A Details for Experiments

The models are implemented through the 'transformers' package developed by Wolf et al. [15]. When using Switch Transformer for the GLUE classification tasks, we add a classification head according to the setting of T5.

A.1 Details of Experiment Settings

In the zero-shot context, we adhere to the standard practice for zero-shot LLM evaluation. This involves assessing the pre-trained LLM on various downstream tasks without any additional fine-tuning. In the fine-tuning scenario, we employ the typical procedure of initially fine-tuning the pre-trained model on downstream tasks, followed by evaluation during the inference stage. Our approach is concentrated on enhancing the inference stage, ensuring that after the implementation of our methods, no further fine-tuning is required, and the compressed model will be ready-to-use.

Table 1: Fine-tuning hyper-parameters setting for Switch Transformer.

	Value
Optimizer	AdamW
Adam ϵ	1e-08
Adam β	(0.9, 0.98)
warm-up steps	8
weight decay	0.01

For Mixtral, we adhere to the model's configuration card, as no hyperparameter tuning is required for zero-shot tasks. For Switch Transformer, we employ AdamW optimizer with a linear warm-up step count of 8. We explore the hyper-parameters settings during the supervised fine-tuning stage. For Switch Transformer, the learning rate is searched in the range of {1e-5,2e-5,3e-5,5e-5,1e-4,2e-4,3e-4,5e-4,1e-3}, the batch size within the range of {16,32,64}, and the training epoch within the range of {3,5,10,15,20}. The details of the AdamW optimizer which is fixed for all datasets are given in Table 1.

To ensure comparability across methods, we standardize the parameter count reduction for the experts to around 75%, which means 25% of the parameters will be retained. All the methods are performed at the top 24 layers of Mixtral, and the top 8 MoE layers of Switch Transformer. All the methods are applied during the inference stage.

A.2 Experiment Results of DeepSeekMoE

DeepSeekMoE [4] features more fine-grained experts compared to alternative structures, with each expert sized at just 8.7M, markedly smaller than Mixtral's 176.2M. It introduces a unique, independent shared expert atop each MoE layer, aiming to encapsulate universal information across layers. This approach is inspired by observations from Mixtral's router analysis, which indicates a roughly equal routing probability for each expert during the inference stage, suggesting the presence of universal knowledge within each expert. Consequently, an additional shared expert is integrated, anticipated to store universal information and thereby enhance the diversity of the remaining experts. In our experiments with DeepSeekMoE, we exclude this shared expert, considering its anticipated information content significantly differs from that of the non-shared experts.

Table 2 provides the results for DeepSeekMoE. Note that the results for LAMBADA dataset are not included here, since DeepSeekMoE produces extremely bad results for this dataset (0.04% Accuracy). Even though the experts of DeepSeekMoE are also initialized through copy-and-paste, the existence of the shared expert distinguishes those experts in the MoE layers from each other, leading to the poor performance of the merge methods, which aligns with our proposition that merge methods will potentially impair the generalization ability of the original model.

Table 2: Zero-shot results of DeepSeekMoE. For Pruning, we choose Unstructured Pruning over Structured Pruning based on the observations from the previous experiments that Unstructured Pruning usually has better results on NLG tasks.

	WikiText (PPL)↓	PIQA (ACC)	WinoGrande (ACC)
DeepSeekMoE	6.51	78.84	68.75
Pruning	10.46	73.12	62.83
SVD	26.93	63.06	57.46
M-SMoE	34.76	62.79	54.22
MEO	33.94	62.35	54.14
ResMoE (UP)	**10.39±0.10**	**73.39±0.01**	**64.35±0.01**

A.3 Compression Setting for Each Method

In order to match our setting of the 75% compression rate, each method has to be tailored differently. For Pruning, we mask 75% weights units with the lowest L1-norm within each expert. For SVD, the details of calculating the rate of the parameters can be referred to in Appendix A.4. For M-SMoE, Git Re-Basin, and MEO, we reduce the expert count of each MoE layer from 8 to 2. For MLP Fusion, we reduce the intermediate dimension to 25%. For ResMoE, we mask 75% weights units with the lowest L1-norm within each residual matrix. We do not count the overhead storage of the barycenter experts here since we aim to prove the effectiveness of our algorithm, and as the number of experts grows, the redundancy of this overhead will diminish. We provide additional experiments performed at DeepSeekMoE (64 experts per layer) [4] as a support, which can be found in Appendix A.2.

