Conference on Learning Representations*.

# Task Description

- Most Important: During merging, avoid problems like parameter interference, etc. to preserve performance on both math (GSM8K) and science (ARC) reasoning tasks
- TAs have finished PEFT on two individual tasks (freeze base model).
  - base model: llama-2-7b-chat-bnb-4bit (llama-2-7b-chat quant ver.)
  - LoraConfig: r=8, target_modules=["q_proj", "k_proj", "v_proj"], lora_alpha=16, lora_dropout=0.05 (LoRA related, refer to HW5)
  - Performances (Accuracy)
    - after FT: science (63%), math (52.5%)
    - base: science (44%), math (37%)

# Task Description

- Science (ARC) example after PEFT

Question:

A scientist discovers that a certain substance will be useful in treating nerve damage. What is the best way she can communicate her results to the scientific community?  (A) She can call other scientists who also work in nerve research. (B) She can write a letter to hospitals that treat patients with nerve damage. (C) She can put her conclusions on a Web site that focuses on the nervous system. (D) She can publish her results in a scientific journal that covers issues relating to nerves.

Answer after PEFT:

The answer is (D) She can publish her results in a scientific journal that covers issues relating to nerves. Scientific journals are the primary means of communicating scientific research to the scientific community. They provide a platform for scientists to share their findings with other scientists, and they are peer-reviewed, meaning that other scientists review and evaluate the research before it is published. This helps ensure that the research is accurate and reliable. Calling other scientists, writing a letter to hospitals, or putting conclusions on a Web site are not as effective in communicating the results to the scientific community.</s>

# Task Description

- Math (GSM8K) example after PEFT

Question:

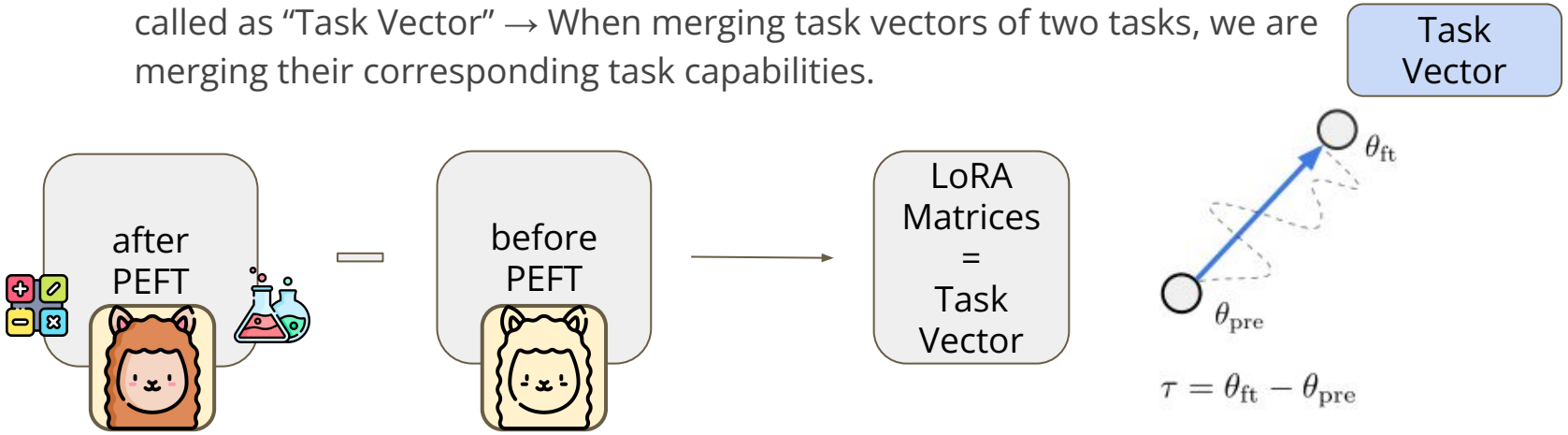
Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week? (A) 294 (B) 312 (C) 320 (D) 432

Answer after PEFT:

Lloyd has 252 eggs / 12 eggs = <<252/12=21>>21 dozens of eggs. Lloyd makes 21 dozens of eggs * $2/dozen = $<<21*2=42>>42 per day. Lloyd makes $42/day * 7 days/week = $<<42*7=294>>294 per week. The answer is (A) 294. </s>

# Task Description

- *peft*: Parameter-Efficient Fine-Tuning package can help us operate arithmetic on LoRA matrices
- relationship between LoRA weights and task vectors
  - The difference between “before peft” and “after peft” is the newly fine-tuned LoRA weights. → Task information is respectively stored in low-rank matrices. → called as “Task Vector” → When merging task vectors of two tasks, we are merging their corresponding task capabilities.

