Multi-Scale Model Compression via Nested Matrix Learning

Xiangjue Dong[1*], **Aditya Anantharaman**[2], **Hemant Pugaliya**[2*], **Kai Zhong**[2*]
[1] Texas A&M University, [2] Amazon Inc.
xj.dong@tamu.edu, aditanan@amazon.com, hemantpugaliya@gmail.com, kaizhong89@gmail.com

Abstract

Large language models (LLMs) have been widely deployed and have achieved remarkable success in downstream tasks. However, their high latency continues to pose challenges for real-time applications that require fast inference, and the need to train and deploy distinct models for different hardware constraints increases both financial and computational costs. To address this, we propose **Nested Matrix Learning (NML)**, a method that trains a single, flexible model capable of generating multiple high-performing "student" models of varying sizes. This is achieved by simultaneously optimizing a pre-trained teacher model and its nested sub-models in a single training process, without sacrificing the teacher's performance. NML provides a flexible and scalable solution, allowing models to adapt to different computational budgets. Our extensive experiments show that student models produced by NML, which can be up to $10\times$ smaller than the full-size model, can be directly deployed for efficient inference or serve as superior initialization points for further fine-tuning in downstream tasks. By preserving the performance of the teacher model while delivering compact and efficient student models of various sizes, NML enhances the usability and adaptability of LLMs in real-world scenarios.

Keywords: Large language models (LLMs), Efficient training

1. Introduction

Recent advances in large language models (LLMs) have demonstrated their strength in both generative capabilities (e.g., Mistral (Jiang et al., 2023), LLaMA (Touvron et al., 2023)) and representation learning (e.g., E5-Mistral (Wang et al., 2023), LLM2Vec (BehnamGhader et al., 2024)). However, LLMs still suffer from high latency, which limits their direct deployment in applications requiring fast inference. Knowledge Distillation (KD), which transfers knowledge from large teacher models to smaller student models through the teacher's knowledge is a well-studied method to learn smaller models which can be deployed in low-latency settings. KD typically transfers "knowledge" from the larger teacher model to a smaller student model through (i) the teacher's soft labels or logits; and (ii) the teacher's hidden representations or learned features. However, this requires (i) a multi-step training process of training the teacher model, extracting relevant knowledge from the teacher model followed by training the student model on the extracted knowledge which is time-consuming; and (ii) training multiple student models of varying sizes involves training each student model separately on the extracted knowledge from the teacher. This raises two critical research questions:

1. **Can we bypass the multi-step distillation pipeline and instead learn variable-sized models within a single, unified training run?**

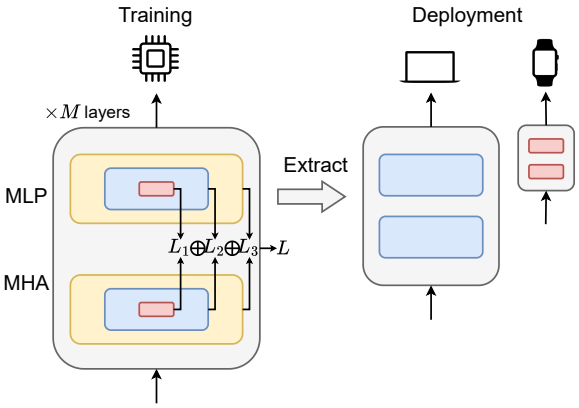

Figure 1: NML introduces a nested structure that enables the joint optimization of multiple nested student models of varying sizes, allowing them to be deployed across different resource settings and constraints.

2. **Can we simultaneously produce a suite of smaller models of different sizes, all trained efficiently at once?**

To answer these questions, we propose **Nested Matrix Learning (NML)**, a method that directly learns the student model's parameters within the teacher model's weight matrices. During fine-tuning, we introduce additional learning objectives that focus on compressing the teacher's knowledge into smaller weight sub-matrices and hidden representations to form coherent, high-performing smaller models (Figure 1). These nested sub-models can be extracted and deployed as-is or used as a vastly superior initialization for further

[*] Work done during Xiangjue's internship and Hemant and Kai's full-time employment at Amazon.

task-specific fine-tuning. Compared to methods like (Devvrit et al., 2023; Cai et al., 2024), NML offers greater flexibility by generating smaller student models with reduced hidden dimensions, enabling more effective scaling of dimensionality and achieving a significantly higher compression ratio.

To evaluate NML, we conduct experiments on the STS suite of tasks (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017; Marelli et al., 2014). Our results show that both the teacher model and the student models extracted through NML perform competitively with fine-tuned and distilled baselines. Moreover, further fine-tuning of the extracted student models significantly improves their performance, indicating the extracted student weights can serve as a better initialization point. We also explore the impact of training multiple nested student models together and show that NML can adapt to various selection strategies.

Our work has the following contributions:

- We introduce NML, a framework that jointly optimizes a full-size model and multiple nested student models of varying hidden dimensions in a single training pass, enhancing flexibility by allowing dynamic scaling based on computational constraints.
- We demonstrate that NML-generated student models, up to $10\times$ smaller than the teacher, can be used directly or serve as superior initialization points, outperforming strong baselines.
- We also analyze NML across several factors, including the impact of the number of nested student models, various weight selection strategies, and loss weighting.

