Compression of Large Language Models by Condensed Weight Representation

Yancheng Wang [1] **Dongfang Sun** [1] **Yingzhen Yang** [1]

Abstract

The rapid growth in the size of Large Language Models (LLMs) poses significant challenges for deployment, particularly in resource-limited environments. To address this issue, we propose *Neuron Summary* (NS), a novel approach for compressing LLMs by constructing compact and condensed representations of the weights in their linear layers. Given that these layers contribute the most to the overall model size, NS offers an effective method to reduce the model size and computational costs while maintaining strong performance in downstream natural language processing tasks. Our compressed model, *NSNet*, substitutes each linear layer in an LLM with an *NS-Linear* layer, where the weights are represented using NS. The transition from a pre-trained LLM to NSNet is achieved through regression-based initialization, followed by knowledge distillation to preserve the original model's capabilities. Extensive experiments on compressing various LLMs, including DeBERTaV3-base and Llama-2, demonstrate that NS significantly outperforms existing compression methods across multiple tasks, such as natural language understanding, question answering, and text generation. Additionally, NS is complementary to other compression techniques, such as quantization and layer-wise parameter sharing, enabling further reduction in model size while maintaining competitive performance. The code of NSNet is available at https://anonymous.4open.science/r/NSNet-D6B8/.

[*]Equal contribution [1]School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, USA. Correspondence to: Yingzhen Yang <yingzhen.yang@asu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Large language models (LLMs) have achieved state-of-the-art performance across a wide range of natural language processing (NLP) tasks, including classification, translation, and generation (Devlin et al., 2018; Liu et al., 2019; He et al., 2020; Radford et al., 2019; Brown et al., 2020). However, these performance gains come at the cost of substantial computational and memory overheads, primarily due to the vast number of parameters in modern LLMs. To alleviate this issue, numerous compression strategies have been developed, including pruning (Frantar & Alistarh, 2023; Ma et al., 2023a), quantization (Lin et al., 2024; Zhao et al., 2024; Dettmers et al., 2022), and knowledge distillation (Gu et al., 2024; Magister et al., 2023; Jiang et al., 2023; Huang et al., 2022; Qiu et al., 2024).

Among these methods, low-rank approximations, particularly those based on singular value decomposition (SVD) (Yuan et al., 2023; Hsu et al., 2022; Wang et al., 2024), have emerged as effective tools for reducing parameter count while maintaining compatibility with pretrained weights. These approaches are hardware-agnostic and can be integrated with other techniques such as pruning and quantization (Cheng et al., 2017). Additionally, parameter sharing has proven to be another powerful means of model size reduction by reusing weights across different layers (Dehghani et al., 2019; Takase & Kiyono, 2021; Hay & Wolf, 2024; Wang et al., 2025).

Despite these advances, existing methods primarily exploit redundancy between layers, overlooking significant redundancy within individual linear layers themselves—where most LLM parameters reside (e.g., 90.8% in BERT (Devlin et al., 2018), 98.0% in Llama-2-7B (Touvron et al., 2023c)). To address this gap, we introduce *Neuron Summary* (NS), a novel compression method that captures intra-layer redundancy by representing each linear layer as a shared 1D structure. Substituting all linear layers with their NS counterparts yields *NSNet*, a compact network that maintains the architecture of the original model. We further propose a regression-based initialization strategy to construct NSNet from a pretrained LLM, followed by knowledge distillation to fine-tune performance alignment.

Contributions. Our contributions are summarized as follows:

First, we propose a new model compression method, *Neuron Summary* (NS), which encodes the weights of linear layers using a compact and learnable 1D representation. A given LLM is transformed into an *NSNet* by replacing its linear layers with *NS-Linear* layers, each parameterized by a corresponding NS vector. To initialize NSNet from pre-trained LLMs, we design a novel initialization mechanism based on linear regression. To maintain task accuracy, we further apply knowledge distillation to align NSNet's outputs with those of the original model.

Second, we demonstrate that NS significantly outperforms existing compression techniques across multiple popular LLMs—including DeBERTaV3-base (He et al., 2021), BERT-base (Devlin et al., 2018), Llama-2-7B, and Llama-2-13B (Touvron et al., 2023c)—on natural language understanding, question answering, and text generation tasks. **As shown in Figure 3 and Figure 2 in Appendix A.7, NSNet consistently achieves faster inference and smaller model size while matching or exceeding the accuracy of competing baselines.** Moreover, NS can be seamlessly combined with other compression techniques such as quantization (Lin et al., 2023) and inter-layer parameter sharing (Wang et al., 2025), enabling further gains in efficiency, as demonstrated in Table 4 (Appendix A.3.1) and Figure 2 (Appendix A.7).