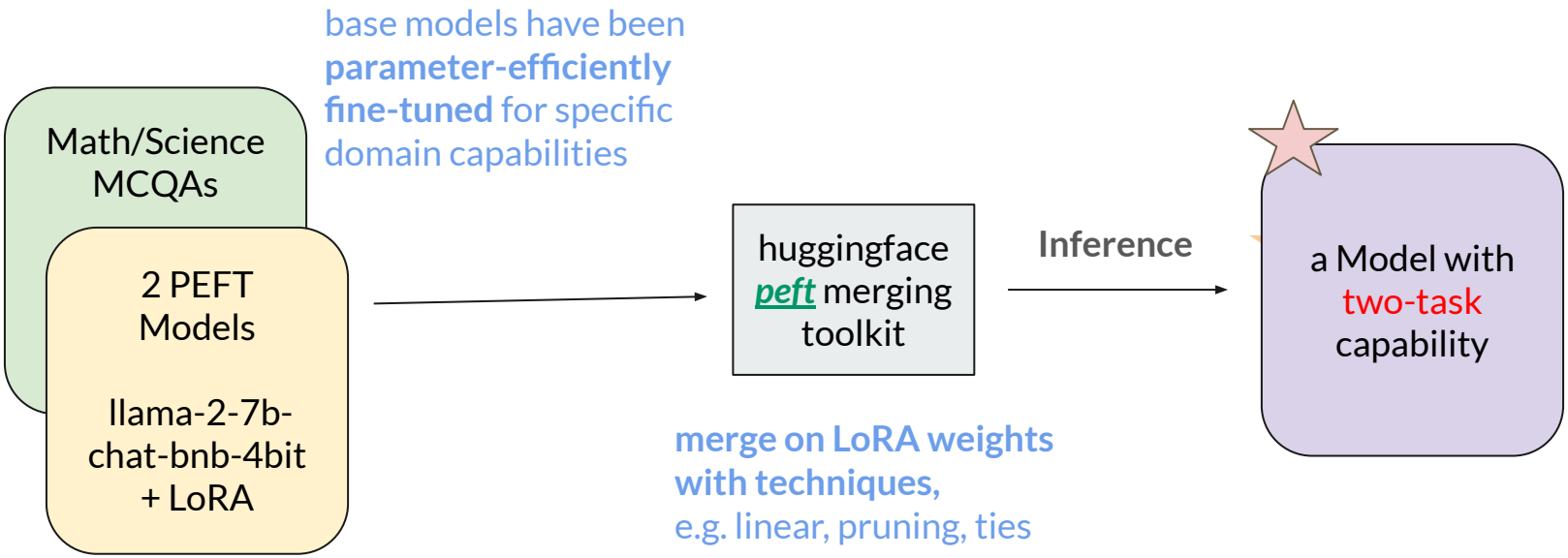


# Task Description: Overall

- Use *peft* package to apply arithmetic operations on LoRA weights and inference the merged model on two tasks.

- during inference of all questions, you must use the same merged model (consistent merging setting)

# Task Description: Overall



# Dataset

# Dataset

- 400 multi-choice questions from both ARC(Easy, Challenge) and GSM8K(MCQA ver.) datasets
- correct dataset link (huggingface)

# Dataset

- ARC (grade-school level, multiple-choice science questions)

| id<br>string · *lengths*<br>6↔7 97.8% | task_name<br>string · *classes*<br>ARC 100% | instruction<br>string · *classes*<br>You are gi… 100% | question<br>string · *lengths*<br>144↔205 18.8% | options<br>dict |
|---|---|---|---|---|
| arc_201 | ARC | You are given a science question and four answer options (associated with "A", "B", "C", "D"). Your task is to find the correct answer based on scientific facts, knowledge, and reasoning. There is only one correct answer for each question. | Winds blowing inland from oceans tend to have greater moisture than winds blowing over land. How does the high moisture content affect the coastal area climate? | { "A": "There is less condensation.", "B": "There are fewer hurricanes.", "C": "There is greater precipitation.", "D": "There are more smog-filled areas." } |

# Dataset

- GSM8K (grade school math word problems, multiple-choice questions ver.)

| id<br>string · *lengths*<br>9–10 93.8% | task_name<br>string · *classes*<br>GSM8K 100% | instruction<br>string · *classes*<br>You are gi… 100% | question<br>string · *lengths*<br>186–240 26% | options<br>dict |
|---|---|---|---|---|
| gsm8k_1124 | GSM8K | You are given a math question and four answer options (associated with "A", "B", "C", "D"). Your task is to carefully analyze the problem, apply logical reasoning, and select the correct answer. There is only one correct answer for each question. | Elijah has one dog that is one-fourth the weight of Kory's dog and another dog that is half the weight of Kory's dog. If Kory's dog is 60 pounds, how much do Elijah and Kory's dogs weigh altogether, in pounds? | {<br>"A": "82.5",<br>"B": "60",<br>"C": "105",<br>"D": "72"<br>} |

# Eval Metric and Answer Extraction

- Evaluation Metric
  - Accuracy on 400 MCQA problems
- Answer Extraction Methods (on Judgeboi)
  - LLM Judge (GPT-4o) to retrieve the actual predicted option

# Merging Algorithms, TODO and Hints

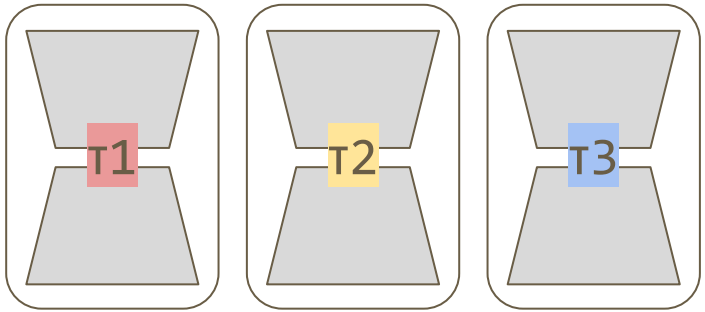
# Merging Algorithms

- In HW9, we implement merging on LoRA A/B matrices of the two fine-tuned checkpoints (task vectors of the science and math mcqa tasks).
- *peft* package helps apply merging algorithms directly on tensor (matric) level, on all task vectors.