2. Related Work

Knowledge Distillation. Knowledge distillation involves transferring the knowledge from a larger deep neural network into a smaller one (Gou et al., 2021; Xu et al., 2024a), allowing the student model to achieve comparable performance to the teacher while being more computationally efficient. In vanilla knowledge distillation, the logits of the large teacher model are used as the source of knowledge for training the student model (Mirzadeh et al., 2020). (Taori et al., 2023) train student models using sequences generated by teacher LLMs through supervised fine-tuning. There are different approaches to optimizing the student-teacher relationship. Divergence-based methods minimize the divergence between the probability distributions of the teacher and student models (Timiryasov and Tastet, 2023). On the other hand, similarity-based methods focus on aligning the hidden states or features of the student model with those of the teacher, utilizing various similarity metrics to ensure the internal representations of both models are well aligned (Liang et al., 2023).

Student Model Initialization. To leverage the common dimensionality between teacher and student networks, (Sanh et al., 2019) initialize the student model by selecting one out of every two layers from the teacher model. However, this approach requires the smaller model to have the same width as the teacher's to maintain compatibility. (Chen et al., 2022) propose function-preserving initialization which duplicates and stacks the parameters of the existing smaller pre-trained language models as an effective initialization for larger models to speed up convergence. (Trockman et al., 2023) show that initializing convolutional filters with covariance matrices outperforms traditional univariate initialization. (Xia et al., 2024) apply structured pruning to reduce the size of existing large language models and then use the pruned model as an initialization for smaller models. (Xu et al., 2024b) propose a weight selection method, where smaller models are initialized by selecting a subset of weights from a pre-trained larger model. Ensuring consistency in the weight selection process is critical for achieving optimal performance.

Elastic Training. Matryoshka Representation Learning (MRL) (Kusupati et al., 2024), while not operating directly in the weight space, encodes information at varying levels of granularity in a nested fashion. However, MRL only focuses on the final layer's hidden representation, which does not reduce the overall latency or memory footprint needed to load the model. Thus, Espresso Sentence Embeddings (Li et al., 2024) adopts a similar method but learns lower-dimensional hidden representations at each layer using a learning objective combined with PCA. MatFormer (Devvrit et al., 2023) incorporates nested smaller FFN blocks within the standard FFN block, jointly optimizing all granularities to create a single, universal elastic model. Flextron (Cai et al., 2024), building on MatFormer's nested weight structure, adds elasticity to both the MLP and MHA layers, offers a wider range of operations, and supports automatic input-adaptive sub-network selection based on latency for improved efficiency. Unlike MRL, which utilizes full Transformer layers for sentence embedding inference – leading to high computational costs, NML transforms any pre-trained model into a nested model during fine-tuning. This creates multiple sub-models of varying sizes without sacrificing the performance of the teacher model. Compared to weight selection (Xu et al., 2024b), NML offers better initialization for student models. Compared to (Devvrit et al., 2023; Cai et al., 2024), which has

a lower bound of the model size it can reduce to, NML offers greater flexibility by producing smaller student models with reduced hidden dimensions. This flexibility allows NML to scale down the dimensionality of student models more effectively, achieving a significantly higher compression ratio and producing much smaller models.

3. Nested Matrix Learning (NML)

In this section, we first introduce the Nested Matrix Learning (NML) framework aiming to learn sub-matrices for each layer during the teacher model training. Then, we describe the training and inference process for multiple nested student models.

3.1. NML Framework

Problem Formulation. Consider a teacher model with M layers, where the operation of each layer $i \in [1, M]$ is denoted as $f_i(\mathbf{X}_i, \boldsymbol{W}_i)$, with $\mathbf{X}_i$ representing the input to the i-th layer, and $\boldsymbol{W}_i$ being the weight matrix associated with that layer. NML aims to jointly optimize the full model (teacher model) and N nested sub-models (student models) at varying levels of granularity during training. This is achieved by learning the weight sub-matrices $\boldsymbol{W}_i^j$, where $j \in [1, N]$ refers to the j-th sub-model within the nested structure.

3.1.1. Nested Multi-Head Attention (MHA).

NML leverages a nested structure for Multi-Head Attention (MHA) layers, which are critical components of Large Language Models (LLMs) but also contribute significantly to both runtime and memory consumption. By applying the nested design to these MHA layers, NML optimizes them for enhanced computational efficiency and reduced memory usage, making the model more scalable and resource-efficient. Given the hidden states $\mathbf{X}_i$, the nested MHA with h heads is formulated as

$$\mathrm{MHA}^j(x) = \mathrm{Concat}(\mathsf{head}_1, \ldots, \mathsf{head}_n) \cdot \boldsymbol{W}^{O,j},$$

where each attention head for the j-th sub-model is computed as:

$$\mathsf{head}_i = \mathrm{Attn}(\mathbf{X}\boldsymbol{W}_i^{Q,j}, \mathbf{X}\boldsymbol{W}_i^{K,j}, \mathbf{X}\boldsymbol{W}_i^{V,j}),$$

and $\boldsymbol{W}_i^{Q,j}$, $\boldsymbol{W}_i^{K,j}$, $\boldsymbol{W}_i^{V,j}$, and $\boldsymbol{W}_i^{O,j}$ are the sub-matrices used for the query, key, value, and output projections in the j-th model. This nested structure allows NML to scale down the dimensionality of these matrices progressively and flexibly for each nested sub-model.