A.4 The Parameter Count of SVD

For SVD, to make the the parameters retained for each expert equal, for Switch Transformer we have:

$$p_{\mathrm{I}} \times k + k + k \times 2p \approx k \times (p_{\mathrm{I}} + 2p)$$
$$s \times p_{\mathrm{I}} \times 2p = 2spp_{\mathrm{I}},$$

where s is the parameter rate we retain (25% here), and k is the number of top-k singular values in SVD. For Switch Transformer, we have $p_{\mathrm{I}} = 4p$, so $k = \frac{1}{3}sp_{\mathrm{I}}$.

For Mixtral:

$$p_{\mathrm{I}} \times k + k + k \times 3p \approx k \times (p_{\mathrm{I}} + 3p)$$
$$s \times p_{\mathrm{I}} \times 3p = 3spp_{\mathrm{I}},$$

here we have $p_{\mathrm{I}} = 3.5p$, so $k = \frac{6}{13}sp_{\mathrm{I}}$.

For DeepSeekMoE:

$$p_{\mathrm{I}} \times k + k + k \times 3p \approx k \times (p_{\mathrm{I}} + 3p)$$
$$s \times p_{\mathrm{I}} \times 3p = 3spp_{\mathrm{I}},$$

here we have $p_{\mathrm{I}} = \frac{11}{16}p$, so $k = \frac{48}{59}sp_{\mathrm{I}}$.

A.5 Evaluation of Approximation Error for MLP Fusion

We first reformulate MLP fusion [1] with the notations in this paper. In computing the fused MLP for expert k, we obtain the centroid weight/bias $\widetilde{\mathbf{W}}_k = \left[\widetilde{\mathbf{W}}_k^{(1)}, \tilde{\mathbf{b}}_k^{(1)}, (\widetilde{\mathbf{W}}_k^{(2)})^T\right] \in \mathbb{R}^{c\times(2p+1)}$, as well as the one-hot clustering matrix $\mathbf{C}_k \in \mathbb{R}^{c\times p_{\mathrm{I}}}$ indicating how the p_{I} neurons are partitioned into c clusters. MLP fusion then proposes to compute

$$\widetilde{\mathbf{W}}_k^{(2)}\left(\mathbf{C}_k\mathbf{C}_k^T\right)\sigma\left(\widetilde{\mathbf{W}}_k^{(1)}\mathbf{x} + \tilde{\mathbf{b}}_k^{(1)}\right) + \mathbf{b}_k^{(2)}$$

as the approximation to the original expert k.

Ai et al. [1] suggest the expression above is equivalent to replacing $\mathbf{W}_k = \left[\mathbf{W}_k^{(1)}, \mathbf{b}_k^{(1)}, (\mathbf{W}_k^{(2)})^T\right] \in \mathbb{R}^{p_{\mathrm{I}}\times(2p+1)}$ by $\mathbf{C}_k^T\widetilde{\mathbf{W}}_k$, considering

$$\begin{aligned}&\widetilde{\mathbf{W}}_k^{(2)}\left(\mathbf{C}_k\mathbf{C}_k^T\right)\sigma\left(\widetilde{\mathbf{W}}_k^{(1)}\mathbf{x} + \tilde{\mathbf{b}}_k^{(1)}\right) + \mathbf{b}_k^{(2)}\\ =&\left(\widetilde{\mathbf{W}}_k^{(2)}\mathbf{C}_k\right)\sigma\left(\mathbf{C}_k^T\widetilde{\mathbf{W}}_k^{(1)}\mathbf{x} + \mathbf{C}_k^T\tilde{\mathbf{b}}_k^{(1)}\right) + \mathbf{b}_k^{(2)}.\end{aligned}$$

We can thus calculate $\left\|\mathbf{W}_k - \mathbf{C}_k^T\widetilde{\mathbf{W}}_k\right\|_F^2$ as the approximation error on expert k for MLP fusion.