# Terminology in Merging Algorithms

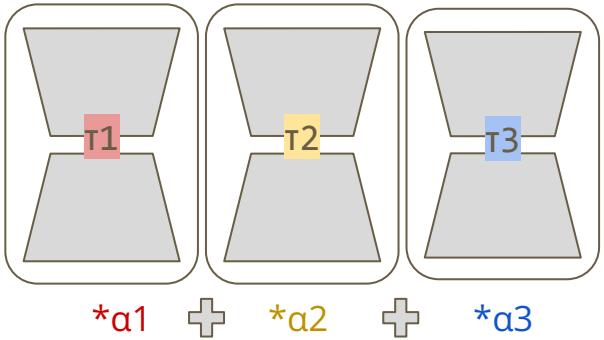
3 common variables from the implementation in *peft*...

- task vector τ
  - LoRA A and B matrices within [q_proj, k_proj, v_proj] modules in all attention layers, each matrix with shape (4096,8) (tensor)
- weights `list(α)`
  - weights or scalar coefficients of task vectors
- density d
  - fraction of values to preserve in a matrix



# Merging Algorithms

- Task Arithmetic / linear (weights)



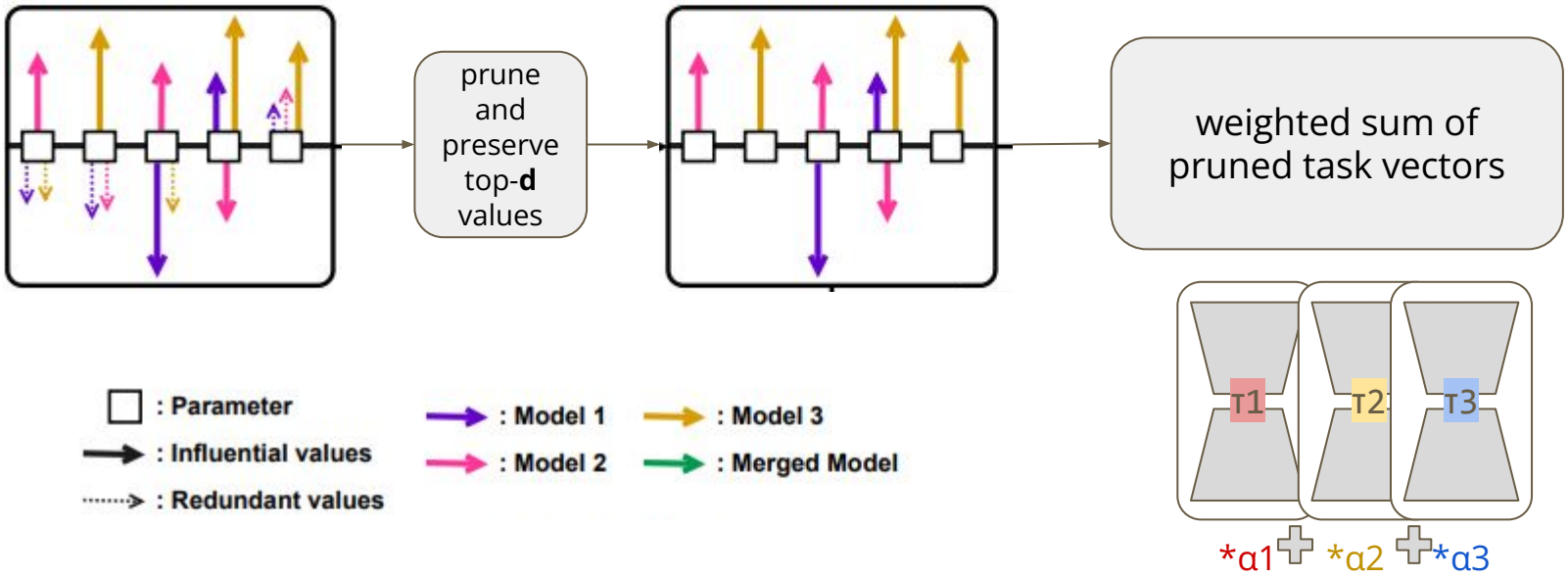

weighted sum of task vectors

Ilharco, Gabriel, et al. "Editing models with task arithmetic."
*arXiv preprint arXiv:2212.04089* (2022).