Sub-matrices Selection. We use a consecutive selection strategy inspired by (Xu et al., 2024b) to select elements for the sub-matrices. Specifically, for the weight matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{D\times H}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{D\times H}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{D\times H}$ in each attention head, we select the first $D^j = \frac{D}{2^j}$ rows and $H^j = \frac{H}{2^j}$ columns for the j-th sub-matrices. Here, D represents the original size of a single attention head, H is the hidden size, and $D = \frac{H}{n}$, where n is the total number of attention heads. Thus, the concatenated sub-matrices of n heads are

$$\boldsymbol{W}^{Q,j}, \boldsymbol{W}^{K,j}, \boldsymbol{W}^{V,j}, \boldsymbol{W}^{O,j} \in \mathbb{R}^{nD^j \times H^j}.$$

This consecutive selection strategy ensures that each sub-matrix captures a reduced but proportionate portion of the full-size matrix in the original attention mechanism, allowing for consistent and meaningful learning across varying dimensions. In addition to the consecutive selection strategy, we experiment with an alternative method – uniform selection, which selects evenly spaced elements from the full-size matrices rather than consecutive rows and columns. This strategy introduces more diversity into the sub-matrices by spreading the selection across the entire matrix. The results of these experiments, along with a detailed comparison analysis of the impact of different selection strategies, are presented in Section 6.

3.1.2. Nested Multi-Layer Perceptron (MLP).

NML employs a nested structure for MLP layers, further leading to reduced memory usage and enhanced efficiency. In addition, NML ensures that the system can dynamically scale across varying resource constraints. Specifically, given the hidden states X_i, the nested MLP for the j-th sub-matrix is formulated as:

$$\mathrm{MLP}^j(x) = \sigma\big(\mathbf{X} \cdot \boldsymbol{W}_1^{jT}\big) \cdot \boldsymbol{W}_2^j,$$

where the full-size weight matrices for the MLP are $\boldsymbol{W}_1$ and $\boldsymbol{W}_2 \in \mathbb{R}^{I\times H}$, with I representing the intermediate size and H the hidden size. The matrices $\boldsymbol{W}_1^j$ and $\boldsymbol{W}_2^j$ are the corresponding sub-matrices, tailored to the reduced dimensionality. The function $\sigma(\cdot)$ denotes a non-linear activation function.

Sub-matrices Selection. Similar to the consecutive selection method used in the nested MHA layers, we apply a similar strategy to extract sub-matrices from the weight matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ in the MLP layers. For the j-th sub-matrix, we select the first $H^j = \frac{H}{2^j}$ columns of the weight matrices, resulting in the sub-matrices:

$$\boldsymbol{W}_1^j = \boldsymbol{W}_1[:, : H^j], \boldsymbol{W}_2^j = \boldsymbol{W}_2[:, : H^j].$$

Further Scale Down MLP. To further reduce the size of the nested MLP layer, in addition to nesting the matrices along the hidden size dimension,

we also scale down by considering the intermediate size like (Devvrit et al., 2023; Cai et al., 2024). Specifically, we extract sub-matrices by selecting the first $H^j = \frac{H}{2^j}$ columns and the first $I^j = \frac{I}{2^j}$ rows of the weight matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ for the j-th sub-matrices:

$$\boldsymbol{W}_1^j = \boldsymbol{W}_1[:I^j, :H^j], \boldsymbol{W}_2^j = \boldsymbol{W}_2[:I^j, :H^j].$$

By reducing the dimensionality in both the hidden and intermediate size directions, we ensure that each nested sub-model has a progressively smaller parameter space, allowing for efficient scaling across multiple sub-models. The smaller sub-models are particularly advantageous in settings with limited computational resources, as they require fewer parameters and operations.

Similarly, for the initial input token embedding layer, we follow the same principle applied to the MLP layer. Here, we extract the first $H^j = \frac{H}{2^j}$ columns from the weight matrices, ensuring consistency in scaling throughout the model architecture. With the nested MLP and MHA layers, we can train NML with N nested sub-models, each parameterized with hidden sizes of H^j.

3.2. Training and Inference

We now describe the NML training and inference phases, focusing on how both full-size model and their nested sub-models are optimized together to enhance performance across different scales.

Consider a full-size model $\mathcal{M}$ and input $\mathbf{x}$, where the original training loss between the model's output and the target $\mathbf{y}$ on the downstream task is represented as $\mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y})$. In the case of the nested sub-models $\mathcal{M}_j$ where $j \in [1, N]$, we jointly optimize both the full-size model $\mathcal{M}$ and all N nested sub-models. Thus, the joint training loss function is:

$$\begin{aligned}\mathcal{L}_{\text{joint}}(\mathbf{x}, \mathbf{y}) = \lambda \cdot \sum_{j=1}^{N} \mathcal{L}(\mathcal{M}_j(\mathbf{x}), \mathbf{y}) \\ + (1-\lambda) \cdot \mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y}),\end{aligned} \tag{1}$$

where λ is a weighting factor that balances the contribution of the sub-models' loss and the full-size model's loss. This allows for flexibility in prioritizing either the sub-models or the full model during training, depending on the task requirements.

During inference, the sub-matrices extracted from NML offer significant flexibility by enabling dynamic deployment of smaller, more efficient models, depending on the specific requirements of the task or the computational environment. These sub-matrices, which are learned as part of NML's nested structure, can be directly deployed as individual smaller student models in scenarios where lightweight models are required. Moreover, these extracted sub-models can also serve as highly effective initialization points for downstream tasks. Since these sub-models are already aligned with the overarching architecture and knowledge learned during the joint training of the full-size model, they provide a better starting point compared to random or standard initializations. This enables faster convergence during fine-tuning, and improves training efficiency and task performance.