A.6 Details of the Datasets

We provide the details of the datasets we used in the experiment along with their license here. The statistics can be found in Tables 3 and 4.

- PIQA [2]: PIQA, or Physical Interaction Question Answering, is a benchmark dataset that assesses AI systems' commonsense reasoning abilities regarding physical knowledge. It challenges models with multiple-choice questions related to everyday physical interactions, testing their understanding of object manipulation and functionality in real-world scenarios. While humans perform well on PIQA, it presents a significant challenge to AI models, making it crucial for advancing AI research, especially in robotics and conversational AI. PIQA is licensed under Academic Free License v3.0.

Table 3: Dataset statistics of fine-tuned classification tasks.

Dataset	Category	Train size	Test Size	Classes
SST-2	Sentiment Analysis	67,349	872	2
MRPC	Paraphrase Identification	3,668	408	2
CoLA	Linguistic Acceptability Judgment	8,551	1,043	2
MNLI	Textual Entailment	392,702	9,815	3

Table 4: Dataset statistics of zero-shot tasks.

Dataset	Category	Test Size	Average Text Length
PIQA	Commonsense Reasoning	1,838	36.08
WikiText	Language Modeling	4,358	295.00
WinoGrande	Commonsense Reasoning	1,267	100.78
LAMBADA	Text Understanding	4,896	341.72

- WikiText[7]: WikiText-103 is a widely-used dataset in Natural Language Processing, ideal for language modeling and text generation tasks. Derived from verified Wikipedia articles, it offers over 100 million tokens of well-structured text, preserving original formatting. Its extensive vocabulary and varied syntax make it a valuable resource for training advanced language models. WikiText-103 serves as a crucial benchmark for evaluating language models' performance in handling real-world textual data, with a license of CC BY-SA 3.0.
- WinoGrande[10]: Winogrande, an extension of the Winograd Schema Challenge, consists of ambiguous sentence pairs requiring deep language understanding and commonsense reasoning to resolve. It aims to overcome limitations in previous datasets and assesses AI models' ability to comprehend nuanced language and context, making it vital for advancing natural language understanding. Winogrande is licensed under CC-BY.
- LAMBADA[8]: LAMBADA is a challenging benchmark designed for evaluating computational models' language understanding, focusing on predicting the final word in a passage. It requires models to grasp broad context and long-range dependencies within text passages. LAMBADA pushes the boundaries of language models, particularly in handling complex, context-dependent linguistic phenomena, making it a valuable tool for advancing natural language processing. It is licensed under CC BY 4.0.
- SST-2[13]: SST-2, the Stanford Sentiment Treebank version 2, is a popular dataset for sentiment analysis. It contains movie review sentences labeled as positive or negative, excluding neutral sentences, providing a binary classification task. This dataset is notable for its fine-grained annotation, as it includes sentiment labels for every subphase within the sentence parse trees. SST-2 is widely used for training and evaluating models on sentiment analysis, testing their ability to understand nuanced emotional tones in text, with the license of CC0: Public Domain.
- MRPC[5]: The Microsoft Research Paraphrase Corpus (MRPC) evaluates models on paraphrase identification by using sentence pairs from online news sources. MRPC is a part of the GLUE benchmark and is valuable for assessing a model's ability to understand and compare semantic content in sentences, especially in semantic analysis tasks. The license of MRPC is unknown.
- CoLA[14]: The Corpus of Linguistic Acceptability (CoLA) assesses models' linguistic acceptability judgment. It distinguishes between grammatically acceptable and unacceptable sentences, emphasizing the importance of grammatical understanding in language comprehension and model evaluation. The license for CoLA is not specified.
- MNLI[14]: The Multi-Genre Natural Language Inference (MNLI) dataset is a diverse corpus for natural language understanding tasks, focusing on textual entailment. It includes pairs of sentences and challenges models to determine whether the second sentence entails, contradicts, or remains neutral to the first sentence. MNLI's wide range of genres and diverse content makes it a robust benchmark for evaluating models in natural language inference tasks. Most of the data are under the OANC's license, with the other falling under several permissive licenses, a Creative Commons Share-Alike 3.0 Unported License, and Creative Commons Attribution 3.0 Unported Licenses.