2. Related Works

2.1. Compression of Large Language Model without Parameter Sharing

LLM compression aims to reduce model size and latency without degrading performance, using techniques such as distillation, pruning, and quantization. Distillation trains a smaller student model to replicate a larger teacher's behavior (Gu et al., 2024; Huang et al., 2022; Magister et al., 2023; Jiang et al., 2023). Pruning removes unnecessary weights or neurons (Frantar & Alistarh, 2023; 2022; Sun et al., 2024; Ma et al., 2023a), either structurally or unstructured, while quantization reduces parameter precision to enhance hardware efficiency (Zhao et al., 2024; Ashkboos et al., 2024; Lin et al., 2024; Xiao et al., 2023).

SVD-based compression offers tunable compression ratios without retraining (Golub et al., 1987; Lv et al., 2023; Wu et al., 2023). However, it struggles with outlier activations in LLMs. Enhancements include FWSVD, which uses Fisher information but is costly (Hsu et al., 2022); LoSparse, which combines low-rank and sparse components (Li et al., 2023); ASVD, which targets sensitive channels (Yuan et al., 2023); and SVD-LLM, which incorporates a whitening matrix to better handle outliers (Wang et al., 2024).

2.2. Compression of Large Language Model with Parameter Sharing

Weight sharing compresses LLMs by reusing parameters across layers, inspired by RNNs. Universal sharing applies a single set of weights to all encoder/decoder layers (Dehghani et al., 2019), while other methods share attention and feedforward weights separately (Reid et al., 2021), or selectively across specific layers (Takase & Kiyono, 2021). Sharing attention scores also reduces redundancy (Xiao et al., 2019; Bhojanapalli et al., 2021). Dynamic Tying uses reinforcement learning to decide sharing strategies during training (Hay & Wolf, 2024). These methods can be combined with SVD-based approaches. For example, Basis Sharing shares SVD basis vectors across layers for more efficient compression (Wang et al., 2025).

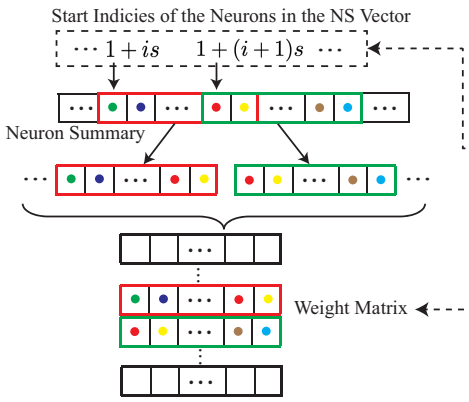
3. Formulation of Neuron Summary for Linear Layers

We introduce the Neuron Summary (NS), which serves as a compact representation of the neurons within a linear layer, as described in Section 3.1. Following that, we present the procedure for compressing a LLM, which entails replacing every linear layer in the original model with an NS-Linear layer and encoding the weights of each original linear layer as a compact NS. A model consisting of these NS-Linear layers, referred to as an NSNet, is initialized from a pre-trained LLM following the approach described in Section 3.2. Finally, in Section 3.3, we introduce a fine-tuning method based on knowledge distillation, specifically developed to preserve the strong performance of the NSNet after the compression and initialization steps.

3.1. Neuron Summary (NS) for Linear Layers: the NS-Linear Layer

In a conventional linear layer, the weight matrix $\mathbf{W} \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ has the out dimension D_{out}, which is also the number of neurons of this linear layer, and D_{in} is the size of the input. Throughout this paper, we use neurons to denote the rows of the weight matrix $\mathbf{W}$ in the linear layer. The total number of parameters of the linear layer is $D_{\text{in}} \times D_{\text{out}}$. To compress this parameter space, we compress all the neurons in a one-dimensional vector called Neuron Summary (NS), denoted as $\mathbf{S}$. Let $r \in (0, 1)$ denote the compression ratio, which is the ratio of the number of parameters of the compressed model to that of the original model. As a result, the length of $\mathbf{S}$, is $L = \lfloor D_{\text{in}} \times D_{\text{out}} \times r \rfloor$. Each neuron is assigned a segment of D_{in} elements in the NS, and these segments overlap to enable weight sharing.