# Merging Algorithms

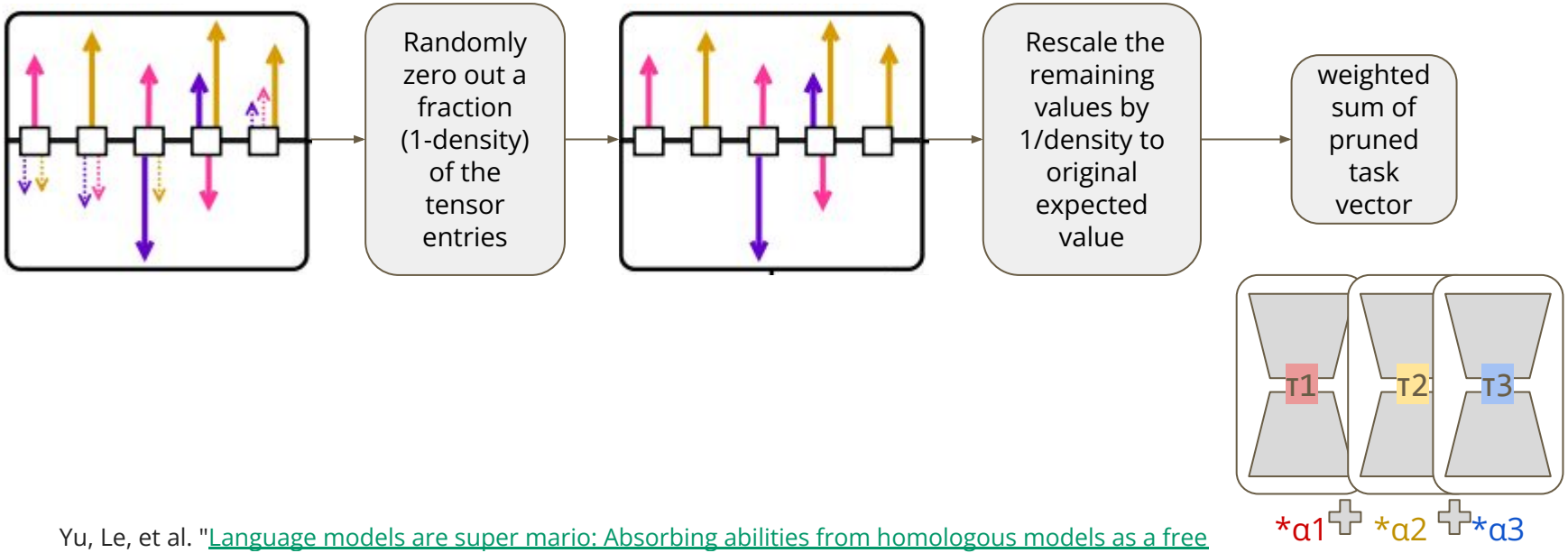
Arrow: direction → sign, length → magnitude

- Magnitude Prune (density, weights)



# Merging Algorithms

- DARE Linear (density, weights)




Yu, Le, et al. "Language models are super mario: Absorbing abilities from homologous models as a free lunch." Forty-first International Conference on Machine Learning. 2024.

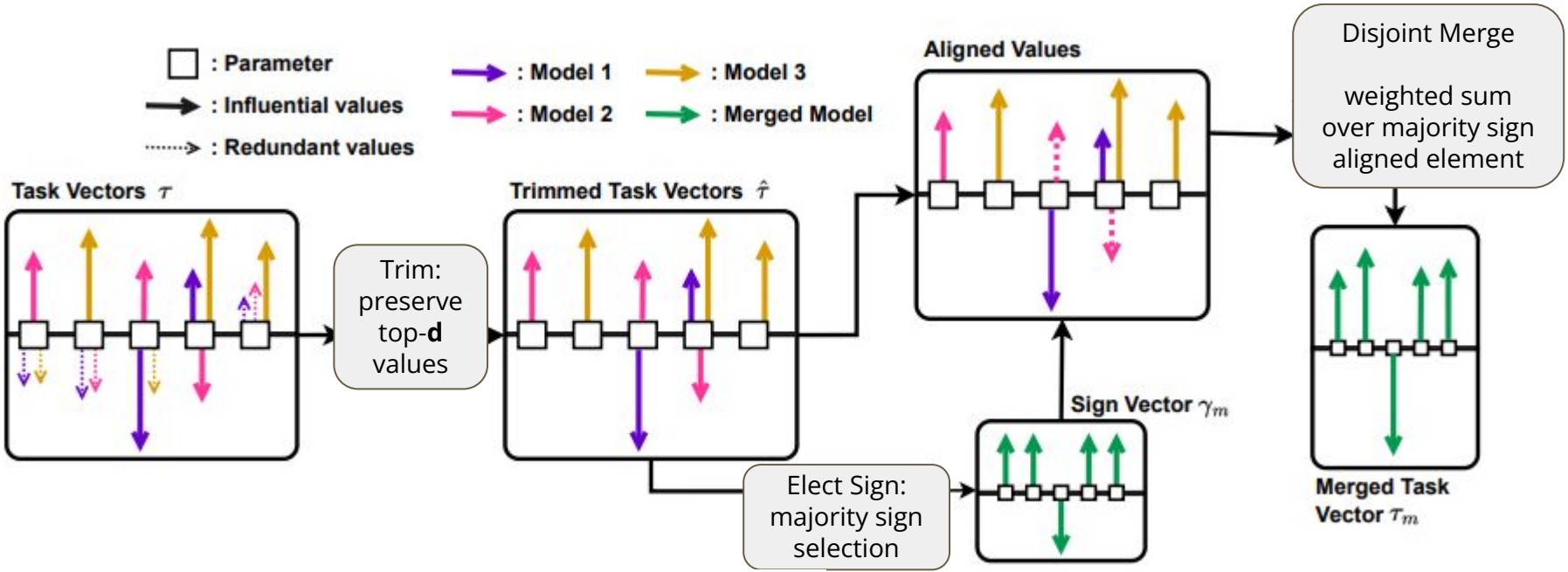
# Merging Algorithms

- DARE Linear
  - derive task vectors
  - **D**rop: randomly zero out a fraction (1 - d) (Bernouli(d) masks) of the tensor entries to preserve vector elements with density d
  - **A**nd
  - **RE**scale: rescale remaining ones by 1/d to approximate expected value of the original embeddings
  - weighted sum refined task vectors

Yu, Le, et al. "Language models are super mario: Absorbing abilities from homologous models as a free lunch." Forty-first International Conference on Machine Learning. 2024.