A.7 Implementation Trick

In our application of unstructured pruning with ResMoE, a key consideration is its impact on memory storage. The Pytorch version we use supports only the Coordinate format (COO) for sparse matrices, which necessitates storing indices as int64. For instance, in an MLP of Mixtral, the original memory footprint is 672MB. However, a version pruned to 75% sparsity consumes more memory, around 840MB, with 672MB used just for storing indices. If the indices could be stored as int16, the memory requirement for the indices would be reduced to 168MB, thus the memory of the entire MLP unit would be significantly reduced to 336MB. Furthermore, if the index is stored in the format of CSR instead of COO, the memory size can be reduced to 252MB. Although addressing this limitation falls outside our current scope, we aim to explore solutions to this challenge in future work.

Table 5: Memory usage of one MoE layer in Mixtral & DeepSeekMoE (MB).

	Mixtral	DeepSeekMoE
Full	5,376	2,112
UP	2,016	1,056
SP	1,344	528
SVD	1,344	528
M-SMoE	1,344	528
Git Re-Basin	1,344	528
MEO	1,344	528
MLP Fusion	1,344	528
ResMoE (UP)	2,688	1,089
ResMoE (SVD)	2,016	561

Table 6: Runtime of Mixtral on Winogrande.

	Runtime (s)
Mixtral	39.44±0.30
UP	39.01±0.21
SP	37.15±0.22
SVD	38.96±0.31
M-SMoE	49.19±0.12
Git Re-Basin	48.53±0.09
MEO	49.51±0.18
MLP Fusion	38.53±0.26
ResMoE (UP)	38.85±0.28
ResMoE (SVD)	38.12±0.19

On the other hand, when choosing SVD as the compression method, it will be able to directly reduce memory usage since SVD will reduce the size of each matrix. We remark that even though utilizing SVD here leads to slightly worse results than unstructured pruning, it still performs better compared to the baseline methods. Also, when the number of experts goes up, the overhead introduced by the center expert diminishes.

Based on such settings, we provide the memory information on Mixtral (8 experts per layer) and DeepSeekMoE (64 experts per layer) in Table 5 with the overhead center expert included. Also, when the number of experts goes up, the overhead memory introduced by the center expert diminishes.

A.8 Efficiency Evaluation

In addition to the memory information, we also provide the evaluation of runtime and FLOPs [3].

The runtime in Table 6 is obtained by testing on 2 A100 GPUs on the WinoGrande Dataset with a batch size of 64. It is worth noting that the runtime of the merged methods is even slower compared to the original Mixtral model. This is likely due to the code we referred from [6], which only creates references from the experts that are merged but does not exactly reduce the number of experts in the model. Note that the sparse matrices induced by unstructured pruning are stored as normal matrices instead of sparse matrices here. We could observe that ResMoE does not influence the time complexity while reducing the memory.

For the FLOPs evaluation, as shown in Table 7, structured pruning and MLP Fusion, which reduce the intermediate dimension of weight matrices, lead to a significant reduction in FLOPs. It is also important to note although unstructured pruning can reduce FLOPs, it does not reduce space storage as ResMoE does as detailed in Appendix A.7. In this regard, ResMoE (UP) and the merge methods maintain similar FLOPs compared to the original Mixtral. ResMoE (SVD) has more FLOPs compared to vanilla SVD due to the extra FLOPs brought in with the center expert, but it still manages to reduce the FLOPs compared to the original model.

A.9 Pseudocode of ResMoE

In considering of reproducibility, we provide the algorithm to perform ResMoE on a model in Algorithm 1, and the dynamic load inference process in Algorithm 2. Please find the source code at https://github.com/iDEA-iSAIL-Lab-UIUC/ResMoE.