4. Experimental Settings

Models. We use the Qwen1.5-0.5B (Bai et al., 2023) as the teacher model for our experiments since it is a relatively smaller model and less resource-intensive to perform fine-tuning as compared to other larger models. However, this method can in practice be extended easily to any teacher model. In our experiments, we focus on learning the sub-matrices at each teacher layer, which means that the student models will have the same number of layers and head numbers as the teacher model and different hidden sizes, significantly reducing the student models' size. Specifically, for Qwen1.5-0.5B, the total number of trainable parameters is 0.5B. The model contains 24 hidden layers, with a hidden dimension H of 1024 and an intermediate size I of 2816. Each MHA layer possesses 16 attention heads. In our implementation, we set the number of nested student models trained with the teacher model N to either 1 or 2. When $N = 1$, NML is trained with a single nested student model with a hidden size of 512. When $N = 2$, NML is trained with two nested student models, with hidden sizes of 512 and 256, and model sizes that are $0.4\times$ and $0.2\times$ that of the teacher model, respectively. Additionally, when we further scale down I, the sizes of the two nested student models are reduced to $0.3\times$ and $0.1\times$ the teacher model's size.

During the inference, we evaluate the performance of directly using the weights extracted from NML as student models, as well as using these extracted weights to initialize and further fine-tune the student models. Specifically, we assess the performance of student models initialized with weights extracted from **NML ($N = 2$)** and **NML ($N = 1$)**, followed by further fine-tuning on NLI datasets.

Baselines. We compare our method to student model baselines: **Randomly Initialization**: the student model is randomly initialized and then fine-tuned on NLI datasets; **Vanilla Distillation**: first, we fine-tune Qwen1.5-0.5B on NLI datasets as the teacher model, then we calculate the cosine similarity between the input text and the positive/negative examples for the teacher model and the randomly initialized student model. Then, we fine-tune the student model by optimizing a distillation loss – MSE loss to these two distributions; **Consecutive**

Selection from Qwen1.5-0.5B: inspired by (Xu et al., 2024b), which demonstrates that weight selection can significantly enhance the performance of small models, we extract weights through consecutive selection from Qwen1.5-0.5B model to initialize the student models and then fine-tune the student model on NLI datasets. Specifically, similar to the consecutive selection strategy used in NML, we extract sub-matrices from each head of the attention module and the first k columns for MLP to initialize the student model, where k is the hidden size of the student model; **Consecutive Selection from fine-tuned Qwen1.5-0.5B**: we extract sub-matrices through consecutive selection from the fine-tuned Qwen1.5-0.5B model which has been fine-tuned on NLI to initialize the student models and then fine-tune the student model on NLI. We do not compare with (Devvrit et al., 2023; Cai et al., 2024) because their methods maintain hidden size unchanged, which differs from our settings. Furthermore, their flexibility limits the compression ratio of their smaller student models, resulting in significantly larger sizes than those from NML when using the same granularity.

Tasks and Metrics: Following previous work (Gao et al., 2021), we train NML on the combination of MNLI (Williams et al., 2018) and SNLI (Bowman et al., 2015) datasets and evaluate the quality of the sentence embeddings from NML on 7 standard semantic textual similarity (STS) tasks: STS 2012–2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), STS Benchmark (Cer et al., 2017) and SICK-Relatedness (Marelli et al., 2014). For the evaluation, we report Spearman's correlation to ensure a fair comparison and use "all" setting, which better reflects real-world scenarios by combining data from different topics.

Implementation Details: In our experiments, we set the learning rate to $1e-5$ and use a batch size of 16, with gradient accumulation steps set to 128 to effectively manage GPU memory. All models are trained for 3 epochs using 8 V100-SXM2-32GB GPUs. For the optimization of student models, we conduct a parameter search over $\{0.3, 0.5, 0.8, 0.9, 1.0\}$ and set the weight for the student model loss λ to 0.9. This ensures that the learning of the student models is prioritized while still allowing for flexibility in the overall training dynamics. In the implementation, we use the objective function $\mathcal{L} = w_1 \cdot \mathcal{L}_{cl} + w_2 \cdot \mathcal{L}_{angle}$, where $\mathcal{L}_{cl}$ is the widely-used supervised contrastive learning objective, and $\mathcal{L}_{angle}$ is the angle objective, which aims to minimize the angle difference for high-similarity pairs compared to those with lower similarity (Li and Li, 2024). We assign a weight of 20.0 to the contrastive loss $\mathcal{L}_{cl}$ and 1.0 to the angle loss $\mathcal{L}_{angle}$ following previous work (Li and Li, 2024). We perform a grid search over gradient accumulation steps from the set $\{64, 128, 256, 512\}$ and learning rates from $\{1e-4, 1e-5, 3e-5, 5e-5\}$ and adopt the hyperparameter settings mentioned in Section 4. The widely-used contrastive learning objective $\mathcal{L}_{cl}$ is defined as:

$$\mathcal{L}_{cl} = -\sum_{b}\sum_{i}^{m} \log \left[\frac{e^{\cos(\mathbf{X}_{b_i}, \mathbf{X}^+_{b_i})/\tau}}{\sum_j^N e^{\cos(\mathbf{X}_{b_i}, \mathbf{X}^+_{b_j})/\tau}} \right],$$