Elements of the neurons are extracted from the NS with a



(a)

```
def forward(self, x):
    # $L$ (int): Length of the neuron summary.
    # D_in (int): Input dimension.
    # D_out (int): Output dimension.

    # Calculate the start indices of the neurons in
        the NS vector
    self.start_indices = self.calculate_start_indices(
        L, D_in, D_out)

    # Fetch the neurons form the NS vector
    weights = [self.neuron_summary[idx:idx + self.D_in
        ]
        for idx in self.start_indices]

    # Construct the weight matrix
    weight_matrix = torch.stack(weights, dim=1)

    # Perform the matrix multiplication
    output = torch.matmul(x, weight_matrix)
return output
```

(b)

Figure 1: (a) Illustration of the weight sharing across the weights of different neurons using Neuron Summary (NS) in a NS-Linear layer. The red and yellow dots represent the naturally shared weights of the two neurons extracted from the NS. (b) Pseudocode of the forward function of a NS-Linear layer. 'calculate_start_indicies' is a function that calculates the start indices of the neurons in the NS vector. The start index for the i-th neuron is $1 + (i-1)s$, where $s = \lfloor \frac{L-D_{\text{in}}}{D_{\text{out}}} \rfloor$.

fixed stride s, which determines the spacing between the segments. The stride is computed as $s = \lfloor \frac{L-D_{\text{in}}}{D_{\text{out}}} \rfloor$. The weights of the i-th neuron are extracted as a contiguous segment starting from position $1+(i-1)s$ in $\mathbf{S}$. As a result, the weights of the i-th neuron are $\mathbf{S}_{1+(i-1)s:(i-1)s+D_{\text{in}}}$ for $i \in \{1, 2, \ldots, D_{\text{out}}\}$.

The NS replaces the conventional weight matrix. During forward propagation, the elements for all neurons are dynamically reconstructed into a matrix $\mathbf{W}$ of size $D_{\text{out}} \times D_{\text{in}}$. The rows of $\mathbf{W}$ are populated by extracting overlapping segments from $\mathbf{S}$. The linear layer then computes its output as $\mathbf{y} = \mathbf{x}\mathbf{W}^\top$, where $\mathbf{x}$ is the input. Figure 1 (a) illustrates the weight sharing across the weights of different neurons using NS. Figure 1 (b) presents the pseudocode of the forward computation where the weight matrix of a linear layer is reconstructed from the NS. Despite the reduction in parameters, the linear layer operates in the same manner as the traditional implementation of a linear layer. Such a new linear layer with a NS is termed a NS-Linear layer. A deep neural network with all its linear layers replaced by the NS-Linear layers is called a Neural Summary Network, or NSNet.

3.2. Initialization of the NSNet From a Pre-Trained LLM

To train LLMs for natural language processing tasks, existing works typically first pre-train the models, such as BERT (Devlin et al., 2018) and LLaMA (Touvron et al., 2023a), on large-scale corpora (He et al., 2020; Touvron et al., 2023c) to learn general language representations. These pre-trained LLMs are then fine-tuned on specific downstream tasks, such as natural language understanding on GLUE (Wang et al., 2019), question answering tasks on SQuAD (Rajpurkar et al., 2016b), and language generation on WikiText (Merity et al., 2016). To obtain compact LLMs with reduced model sizes, model compression methods, such as the SVD decomposition (Li et al., 2023; Yuan et al., 2023) and the parameter sharing (Hay & Wolf, 2024; Wang et al., 2025), have been employed to compress LLMs. To maintain the competitive performance of these compressed models on downstream tasks, existing works either fine-tune the compressed models on the downstream tasks during the compression process (Li et al., 2023) or perform the fine-tuning after completing the compression process (Yuan et al., 2023).

To apply NS to compress the LLMs for downstream natural language processing tasks, we follow the same protocol as existing works by first compressing the pre-trained LLMs with NS and then fine-tuning the compressed models on the downstream tasks. To compress the pre-trained LLMS with NS, we replace each conventional linear layer in the model with the NS-Linear layer described in the previous subsection, whose weights are represented as NS. The pre-trained model does not have NS vectors in their linear layers. Furthermore, it would be very time-consuming and resource-demanding to pre-train the NSNet model. To this end, we propose a novel NS initialization method which initializes a NSNet from an off-the-shelf pre-trained model. Such initialized NSNet is used as the pre-trained NSNet for the subsequent tasks. We note that a carefully designed fine-tuning process with knowledge distillation described in Section 3.3 is employed to ensure the compelling performance of the NSNet for downstream tasks.