Table 7: FLOPs evaluation of Mixtral & DeepSeekMoE.

	Mixtral (TFLOPs)	DeepSeekMoE (GFLOPs)
Full	3.26	670.46
UP	1.64	460.24
SP	1.64	460.24
SVD	2.21	480.52
M-SMoE	3.26	670.46
Git Re-Basin	3.26	670.46
MEO	3.26	670.46
MLP Fusion	1.64	460.24
ResMoE (UP)	3.26	670.46
ResMoE (SVD)	2.73	526.93

Algorithm 1 ResMoE

```
Input: experts weights $E_1, \dots, E_n$, sparsity ratio $r$
Output: center expert $C$, compressed residuals $R_1, \dots, R_n$
// Step 1: Calculate the center expert using free-support Wasserstein barycenter
$C \leftarrow$ free_support_wasserstein_barycenter$(E_1, \dots, E_n)$
// Step 2: Calculate the residual matrices
for $i = 1$ to $n$ do
    $R_i \leftarrow E_i - C$
end for
// Step 3: Apply compression technique to residual matrices with sparsity $r$
for $i = 1$ to $n$ do
    $R_i \leftarrow$ compress$(R_i, r)$
end for
return $C, R_1, \dots, R_n$
```

Algorithm 2 Inference

```
Input: input $x$, center expert $C$, pruned residuals $R_1, \dots, R_n$, selected experts subscript set $S$
Output: inference result $y$
// Step 1: Reconstruct and dynamically load the compressed experts
$E_S \leftarrow \emptyset$
for $R_i \in R_S$ do
    $E_i \leftarrow C + R_i$
    $E_S \leftarrow E_S \cup E_i$
end for
// Step 2: Perform inference using the recovered experts
$y \leftarrow$ perform_inference$(x, E_S)$
return $y$
```

B Miscellanies

B.1 Adaptability with Expert Parallelism and Tensor Parallelism

While the primary focus of this paper is on the inference stage and not on saving memory usage during training, we acknowledge that expert parallelism is an important consideration for the scalability and efficiency of MoE models.

One feasible approach is to assign different center experts to each GPU, allowing each center expert to handle the experts on its respective GPU during inference. This extension of ResMoE could potentially improve the model's performance by capturing more diverse and complex patterns in the data.

ResMoE is also compatible with tensor parallelism. As shown in Equation (3), the output of an MLP in an MoE model can be expressed as the summation of several sub-MLPs. In the context of ResMoE, we can view each sub-MLP as the combination of a center expert and a compressed residual matrix. To utilize Megatron tensor sharding [11], we can partition the center expert and compressed residual matrices into different chunks, with corresponding index ranges residing on different GPUs.

During inference, the input data would be parallelized and passed to the appropriate center expert and residual matrix chunks on each GPU. The partial results from each GPU would then be combined to obtain the final output. This parallelization strategy aligns well with the Megatron tensor sharding approach, which aims to distribute the computation across multiple GPUs to improve efficiency and scalability.

B.2 Illustration of Model Fusion

In the work of OT Fusion [12], for the first layer in an two-layer MLP (for example), their algorithm takes the weights $(\mathbf{W}_k^{(1)}, \mathbf{b}_k^{(1)})$ in each expert E_k and in E_ω as the source distributions and the target distribution, respectively. They directly take $\mathbf{W}_k^{(1)}$ as the design points (empirical distributions) and then regard their free-support Wasserstein barycenter as the common pattern extraction of the first layer in each expert, returning $\mathbf{W}_\omega^{(1)}$ and the corresponding permutation matrix $\mathbf{T}_k^{(1)}$'s (obtained from the transport matrices) for $\mathbf{W}_k^{(1)}$. Accordingly, they then pre-align the second layers as $\mathbf{W}_k^{(2)}\mathbf{T}_k^{(1)}$, repeat the procedure above and similarly obtain the extracted layer $\mathbf{W}_\omega^{(2)}$ and the permutation matrices $\mathbf{T}_k^{(2)}$. To recover the k-th expert E_k, their barycenter expert needs to be transformed as $(\mathbf{T}_k^{(2)})^\mathrm{T} E_\omega(x)$, to fix the order of the output elements. We remark this, along with pre-alignment $\mathbf{W}_k^{(2)}\mathbf{T}_k^{(1)}$ bring overhead during algorithm performing stage. This is supported by the process of applying OT fusion to calculate the barycenter expert on Mixtral, which takes more than 4 days to complete the whole process, while ResMoEonly takes less than a day. A possible explanation is that Mixtral's MLP consists of three layers, so the layer-by-layer strategy costs about three times more than our method.