where b represents the batch index, m is the number of positive pairs in the b-th batch, and N denotes the batch size. The terms $\mathbf{X}^+_{b_i}$ and $\mathbf{X}^+_{b_j}$ denote the respective positive samples corresponding to $\mathbf{X}_{b_i}$ and $\mathbf{X}_{b_j}$, respectively, while $\cos(\cdot)$ indicates the cosine similarity function. τ is the temperature hyperparameter. The angle objective is given by:

$$\mathcal{L}_{angle} = \log \left[1 + \sum_{s_{ij} > s_{mn}} \exp\left(\frac{\Delta\theta_{ij} - \Delta\theta_{mn}}{\tau}\right) \right],$$

where τ is a temperature hyperparameter, s represents the cosine similarity, and $\Delta\theta$ is the angle difference. The condition $s_{ij} > s_{mn}$ stems from the ranking of training data labels. The optimizing of $\mathcal{L}_{angle}$ aims to reduce the angle difference for pairs with higher similarity relative to those with low similarity (Li and Li, 2024). Both the model and dataset are licensed under the Apache-2.0 license. For the test data statistics of STS12-16, STS-B, and SICK-R, we refer to (Muennighoff et al., 2023) and report the following counts: 6216, 3000, 7500, 6000, 2372, 2758, and 19854, respectively.

5. Experimental Results

Our experiments are designed to answer three primary questions about NML's effectiveness.

How effective are NML-extracted student models as standalone models and as initializations? First, we show the performance of student models with different sizes on the STS benchmark in Table 1. We can see when student models are directly extracted from NML ($N = 2$) and NML ($N = 1$) without further fine-tuning, they achieve comparable or better performance to baselines, which involve weight extraction followed by fine-tuning. This demonstrates that weights extracted from NML can be effectively used as student models without the need for further fine-tuning. Then, after further fine-tuning the extracted NML student models on NLI datasets, it further improves the performance to 65.42 and 58.98 for student models with different sizes. This indicates that weights extracted from NML can also serve as a strong initialization point

Models	Fine-tuned?	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Student Model: $H = 512$, $I = 2816$, # params = 0.2B ($0.4\times$)									
Random Init	Yes	43.31	39.25	39.51	62.09	50.56	46.75	62.27	49.11
Vanilla Distillation	Yes	43.96	39.73	40.66	61.92	49.56	47.75	62.44	49.43
Consecutive Selection from Qwen1.5-0.5B	Yes	62.16	**51.72**	55.73	73.20	65.17	65.91	70.25	63.45
Consecutive Selection from fine-tuned Qwen1.5-0.5B	Yes	63.92	50.66	54.66	72.98	65.89	66.44	70.51	63.58
NML ($N = 2$)	No	64.18	49.04	54.31	71.72	64.44	65.04	68.95	62.53
NML ($N = 1$)	No	64.66	48.04	55.10	72.71	65.28	65.37	70.31	63.07
NML ($N = 2$)	Yes	64.38	51.05	56.86	73.57	66.62	67.77	73.06	64.76
NML ($N = 1$)	Yes	**65.95**	50.37	**57.67**	**74.76**	**67.02**	**68.38**	**73.77**	**65.42**
Student Model: $H = 256$, $I = 2816$, # params = 97M ($0.2\times$)									
Random Init	Yes	40.58	34.74	36.38	58.71	48.63	45.84	58.95	46.26
Consecutive Selection from Qwen1.5-0.5B	Yes	60.78	33.75	37.52	59.27	50.60	47.88	65.04	50.69
Consecutive Selection from fine-tuned Qwen1.5-0.5B	Yes	58.53	38.83	43.31	62.58	56.10	55.35	66.38	54.44
NML ($N = 2$)	No	60.39	40.78	43.80	60.55	54.43	53.25	65.89	54.16
NML ($N = 2$)	Yes	**62.28**	**45.00**	**48.77**	**67.63**	**59.15**	**60.32**	**69.74**	**58.98**

Table 1: Performance of NML student models on STS benchmark (Spearman's correlation, "all" setting). We highlight the best performance. "Fine-tuned?" refers to whether the student model was further fine-tuned after extracting weights from the teacher. "N" refers to the number of student models nested and trained within the teacher model.

for student models, offering further performance gains when fine-tuned on downstream tasks. In addition, NML shows good generalization ability for student models with different sizes.

Can we further scaling down MLP intermediate layer? We further reduce the size of the student models by scaling down the MLP intermediate layer dimensions to $1/2^n$ of the original intermediate size, where n represents the number of student models. This results in student models with fewer parameters and a smaller overall size. Here, we present the results of NML approach in Table 2, which shows a trend consistent with previous findings: weights extracted from NML can achieve comparable or better performance than baselines, indicating that they can be effectively used as student models without the need for further fine-tuning. Then, after further fine-tuning the extracted NML student models on NLI datasets, it further improves the performance to 62.81 and 59.63 for student models with different sizes. This indicates that weights extracted from NML can also serve as a strong initialization point for these student models, offering further performance gains when fine-tuned on downstream tasks. Additionally, we notice that for the student model whose hidden size is 256, the performance of the NML student improves from 58.98 to 59.63 even after scaling down the MLP intermediate layer to $0.25\times$ the original intermediate size, suggesting that NML allows us to reduce the model size much further without compromising on the performance.