In the NS initialization method, we initialize the weights of all the NS-Linear layers by solving a regression problem that minimizes the mean squared error (MSE) between the weight matrix reconstructed from the NS and the original weight matrix for each layer. Let M be the number of

linear layers in the original model. Given the weight matrix $\mathbf{W}^{(m)} \in \mathbb{R}^{D_{\text{out}}^{(m)} \times D_{\text{in}}^{(m)}}$ from the m-th linear layer in the pre-trained model, and the NS vector $\mathbf{S}^{(m)} \in \mathbb{R}^{L^{(m)}}$, the weights for the i-th neuron in the NS-Linear layer are reconstructed as $\widehat{\mathbf{w}}_i^{(m)} = \mathbf{S}^{(m)}_{1+(i-1)s^{(m)}:(i-1)s^{(m)}+D_{\text{in}}^{(m)}}$ for every $i \in \{1, 2, \ldots, D_{\text{out}}^{(m)}\}$. The reconstructed weight matrix $\widehat{\mathbf{W}}^{(m)}$ for the m-th layer is formed by stacking these neuron-specific weights by $\widehat{\mathbf{W}}^{(m)} = [\widehat{\mathbf{w}}_1^{(m)}; \widehat{\mathbf{w}}_2^{(m)}; \ldots; \widehat{\mathbf{w}}_{D_{\text{out}}^{(m)}}^{(m)}]$.

The objective is to minimize the reconstruction error between $\widehat{\mathbf{W}}^{(m)}$ and the original weight matrix $\mathbf{W}^{(m)}$, which is computed by $\mathcal{L}(\mathbf{S}) = \frac{1}{2}\sum_{m=1}^{M} \left\| \widehat{\mathbf{W}}^{(m)} - \mathbf{W}^{(m)} \right\|_{\text{F}}^2 = \frac{1}{2}\sum_{m=1}^{M}\sum_{i=1}^{D_{\text{out}}^{(m)}} \left\| \widehat{\mathbf{w}}_i^{(m)} - \mathbf{w}_i^{(m)} \right\|_2^2$, where $\|\cdot\|_{\text{F}}$ and $\|\cdot\|_2$ denote the Frobenius and ℓ^2-norms, respectively, and $\mathbf{w}_i^{(m)}$ represents the i-th row of $\mathbf{W}^{(m)}$. The weight matrix $\mathbf{W}^{(m)}$ can be either the weights of a linear layer in Section 4.2 for compression of Llama models, or the low-rank parts (U and V) of its low-rank decomposition $\mathbf{W}^{(m)} \approx UV$ by SVD based compression such as (Li et al., 2023) in Section 4.1 for compression of the Bert models.

3.3. Fine-Tuning the NSNet via Knowledge Distillation

Although the NSNet is initialized to approximate the pre-trained LLM, an inevitable performance gap exists between the initialized NSNet and the pre-trained LLM due to inherent approximation errors. To reduce the performance gap, we propose to further fine-tune the initialized NSNet using knowledge distillation. The fine-tuning process aims to align the prediction of the NSNet with that of the original pre-trained LLM by knowledge distillation, ensuring that the NSNet retains the knowledge and capabilities of the pre-trained model while benefiting from its reduced model size.

Knowledge distillation involves transferring knowledge from a larger pre-trained teacher model to a smaller compressed student model. In this context, the off-the-shelf pre-trained LLM is the teacher model, and the NSNet is the student model. Given an input $\mathbf{x}$, the goal is to minimize the discrepancy between the outputs of the teacher and student models. The overall loss function for fine-tuning with knowledge distillation consists of two components, which are a distillation loss and a supervised loss. The overall loss function is given by $\mathcal{L}_{\text{fine-tune}} = \mathcal{L}_{\text{distill}} + \lambda\mathcal{L}_{\text{task}}$, where λ is a scale factor that balances the distillation and supervised loss terms.