B.3 The MLP Form for Mixtral and DeepSeekMoE

The output of an MLP in Mixtral and DeepseekMoE can be rewritten into:

$$\begin{aligned}
E_k(\mathbf{x}) &= \mathbf{W}_k^{(2)} \cdot \mathrm{SwiGLU}(\mathbf{x}) + \mathbf{b}_k^{(2)} \\
&= \mathbf{W}_k^{(2)} \cdot \left[\sigma\left(\mathbf{W}_k^{(1)} \cdot \mathbf{x} + \mathbf{b}_k^{(1)}\right) \odot \left(\mathbf{W}_k^{(3)} \cdot \mathbf{x} + \mathbf{b}_k^{(3)}\right)\right] + \mathbf{b}_k^{(2)} \\
&= \sum_{i=1}^{p_I} \mathbf{W}_{k,\cdot,i}^{(2)} \cdot \left[\sigma\left(\left\langle \mathbf{W}_{k,i,\cdot}^{(1)}, \mathbf{x}\right\rangle + \mathbf{b}_{k,i}^{(1)}\right) \cdot \left(\left\langle \mathbf{W}_{k,i,\cdot}^{(3)}, \mathbf{x}\right\rangle + \mathbf{b}_{k,i}^{(3)}\right)\right] + \mathbf{b}_k^{(2)},
\end{aligned}$$

where $\sigma(\mathbf{x}) = \mathrm{Swish}_\beta(\mathbf{x}) = \mathbf{x}\sigma(\beta\mathbf{x}) = \dfrac{\mathbf{x}}{1 + e^{-\beta \mathbf{x}}}$, with β setting to 1.

Similarly, for Mixtral and DeepSeekMoE, the extraction objective is:

$$\begin{aligned}
\min_{\substack{\mathbf{W}_\omega^{(1)}, \mathbf{b}_\omega^{(1)}, \mathbf{W}_\omega^{(3)}, \mathbf{b}_\omega^{(3)}, \mathbf{W}_\omega^{(2)} \\ \mathbf{T}_k \in \mathcal{P}, k \in [N]}} \frac{1}{N} \sum_{k=1}^{N} \Bigg[\Big\| \mathbf{T}_k \left(\mathbf{W}_k^{(1)}, \mathbf{b}_k^{(1)}, \mathbf{W}_k^{(3)}, \mathbf{b}_k^{(3)}\right) \\
- \left(\mathbf{W}_\omega^{(1)}, \mathbf{b}_\omega^{(1)}, \mathbf{W}_\omega^{(3)}, \mathbf{b}_\omega^{(3)}\right) \Big\|_F^2 \\
+ \left\| \mathbf{W}_k^{(2)} \mathbf{T}_k^\mathrm{T} - \mathbf{W}_\omega^{(2)} \right\|_F^2 \Bigg].
\end{aligned}$$

Then we can extend the $\mathbf{W}_k$ to:

$$\mathbf{W}_k = \left[\mathbf{W}_k^{(1)}, \mathbf{b}_k^{(1)}, \mathbf{W}_k^{(3)}, \mathbf{b}_k^{(3)}, (\mathbf{W}_k^{(2)})^\mathrm{T}\right],$$

and ResMoE can then be similary applied to Mixtral and DeepSeekMoE.

C Proof of Proposition 4.1

For the reader's convenience, we recall Proposition 4.1 as follows.