# Merging Algorithms

- TIES (density, weights)




Yadav, Prateek, et al. "Ties-merging: Resolving interference when merging models." *Advances in Neural Information Processing Systems* 36 (2023): 7093-7115.

# Merging Algorithms

- TIES
  - derive task vectors
  - **Tr**I**m**: prune task vector by magnitude, preserve top-**d** important parameters
  - **E**lect **S**ign : Determine +/- for each parameter by summing up (total/frequency)
  - Disjoint **Merge**: weighted sum over majority sign aligned elements

Yadav, Prateek, et al. "Ties-merging: Resolving interference when merging models." *Advances in Neural Information Processing Systems* 36 (2023): 7093-7115.

# Merging Algorithms

- TIES

**Algorithm 1** TIES-MERGING Procedure.

**Input:** Fine-tuned models $\{\theta_t\}_{t=1}^{n}$, Initialization $\theta_{\text{init}}$, $k$, and $\lambda$.

**Output:** Merged Model $\theta_m$

**forall** $t$ ***in*** $1, ..., n$ **do**

▷ Create task vectors.

$\tau_t = \theta_t - \theta_{\text{init}}$

▷ Step 1: Trim redundant parameters.

$\hat{\tau}_t \leftarrow \text{keep\_topk\_reset\_rest\_to\_zero}(\tau_t, k)$

$\hat{\gamma}_t \leftarrow sgn(\hat{\tau}_t)$

$\hat{\mu}_t \leftarrow |\hat{\tau}_t|$

**end**

▷ Step 2: Elect Final Signs.

$\gamma_m = sgn(\sum_{t=1}^{n} \hat{\tau}_t)$

▷ Step 3: Disjoint Merge.

**forall** $p$ ***in*** $1, ..., d$ **do**

$\mathcal{A}^p = \{t \in [n] \mid \hat{\gamma}_t^p = \gamma_m^p\}$

$\tau_m^p = \frac{1}{|\mathcal{A}^p|} \sum_{t \in \mathcal{A}^p} \hat{\tau}_t^p$

**end**

▷ Obtain merged checkpoint

$\theta_m \leftarrow \theta_{\text{init}} + \lambda * \tau_m$

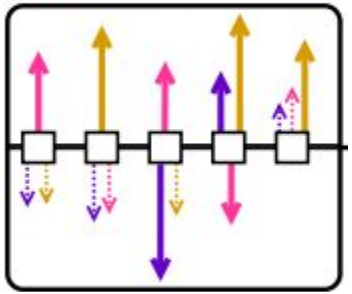
**return** $\theta_m$

Ref. Yadav, Prateek, et al. "Ties-merging: Resolving interference when merging models." *Advances in Neural Information Processing Systems* 36 (2023): 7093-7115.

# Merging Algorithms

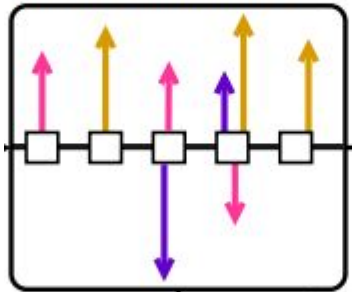
- SCE (density)

$$\eta_{j,m} = \frac{\sum \hat{\delta}^2_{j,m}}{\sum_j \sum \hat{\delta}^2_{j,m}}$$




Select (prune)

consider element variances between task vectors




Calculate coefficient

calculate the sum of squares of elements

Erase minority elements

select majority sign of elements

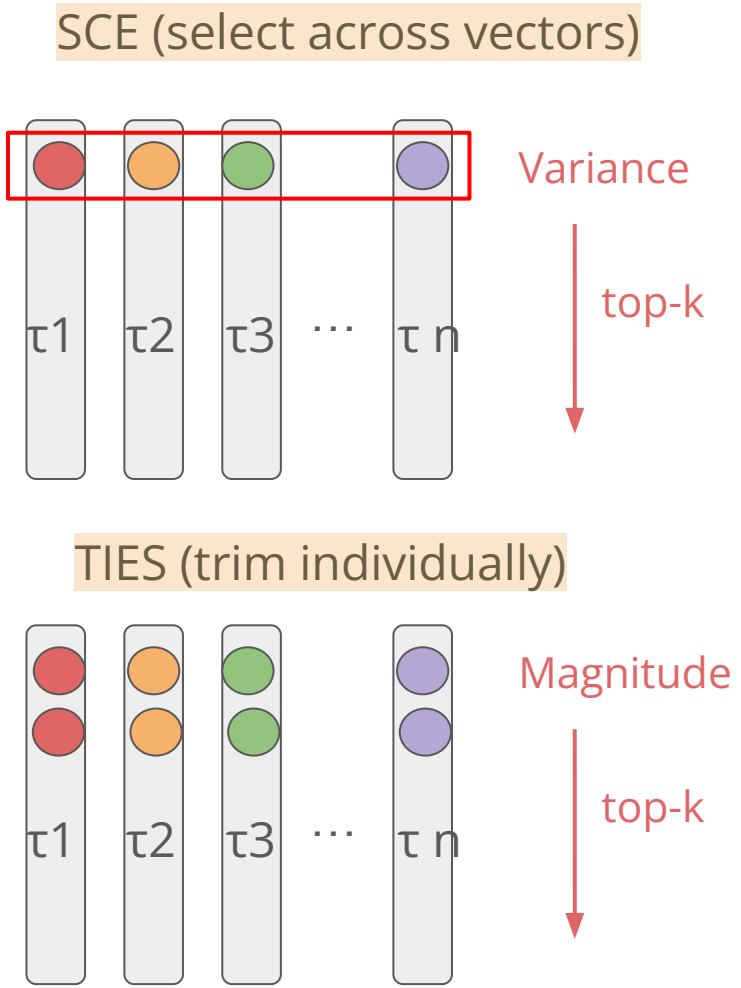
Disjoint Merge

weighted sum over majority sign aligned element

Wan, Fanqi, et al. "Fusechat: Knowledge fusion of chat models."
*arXiv preprint arXiv:2408.07990* (2024).