Does the joint optimization process harm the teacher model's performance? The above results demonstrate that NML can effectively produce multiple student models of varying sizes simultaneously. Since NML is training multiple student models nested in the teacher model training, to illustrate that NML does not compromise the performance of the full-size teacher model, we present the detailed results in Table 3, which compare the performance of the full-size models after NML training with Qwen model without and with downstream task fine-tuning. As shown in the table, the full-size models not only retain comparable performance but, in some cases, exhibit improvement over the fine-tuned Qwen model. This indicates that the application of NML does not result in any degradation of the teacher model's performance, affirming its robustness and utility in scenarios requiring efficient model scaling while preserving high-level performance. This preservation of the performance of the teacher model with the obtained multiple smaller student models further highlights the adaptability and strength of NML, ensuring that the teacher models' performance remains uncompromised in real-world applications.

6. Analyses

To better understand NML's behavior, we conducted several additional studies.

Models	Fine-tuned?	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Student Model: $H = 512$, $I = 1408$, # params = 155M $(0.3\times)$									
Random Init	Yes	44.50	38.14	37.77	60.12	49.31	45.81	60.71	48.05
Vanilla Distillation	Yes	42.14	37.73	39.71	61.55	49.00	45.86	61.39	48.20
Consecutive Selection from Qwen1.5-0.5B	Yes	58.63	45.69	48.99	67.11	63.27	62.82	68.94	59.35
Consecutive Selection from fine-tuned Qwen1.5-0.5B	Yes	60.96	41.94	45.46	66.57	62.16	60.28	68.45	57.97
NML ($N = 2$)	No	57.93	44.11	47.23	67.86	62.37	62.56	67.94	58.57
NML ($N = 1$)	No	59.35	46.30	49.74	68.43	63.48	63.38	69.02	59.96
NML ($N = 2$)	Yes	63.32	46.73	52.33	71.03	64.84	66.41	70.89	62.22
NML ($N = 1$)	Yes	**63.39**	**48.92**	**52.81**	**71.46**	**64.71**	**66.86**	**71.51**	**62.81**
Student Model: $H = 256$, $I = 704$, # params = 58M $(0.1\times)$									
Random Init	Yes	40.19	33.34	33.15	55.40	43.09	39.07	56.16	42.91
Consecutive Selection from Qwen1.5-0.5B	Yes	58.02	44.04	45.47	65.91	61.15	59.30	65.14	57.00
Consecutive Selection from fine-tuned Qwen1.5-0.5B	Yes	57.90	45.50	45.97	63.73	59.49	57.27	64.72	56.37
NML ($N = 2$)	No	57.92	43.81	44.53	64.19	59.19	56.89	64.50	55.86
NML ($N = 2$)	Yes	**61.67**	**45.08**	**47.63**	**68.45**	**63.29**	**63.02**	**68.24**	**59.63**

Table 2: Performance of NML student models with **further scaling down** on STS benchmark (Spearman's correlation, "all" setting). We highlight the best performance. "Fine-tuned?" refers to whether the student model was further fine-tuned after extracting weights from the teacher. "N" refers to the number of student models nested and trained within the teacher model.

Models	Fine-tuned?	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Qwen1.5-0.5B	No	60.84	76.65	65.14	75.25	76.40	74.47	69.56	71.19
Qwen1.5-0.5B	Yes	76.83	83.66	80.29	85.64	83.65	84.56	76.20	81.55
NML ($N = 2$)	Yes	76.54	84.39	80.29	86.00	83.79	85.15	77.12	81.90
NML ($N = 1$)	Yes	76.70	82.54	79.74	85.41	83.50	84.77	76.37	81.29

Table 3: Performance of NML full-size models on STS benchmark (Spearman's correlation, "all" setting).

Models	Intermediate Size	# Params	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
NML ($N = 2$)	1408	155M	57.93	44.11	47.23	67.86	62.37	62.56	67.94	58.57
NML ($N = 1$)	1408	155M	59.35	46.30	49.74	68.43	63.48	63.38	69.02	59.96
NML ($N = 2$)	2816	207M	64.18	49.04	54.31	71.72	64.44	65.04	68.95	62.53
NML ($N = 1$)	2816	207M	64.66	48.04	55.10	72.71	65.28	65.37	70.31	63.07

Table 4: Impact of the number of nested student models trained in NML ($H = 512$).

What is the impact of the number of nested student models training simultaneously? We analyze the performance impact of training configurations where either one student model or two student models are trained simultaneously using a nested approach. In Table 4, we present the results for student models directly extracted from NML without further fine-tuning on downstream tasks. The results consistently show that training with fewer nested student models tends to have slightly better performance, whether or not additional scaling down is applied. When only one student model is nested trained in NML, the model faces fewer constraints, which allows for better overall performance. As the number of nested student models increases, the nested models may influence each other's learning processes. This interaction could introduce competition for resources for each individual model. However, our findings suggest that this competition does not significantly degrade the performance of the nested student models. Instead, the performance remains relatively stable, indicating that the impact of multiple nested models on overall performance is relatively minor. This indicates that NML is a viable solution for scenarios requiring multiple student models training simultaneously while maintaining strong individual performance for each model.