Proposition C.1. *Consider the solution $\mathbf{W}_\omega$ to the following free-support WB problem*

$$\min_{\mathbf{W}_\omega} \frac{1}{N} \sum_{k=1}^{N} W_2^2(\mu_k, \mu_\omega(\mathbf{W}_\omega)). \tag{1}$$

Then $\mathbf{W}_\omega$, along with $\mathbf{T}_k = p_\mathrm{I} \cdot \mathrm{OT}\left(\mu_k, \mu_\omega(\mathbf{W}_\omega)\right)$, is the solution to the optimization problem (4).

Proof. We recall μ_i, μ_ω are uniformly distributed over the p_I rows of $\mathbf{W}_i$ and $\mathbf{W}_\omega$, respectively. $\mathrm{OT}\left(\mu_i, \mu_\omega(\mathbf{W}_\omega)\right)$ is the optimal transport matrix $\mathbf{M}$ from μ_i to μ_ω of the following problem:

$$\mathrm{OT}(\mu, \nu) := \operatorname*{argmin}_{\mathbf{M} \in U(\alpha, \beta)} \langle \mathbf{M}, \mathbf{C} \rangle, \tag{2}$$

where $\mathbf{C} = \big[\|\mathbf{W}_{k_i} - \mathbf{W}_{\omega_j})\|^2\big]_{ij} \in \mathbb{R}^{p_I \times p_I}, U(\frac{1_{p_I}}{p_I}, \frac{1_{p_I}}{p_I}) := \{\mathbf{M} \in \mathbb{R}_+^{p_I \times p_I} \mid \mathbf{M}\mathbf{1}_{p_I} = \frac{1_{p_I}}{p_I}, \mathbf{M}^T\mathbf{1}_{p_I} = \frac{1_{p_I}}{p_I}\}$.

We first relate the transport matrix to the permutation matrices $\mathbf{T}_k$'s. Peyre and Cuturi [9, Proposition 2.1] shows the optimal solution to problem (2) is exactly a permutation matrix, up to a constant factor p_I. Now straightforwardly, problem (4) can be rewritten as:

$$\min_{\substack{\mathbf{W}_\omega \\ \mathbf{T}_k \in P, k \in [N]}} \frac{1}{N} \sum_{k=1}^{N} \Big[\|\mathbf{T}_k \mathbf{W}_k - \mathbf{W}_\omega\|_F^2 \Big], \tag{3}$$

where $\mathbf{W}_k = [\mathbf{W}_k^{(1)}, \mathbf{b}_k^{(1)}, (\mathbf{W}_k^{(2)})^T] \in \mathbb{R}^{p_I \times (2p+1)}$, and $\mathbf{W}_\omega = [\mathbf{W}_\omega^{(1)}, \mathbf{b}_\omega^{(1)}, (\mathbf{W}_\omega^{(2)})^T] \in \mathbb{R}^{p_I \times (2p+1)}$.

We denote the objective function in problem (3) as $f(\mathbf{W}_\omega; \{\mathbf{T}_k\}_{k=1}^N) = \sum_{k=1}^N \frac{1}{N}[\|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega\|_F^2]$, and take $\mathbf{W}_\omega^*$ as the optimal solution to the Wasserstein barycenter problem (1). For the given $\mathbf{W}_\omega^*$, we further denote $\mathbf{T}_k^* := \underset{\mathbf{T}_k}{\operatorname{argmin}} f(\mathbf{W}_w^*; \mathbf{T}_k), \forall k \in [N]$. The rest of the proof is to show $f(\mathbf{W}_\omega^*; \big\{\mathbf{T}_k^*\big\}_{k=1}^N) = (4)$.

① We start with the first side: $(4) \le f(\mathbf{W}_\omega^*; \big\{\mathbf{T}_k^*\big\}_{k=1}^N)$. We indeed immediately have:

$$(4) = \min_{\mathbf{W}_\omega, \mathbf{T}_k} f(\mathbf{W}_\omega; \{\mathbf{T}_k\}_{k=1}^N) \le f(\mathbf{W}_\omega^*; \{\mathbf{T}_k^*\}_{k=1}^N),$$

due to the definition of min in (4).