# Merging Algorithms

- SCE v.s. TIES
  - Select – similar to pruning, further consider variations across different task vectors
  - Trim – prune each task vector individually



# Merging Algorithms

- SCE
  - derive task vectors
  - **S**: select top-k variance elements in matrices (among different task vectors)
    - v.s. TIES (pruning individually)
  - **C**: sum of squares of elements to obtain merging coefficient for each target LLM
  - **E**: filter elements with minority directions

Wan, Fanqi, et al. "Fusechat: Knowledge fusion of chat models."
*arXiv preprint arXiv:2408.07990* (2024).

# Merging Algorithms

- SCE

**Algorithm 1** SCE Procedure

**Input:** target LLMs parameters $\{\phi_j\}_{j=1}^{K-1}$, pivot LLM parameters $\theta_v$, threshold $\tau$.

**Output:** merged LLM parameters $\Phi$

▷ Create fusion vectors

$$\{\delta_j\}_{j=1}^{K-1} = \{\phi_j - \theta_v\}_{j=1}^{K-1} \quad (5)$$

▷ Calculate parameter matrix-level merging coefficients

**for** $\{\delta_{j,m}\}_{j=1}^{K-1} \in \{\delta_j\}_{j=1}^{K-1}$ **do**

▷ Step 1: Select salient elements

$$\{\hat{\delta}_{j,m}\}_{j=1}^{K-1} = \text{Select}(\{\delta_{j,m}\}_{j=1}^{K-1}, \tau) \quad (6)$$

▷ Step 2: Calculate coefficients

$$\{\eta_{j,m}\}_{j=1}^{K-1} = \text{Calculate}(\{\hat{\delta}_{j,m}^2\}_{j=1}^{K-1}) \quad (7)$$

$$\eta_{j,m} = \frac{\sum \hat{\delta}_{j,m}^2}{\sum_j \sum \hat{\delta}_{j,m}^2}$$

▷ Step 3: Erase minority elements

$$\{\delta'_{j,m}\}_{j=1}^{K-1} = \text{Erase}(\{\hat{\delta}_{j,m}\}_{j=1}^{K-1}) \quad (8)$$

▷ Update merged LLM parameters

$$\Phi_m = \theta_{v,m} + \sum_{j=1}^{K-1} \eta_{j,m}\delta'_{j,m} \quad (9)$$

**end**

**return** $\Phi$

Ref. Wan, Fanqi, et al. "Fusechat: Knowledge fusion of chat models." *arXiv preprint arXiv:2408.07990* (2024).

# Merging Algorithms - TODO

Reference Repo: [acree-ai mergekit](#) (Recommend)

- Implement SCE in *peft*

```python
@merge_method(
    name="sce",
    pretty_name="SCE",
    reference_url="https://arxiv.org/abs/2408.07990",
)
def sce_merge(
    tensors: List[torch.Tensor],
    base_tensor: torch.Tensor,
    int8_mask: bool = False,
    select_topk: float = 1.0,
) -> torch.Tensor:
    if not tensors:
        return base_tensor
    mask_dtype = torch.int8 if int8_mask else base_tensor.dtype
    task_vectors = torch.stack([t - base_tensor for t in tensors], dim=0)

    if select_topk < 1:
        mask = sce_mask(task_vectors, select_topk, mask_dtype)
        task_vectors = task_vectors * mask.unsqueeze(0)

    erase_mask = sign_consensus_mask(task_vectors, method="sum", mask_dtype=mask_dtype)

    tv_weights = sce_weight(task_vectors)
```

# Merging Algorithms - TODO

- Additional functions for implementing SCE Algorithms

```
def sce_weight(task_tensors: torch.Tensor) -> torch.Tensor:
    # Implementation of C step
    # Compute squared magnitude (energy) per task
   weights = torch.mean(task_tensors**2, dim=list(range(1, task_tensors.dim())))
    # Sum all weights to normalize
   weight_sum = torch.sum(weights).item()
    # Handle edge case: if all task tensors are 0, fallback to uniform weights
   if abs(weight_sum) < 1e-6:
       return torch.ones_like(weights) / weights.shape[0]
    # Normalize to form a probability distribution over tasks
   return weights / weight_sum
```

Ref 1: acree-ai mergekit (Recommend)
Ref 2: FuseChat mergekit

# Merging Algorithms - TODO

```
def sce_mask(task_tensors: torch.Tensor, density: float, mask_dtype: Optional[torch.dtype] = None):
    # Implementation of S step (sce_mask)
    if density <= 0: # If density is zero, mask out everything
        return torch.zeros_like(task_tensors, dtype=mask_dtype)
    if density >= 1: # If density is one, keep everything
        return torch.ones_like(task_tensors, dtype=mask_dtype)
    var = torch.var(task_tensors, dim=0, unbiased=False) # Compute variance over the task dimension (T) for
each parameter
    nonzero = torch.count_nonzero(var) # Count how many parameters have non-zero variance
    k = int(nonzero * density) # Compute number of parameters to keep based on density
    if k == 0:
        return torch.zeros_like(task_tensors, dtype=mask_dtype)
    _, indices = torch.topk(var.abs().view(-1), k=k, largest=True) # Select the indices of top-k variances
    # Build binary mask with 1s in selected indices
    mask = torch.zeros_like(var, dtype=mask_dtype)
    mask.view(-1)[indices] = 1
    return mask
```