Which sub-matrix selection strategy is more effective? To further investigate the effectiveness of different sub-matrix selection strategies within

Selection Strategy	Hidden Size	Model Size	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Uniform	1024	0.5B	76.65	83.76	79.76	85.57	83.37	84.16	75.99	81.32
	512	207M	63.45	50.57	51.77	68.88	63.71	64.05	69.02	61.64
	256	97M	55.82	44.63	44.34	63.29	52.91	51.51	63.82	53.76
Consecutive	1024	0.5B	76.54	84.39	80.29	86.00	83.79	85.15	77.12	81.90
	512	207M	64.18	49.04	54.31	71.72	64.44	65.04	68.95	62.53
	256	97M	58.53	38.83	43.31	62.58	56.10	55.35	66.38	54.44

Table 5: Comparison results of NML ($N = 2$) with different selection strategies on STS benchmark.

the NML framework, we conducted a comparison study by implementing a uniform selection method, following the approach proposed by (Xu et al., 2024b). In this method, evenly spaced elements are selected from the teacher model's weight matrices, in contrast to the consecutive selection strategy where contiguous sub-matrices are chosen. Table 5 presents the performance results of NML ($N = 2$) when applied with both the uniform and consecutive selection strategies, along with the corresponding performance of the full-size teacher model and the two nested student models. The results indicate that NML using the consecutive selection strategy consistently outperforms the uniform selection method across all models. This performance boost suggests that the consecutive selection strategy may better preserve the coherence of the original weight matrices, enabling the nested student models to benefit from more effective parameter sharing.

How does the loss weight (λ) influence the nested student models' performance? To analyze the impact of loss weight of nested student models, we present the results of NML ($N = 2$) in Figure 2. The findings reveal a relationship between the loss weight λ and model performance. Specifically, as the loss weight of the nested student models increases from 0.8 to 0.9, we observe an improvement in the performance of the smaller model with a larger hidden size ($H = 512$). This enhancement can be attributed to the increased emphasis on student learning, which allows the model to leverage more information from the training process. Conversely, the smaller model with a smaller hidden size ($H = 256$) experiences a decrease in performance, highlighting a trade-off inherent in nested model training. As the loss weight reaches 1.0, the focus shifts entirely to training the nested student models, regardless of the full model parameters, which significantly impacts the overall performance of the full-size model. Therefore, it is important to carefully calibrate the loss weights to achieve an optimal balance between the performance of larger and smaller student models, and to fully leverage the potential of nested model architectures.

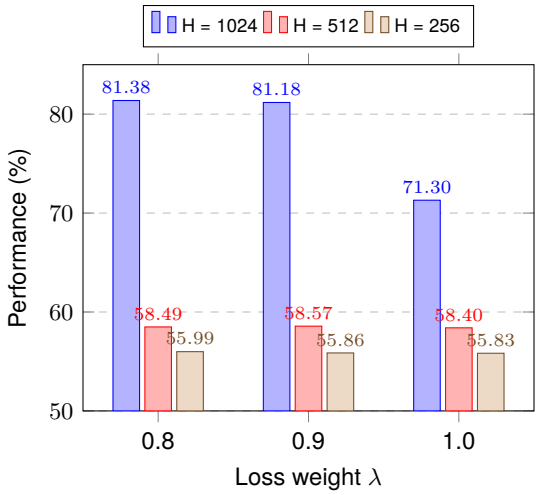

Figure 2: Performance comparison for different student model loss weights λ of NML ($N = 2$).

7. Conclusion

We introduce Nested Matrix Learning (NML), a new highly efficient framework for generating multiple student models of varying sizes during the fine-tuning process of a pre-trained teacher model, all while preserving the original performance of the teacher model. This flexibility allows NML to create scalable models that are well-suited for different computational budgets and downstream tasks. The student models produced by NML can either be deployed directly in scenarios where lightweight models are needed, or they can serve as strong initialization points for further fine-tuning on downstream tasks. NML's capacity to create high-performing, compact smaller models makes it a powerful tool for real-world applications where memory efficiency and computational speed are crucial, while still maintaining competitive accuracy. This approach can reduce the financial and environmental costs associated with training and maintaining multiple separate models.

Future work can extend this research in several promising directions. The most direct path is applying NML during the pre-training phase, learning a nested structure from scratch. Additionally, the submatrix selection strategy itself could be a learned

policy rather than a fixed one, potentially discovering more optimal sub-architectures. Extending NML to other modalities, such as vision transformers, is another promising direction.

8. Ethics Statement

Compressing large models may inadvertently amplify the existing biases present in the original pretrained models. If the student models inherit biases from the teacher model, it could lead to unfair or discriminatory outcomes, especially if the models are deployed in sensitive applications like hiring, healthcare, or law enforcement. To ensure fairness and prevent bias, a thorough evaluation of the smaller models is necessary. In addition, while NML aims to reduce model size and improve computational efficiency, training multiple nested models could still raise concerns about the environmental impact.

9. Bibliographical References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. Flextron: Many-in-one flexible large language model. In *Forty-first International Conference on Machine Learning*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2022. bert2BERT: Towards reusable pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2148, Dublin, Ireland. Association for Computational Linguistics.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. 2023. Matformer: Nested transformer for elastic inference.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2024. Matryoshka representation learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Xianming Li and Jing Li. 2024. AoE: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1839, Bangkok, Thailand. Association for Computational Linguistics.

Xianming Li, Zongxi Li, Jing Li, Haoran Xie, and Qing Li. 2024. Ese: Espresso sentence embeddings.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pages 20852–20867. PMLR.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5191–5198.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Inar Timiryasov and Jean-Loup Tastet. 2023. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 279–289, Singapore. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Asher Trockman, Devin Willmott, and J Zico Kolter. 2023. Understanding the covariance structure of convolutional filters. In *The Eleventh International Conference on Learning Representations*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024a. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. 2024b. Initializing models with larger ones. In *The Twelfth International Conference on Learning Representations*.