② For the other direction, we first show the barycenter loss (1) $\le$ (4). Through the definition of W_2 distance, we have

$$\begin{aligned} & W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega)) \le \|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega\|_F^2, \ \forall \mathbf{T}_k, \mathbf{W}_\omega \\ \Rightarrow\ & \frac{1}{N}\sum_{k=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega)) \le \frac{1}{N}\sum_{k=1}^N \|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega\|_F^2, \ \forall \mathbf{T}_k, \mathbf{W}_\omega \\ \Rightarrow\ & \frac{1}{N}\sum_{k=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega)) \le \frac{1}{N}\sum_{k=1}^N \min_{\mathbf{T}_k} \|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega\|_F^2, \ \forall \mathbf{W}_\omega. \end{aligned}$$

The inequality will still hold when we minimize the two sides both over $\mathbf{W}_\omega$:

$$\begin{aligned} (1) &= \min_{\mathbf{W}_\omega} \frac{1}{N}\sum_{k=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega)) \\ &\le \min_{\mathbf{W}_\omega} \frac{1}{N}\sum_{k=1}^N \min_{\mathbf{T}_k} \|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega\|_F^2 \\ &= (4). \end{aligned}$$

To close the proof, it suffices to show that $f(\mathbf{W}_\omega^*; \mathbf{T}_k^*) = (1)$. We show the equivalence as follows:

$$\begin{aligned} f(\mathbf{W}_\omega^*; \mathbf{T}_k^*) &= \frac{1}{N}\sum_{k=1}^N \big\|\mathbf{T}_k^*\mathbf{W}_k - \mathbf{W}_\omega^*\big\|_F^2 \\ &= \frac{1}{N}\sum_{k=1}^N \min_{\mathbf{T}_k}[\big\|\mathbf{T}_k\mathbf{W}_k - \mathbf{W}_\omega^*\big\|_F^2] \\ &= \frac{1}{N}\sum_{k=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega^*)), \end{aligned}$$

where the last equation holds again thanks to Peyre and Cuturi [9, Proposition 2.1]. Using the fact that W_ω^* is the optimal solution to Wasserstein barycenter problem (1), we finally attain

$$\begin{aligned} f(\mathbf{W}_\omega^*; \mathbf{T}_k^*) &= \frac{1}{N}\sum_{k=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega^*)) \\ &= \min_{\mathbf{W}_\omega} \frac{1}{N}\sum_{i=1}^N W_2^2(\mu_i, \mu_\omega(\mathbf{W}_\omega)) \\ &= (1), \end{aligned}$$

which completes the proof. ◊

References

[1] Mengting Ai, Tianxin Wei, Yifan Chen, Zeming Guo, and Jingrui He. 2025. MLP Fusion: Towards Efficient Fine-tuning of Dense and Mixture-of-Experts Language Models. arXiv:2307.08941 [cs.LG] https://arxiv.org/abs/2307.08941

[2] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7432–7439.

[3] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the State of Neural Network Pruning? arXiv:2003.03033 [cs.LG] https://arxiv.org/abs/2003.03033

[4] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. arXiv:2401.06066 [cs.CL]

[5] William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. https://aclanthology.org/I05-5002

[6] Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2023. Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy. *arXiv preprint arXiv:2310.01334* (2023).

[7] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).

[8] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, Berlin, Germany, 1525–1534. https://doi.org/10.18653/v1/P16-1144

[9] Gabriel Peyre and Marco Cuturi. 2020. Computational Optimal Transport. arXiv:1803.00567 [stat.ML]

[10] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM* 64, 9 (2021), 99–106.

[11] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053 [cs.CL]

[12] Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. *Advances in Neural Information Processing Systems* 33 (2020), 22045–22055.

[13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.

[14] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics* 7 (2019), 625–641. https://doi.org/10.1162/tacl_a_00290

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6