# 2025/05/21 Update: TODO - download and modify peft package

- Download and modify TA-version peft package; either
  - Use terminal commands to download and unzip the *peft* package. After making your modifications on Colab or Kaggle, install the package in editable mode ( `pip install -e .`) so that the modified version can be used directly on Colab/Kaggle.
  - Download and modify the peft package on your local machine, then upload the modified version to Google Drive. After that, install it on Colab/Kaggle to use the updated package.
- Google Drive Links: link1, link2, link3, link4

- Either modify existing algorithms or add new algorithms into peft package
- modules to be modified
  - add functions to include your own merging methods `peft/src/peft/utils/merge_utils.py`
  - add combination_type: `peft/src/peft/tuners/lora/model.py LoraModel.add_weighted_adapter()`

# TODO - how to modify peft package

in peft/src/peft/utils/merge_utils.py:

```
def ties(
    task_tensors: List[torch.Tensor],
    weights: torch.Tensor,
    density: float,
    majority_sign_method: Literal["total", "frequency"] = "total",
) -> torch.Tensor:
      ...
      # sparsify
      # Elect Sign
      # weighted task tensors
      # Disjoint Merge
      return...

#### todo: Add new methods, reuse modules in other algorithms ####
#### e.g. if you want to implement “sce” algorithm ####

def sce(task_tensors, density, majority_sign_method) -> torch.Tensor:
      ...
      return ...
```

# TODO - how to modify peft pachage

in peft/src/peft/tuners/lora/model.py:

```
#### todo: import function of new methods here ####
from peft.utils.merge_utils import magnitude_prune, ties

class LoraModel:
      def add_weighted_adapter(self, adapters, weights, adapter_name, combination_type, density,
      majority_sign_method):
            ...
            if ...
                  ...
            #### todo: remember to add func names of new methods here ####
            elif combination_type in ["linear", "ties", "dare_linear", "dare_ties", "magnitude_prune"]:
                  ...

      def _generalized_task_arithmetic_weighted_adapter(self, combination_type, adapters, weights, target,
      density, majority_sign_method):
            ...
            #### todo: remember to add corresponding combination_type to call functions here ####
            elif combination_type == "ties":
                lora_deltas[i] = ties(task_tensors, valid_weights, density, majority_sign_method)
```

# 2025/05/21 Update: TODO - Install modified peft on Colab/Kaggle

- Install customized peft package in editable mode on colab notebook

On colab/kaggle

```
%cd /content/drive/MyDrive/ml2025_hw9/peft-ml2025-hw9 #peft package path
!pip install -e . # install modified package in editable mode
# add src directory to system path
%cd /content
import sys
sys.path.append("/content/drive/MyDrive/ml2025_hw9/peft-ml2025-hw9/src")
```

- remember to add new if/else conditions in the sample code to merge weights with new algorithms in peft package

# TODO - experiment with different algorithms

- merge in possible (weights, density) pairs and inference on two tasks
- modify generation config
- select the optimal results, save 400 {“id”: “response”} pairs to a json file and submit to Judgeboi
    - e.g. {{“arc_1”: “Therefore, option (A) is the correct answer.”}, ...}

Estimate inference time: 2~4 hr /400 samples (Colab T4)

# Hints

- Experiment with or without some steps in an algorithm to understand which step plays more important role in merging.
- "Pruning" sometimes mitigate parameter interference effectively, but this condition may change in other algorithms.
- Modify GenerationConfig (hyperparamter tuning) (ref 1, ref 2, hw5 ppt)
  - decoding strategy: greedy decoding (`do_sample=None`), temperature, top-k, top-p, beam search(`num_beam > 0`)
  - max length of generation: `max_new_len` (also affect inference time)
- Print and observe the generated responses during inference to assess whether a new merging configuration might lead to improved performance.

# Hints

- Possible reasons for long inference (> 4 hr):
  - `max_new_length` (default: 400)
  - beam search (when `num_beam` > 1)
  - degeneration (repetition, illogical or redundant texts) because of strong parameter interference

[Reminder!] For arithmetic reasoning tasks like GSM8K, a step-by-step reasoning process is essential. The correctness of intermediate steps plays a critical role, as any logical or computational mistake along the way can ultimately result in an incorrect final answer.

[Suggestion!] Pay particular attention to the consistency between intermediate reasoning steps and the final selected option. (If the predicted answer doesn't exactly match one of the provided choices, model may choose the closest matching answer rather than the correct one.)