10. Language Resource References

Agirre, Eneko and Banea, Carmen and Cardie, Claire and Cer, Daniel and Diab, Mona and Gonzalez-Agirre, Aitor and Guo, Weiwei and Lopez-Gazpio, Iñigo and Maritxalar, Montse and Mihalcea, Rada and Rigau, German and Uria, Larraitz and Wiebe, Janyce. 2015. *SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability*. Association for Computational Linguistics.

Agirre, Eneko and Banea, Carmen and Cardie, Claire and Cer, Daniel and Diab, Mona and Gonzalez-Agirre, Aitor and Guo, Weiwei and Mihalcea, Rada and Rigau, German and Wiebe, Janyce. 2014. *SemEval-2014 Task 10: Multilingual Semantic Textual Similarity*. Association for Computational Linguistics.

Agirre, Eneko and Banea, Carmen and Cer, Daniel and Diab, Mona and Gonzalez-Agirre, Aitor and Mihalcea, Rada and Rigau, German and Wiebe, Janyce. 2016. *SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation*. Association for Computational Linguistics.

Agirre, Eneko and Cer, Daniel and Diab, Mona and Gonzalez-Agirre, Aitor. 2012. *SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity*. Association for Computational Linguistics.

Agirre, Eneko and Cer, Daniel and Diab, Mona and Gonzalez-Agirre, Aitor and Guo, Weiwei. 2013. **SEM 2013 shared task: Semantic Textual Similarity*. Association for Computational Linguistics.

Jinze Bai and Shuai Bai and Yunfei Chu and Zeyu Cui and Kai Dang and Xiaodong Deng and Yang Fan and Wenbin Ge and Yu Han and Fei Huang and Binyuan Hui and Luo Ji and Mei Li and Junyang Lin and Runji Lin and Dayiheng Liu and Gao Liu and Chengqiang Lu and Keming Lu and Jianxin Ma and Rui Men and Xingzhang Ren and Xuancheng Ren and Chuanqi Tan and Sinan Tan and Jianhong Tu and Peng Wang and Shijie Wang and Wei Wang and Shengguang Wu and Benfeng Xu and Jin Xu and An Yang and Hao Yang and Jian Yang and Shusheng Yang and Yang Yao and Bowen Yu and Hongyi Yuan and Zheng Yuan and Jianwei Zhang and Xingxuan Zhang and Yichang Zhang and Zhenru Zhang and Chang Zhou and Jingren Zhou and Xiaohuan Zhou and Tianhang Zhu. 2023. *Qwen Technical Report*. arXiv.

Bowman, Samuel R. and Angeli, Gabor and Potts, Christopher and Manning, Christopher D. 2015. *A large annotated corpus for learning natural language inference*. Association for Computational Linguistics.

Cer, Daniel and Diab, Mona and Agirre, Eneko and Lopez-Gazpio, Iñigo and Specia, Lucia. 2017. *SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation*. Association for Computational Linguistics.

Marelli, Marco and Menini, Stefano and Baroni, Marco and Bentivogli, Luisa and Bernardi, Raffaella and Zamparelli, Roberto. 2014. *A SICK cure for the evaluation of compositional distributional semantic models*. European Language Resources Association (ELRA).

Williams, Adina and Nangia, Nikita and Bowman, Samuel. 2018. *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*. Association for Computational Linguistics.

A. More Implementation Details

We perform a grid search over gradient accumulation steps from the set $\{64, 128, 256, 512\}$ and learning rates from $\{1e-4, 1e-5, 3e-5, 5e-5\}$ and adopt the hyperparameter settings mentioned in Section 4. The widely-used contrastive learning objective $\mathcal{L}_{cl}$ is defined as:

$$\mathcal{L}_{cl} = -\sum_{b}\sum_{i}^{m} \log \left[\frac{e^{\cos(\mathbf{X}_{b_i}, \mathbf{X}^+_{b_i})/\tau}}{\sum_j^N e^{\cos(\mathbf{X}_{b_i}, \mathbf{X}^+_{b_j})/\tau}} \right],$$

where b represents the batch index, m is the number of positive pairs in the b-th batch, and N denotes the batch size. The terms $\mathbf{X}^+_{b_i}$ and $\mathbf{X}^+_{b_j}$ denote the respective positive samples corresponding to $\mathbf{X}_{b_i}$ and $\mathbf{X}_{b_j}$, respectively, while $\cos(\cdot)$ indicates the cosine similarity function. τ is the temperature hyperparameter. The angle objective is given by:

$$\mathcal{L}_{angle} = \log \left[1 + \sum_{s_{ij} > s_{mn}} \exp\left(\frac{\Delta\theta_{ij} - \Delta\theta_{mn}}{\tau}\right) \right],$$

where τ is a temperature hyperparameter, s represents the cosine similarity, and $\Delta\theta$ is the angle difference. The condition $s_{ij} > s_{mn}$ stems from the ranking of training data labels. The optimizing of $\mathcal{L}_{angle}$ aims to reduce the angle difference for pairs with higher similarity relative to those with low similarity (Li and Li, 2024). Both the model and dataset are licensed under the Apache-2.0 license. For the test data statistics of STS12-16, STS-B, and SICK-R, we refer to (Muennighoff et al., 2023) and report the following counts: 6216, 3000, 7500, 6000, 2372, 2758, and 19854, respectively.