# Grading and Submission

# Grading and Submission - Baselines & Grading

| | ARC Acc. | GSM8K Acc. | Score | Hint | |
|---|---|---|---|---|---|
| **Public Simple Baseline** | **≥ 49%** | **≥ 38%** | +1 | Task Arithmetic, Magnitude Prune, TIES, DARE, SCE | **Estimated Inference Time**<br><br>**2~4hr /400q** |
| **Private Simple Baseline** | | | +1 | | |
| **Public Medium Baseline** | **≥ 53%** | **≥ 42%** | +1 | | |
| **Private Medium Baseline** | | | +1 | | |
| **Public Strong Baseline** | **≥ 56%** | **≥ 48%** | +1 | | |
| **Private Strong Baseline** | | | +1 | | |
| **Code Submission** | - | - | +4 | | |

# Grading and Submission - Judgeboi

- Please submit the pred.json to judgeboi. (only .json file is allowed)
- The prediction file (pred.json) must follow the below structure:
  - Root must be a dictionary ({}).
  - Each key must be a string representing an ID (e.g., "arc_1", "gsm8k_32").
  - Each value must be a string containing the model-generated response without input prompt.

- **5** submission quota per day, reset at 23:59 (UTC+8).
- Each submission takes about a minimum of 6~7 minutes to evaluate.

# Grading and Submission - NTU COOL

submit before deadline: **2025/06/06 23:59:59 (UTC+8)**.

No late submission is allowed.

- Submit your code to NTU COOL. (4 points)
  - Please remember to submit all the .py files you have modified or added in the peft package. These files will be used by the TAs to reproduce your code (merging settings). During the reproduction process, the TA will replace the corresponding files in the original TA version of the peft package with the ones you submitted.
  - Remember to submit the main .ipynb file or .py scripts that cover the complete inference process.
  - DO NOT INCLUDE YOUR PRIVATE TOKEN IN YOUR SUBMISSION.
  - You need to provide a README, regardless of the program execution environment.
  - In the README, you must specify the absolute path (peft package) of modified files. This is to help the TAs correctly replace and overwrite the corresponding files during the reproduction process.
  - We can only see your last submission.
  - Compress your code into _hw9.zip. (e.g. b13901001_hw9.zip)
  - After TAs unzip your _hw9.zip, all your files should locate under a directory called _hw9.
  - All the English alphabets in your student ID should be in lowercase.

# Grading and Submission - NTU COOL

- How to write a README?
  - For the submitted files related to the peft package, clearly specify its absolute path within the peft package, e.g. “b13901001_hw9_1.py: /peft/src/peft/utils/merge_utils.py”
  - Specify your environment(colab, kaggle...) and GPU(T4, T4*2, P100...).
  - List all references used to finish the homework.
    - Which part of code is generated by which model(GPT, Gemini, Grok...). Shared link for the chat is better.
    - Website link, NTU Cool discussion, Offline discussion with classmates(Student IDs)...
  - If you run the code in your environment instead of colab or kaggle.
    - Specify the python version.
    - Provide a requirements.txt for additional installed packages.
  - If you decompose sample code into multiple scripts.
    - Specify the function of each file.
    - Provide a step-by-step instruction for running your scripts with correct commands and execution order.
  - If you have no idea.
    - Ask README Generator.

# Grading and Submission - NTU COOL

- README Example
  - To assist TAs in automatically reproducing your results, please clearly list any files you have modified or added within the PEFT package.
  - Provide a code block tagged as `replace` with the following format:
    ```replace <your_filename.py>: <absolute_path_in_peft> ```

````
# In README.md
## Main file
```main
b13901001_hw9_1.ipynb
```

## PEFT Package Modification for TA Reproduction
```replace
b13901001_hw9_2.py: /peft/src/peft/utils/merge_utils.py
b13901001_hw9_3.py: /peft/src/peft/tuners/lora/model.py
```
````
````

# Grading and Submission - NTU COOL

- Structure of the zipped file:
  - _hw9
    - _hw9_1.ipynb or .py
    - _hw9_2.ipynb or .py
    - ...
    - README.md
- Examples for valid structure of the zipped file:
  - b13901001_hw9
    - b13901001_hw9_1.ipynb
    - b13901001_hw9_2.py
    - b13901001_hw9_3.py
    - README.md

# Grading and Submission - NTU COOL

- If your code is not reasonable or reproducible, you will receive 0 points for this homework.
- Deadline: **2025/06/06 (Fri.) 23:59 (UTC +8)**

# Regulations

- Do NOT share codes or prediction files with any living creatures.
- Do NOT use any approaches to submit your results more than 5 times a day.
- During the inference of 400 questions, you must use the same merged model. (consistent merging setting, any cheating is not allowed)
- You are NOT allowed to fine-tune your own ckpts of two tasks in this homework.
- You are NOT allowed to modify instructions, questions and options (prompts) in HW9.
- All refining code should also be included in the code submission, involving all modified files/modules in your peft package.
- You should NOT modify your input file or prediction files manually.
- Do NOT search for the answers for the inference data.
- Make sure that TAs can reproduce the predictions using the code you submit.
- Please protect your own work and ensure that your answers are not accessible to others. If your work is found to have been copied by others, you will be subject to the same penalties.
- You will receive 0 points for this homework if you violate any of the above rules.
- Prof. Lee & TAs preserve the rights to change the rules & grades.

# If any questions, you can ask us via...

- NTU COOL (recommended)
- Email
  ntu-ml-2025-spring-ta@googlegroups.com
  The title should begin with “[hw9]”
- TA hours
  Each Friday During Class
  Time : 13:30 - 14:10 ; 17:30 - 18:00

# Reference

# Reference

base model: https://huggingface.co/unsloth/llama-2-7b-chat-bnb-4bit

PEFT related:
https://huggingface.co/docs/peft/index,
https://huggingface.co/docs/peft/developer_guides/model_merging

papers of Merging Algorithms: Task Arithmetic, TIES, DARE, SCE

Mergitkit: acree-ai mergekit, FuseChat mergekit