The distillation loss aligns the output distribution of the student model, NSNet, with that of the teacher model, the pre-trained LLM. By using predicted probabilities over the vocabulary from the teacher model as the soft targets, the distillation loss encourages the student model to mimic the behavior of the teacher model. Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ represent the dataset containing inputs $\mathbf{x}_i$ and corresponding ground-truth outputs $\mathbf{y}_i$. The distillation loss is computed using the Kullback-Leibler (KL) divergence between the softmax outputs of the teacher and student models, that is, $\mathcal{L}_{\text{distill}} = \sum_{i=1}^N \tau^2 \text{KL}\left(\text{softmax}\left(\mathbf{z}_i^{(\text{tea})}/\tau\right) \| \text{softmax}\left(\mathbf{z}_i^{(\text{stu})}/\tau\right)\right)$ where $\mathbf{z}_i^{(\text{tea})}$ and $\mathbf{z}_i^{(\text{stu})}$ are the output probability distributions of the i-th training data for the teacher model and the student model, respectively, computed at a temperature $\tau > 1$.

The supervised loss ensures that the student model maintains high performance on the downstream tasks by minimizing the discrepancy between the model predictions and the task-specific ground-truth labels. The supervised loss can take different forms depending on the nature of the task. The supervised loss is computed by $\mathcal{L}_{\text{task}} = \frac{1}{N}\sum_{i=1}^N \ell_{\mathcal{T}}(\mathbf{y}_i, \mathbf{z}_i^{(\text{stu})})$, where $\ell_{\mathcal{T}}$ represents a task-specific loss function. For instance, $\ell_{\mathcal{T}}$ corresponds to the KL-Divergence for tasks such as sentence classification or natural language generation, while it takes the form of the ℓ^2-norm for semantic similarity measurement tasks.

4. Experiments

We evaluate the proposed Neuron Summary (NS) for compressing two different categories of LLMs, which are the Bert models (Devlin et al., 2018; He et al., 2021) and the Llama models (Touvron et al., 2023b;d). The evaluation results on the Bert models for the natural language understanding and the question answering tasks are presented in Section 4.1. More empirical and theoretical results are deferred to the appendix. The evaluation results on the Llama models are presented in Section A.2 of the appendix. We compare NS with competing baselines for compressing the Llama models on the language generation task. We perform comprehensive ablation studies in Section A.3 of the appendix. In Section A.3.1, we study the performance of NS when combined with quantization methods. In Section A.3.2, we study the performance of NSNet with different parameter sizes. Ablation study on verifying the effectiveness of the NS initialization method is performed in Section A.3.3. Additional experiment results are deferred to Section A of the appendix. In Section A.5, we compare NSNet against pruning-based methods for compressing the Llama models. In Section A.6, we compare NSNet with LoSparse without using models compressed by LoSparse as the original model. In Section A.7, we combine NS and the compression method that shares parameters across different layers for enhanced compression. In Section A.8, we assess the performance of the Llama models compressed

Table 1: Results of Compressed DeBERTaV3-base (He et al., 2021) models on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). 'Ratio' denotes the ratio of the number of parameters of the models compressed by the baseline methods to that of the original model. The compression ratio of NSNet is 90% of that of the baseline models, which applies to all tables in this paper. Results with 'N.A.' indicate the model does not converge as reported in (Li et al., 2023). The best results on each dataset are shown in bold. These conventions are followed in all the tables in this paper.

Ratio	Method	MNLI m / mm	RTE Acc	QNLI Acc	MRPC Acc / F1	QQP Acc / F1	SST-2 Acc	CoLA Mcc	STS-B P/S Corr
100%	DeBERTaV3-base (He et al., 2021)	90.5 / 90.6	82.0	94.0	89.5 / 93.3	92.4 / 89.8	95.3	69.2	91.6 / 91.1
20%	ITP (Molchanov et al., 2019)	82.8 / 82.5	N.A.	87.8	82.0 / 87.0	90.0 / 86.4	90.8	49.0	87.4 / 87.0
	LoSparse (Li et al., 2023)	84.5 / 83.8	68.0	88.6	85.0 / 89.4	90.6 / 87.2	91.7	50.0	88.6 / 88.6
	Dynamic Tying (Hay & Wolf, 2024)	84.2 / 83.2	68.4	88.8	85.2 / 89.7	90.5 / 87.4	91.6	50.8	88.8 / 88.5
	NSNet (Ours)	**85.7 / 84.4**	**69.9**	**90.4**	**86.1 / 90.7**	**91.2 / 88.2**	**92.9**	**52.2**	**89.7 / 89.6**
15%	ITP (Molchanov et al., 2019)	81.7 / 81.3	N.A.	85.4	80.5 / 86.3	89.1 / 85.2	89.3	45.8	86.8 / 86.3
	LoSparse (Li et al., 2023)	83.3 / 82.9	66.9	87.6	83.6 / 88.0	90.3 / 87.0	90.4	46.8	87.7 / 87.3
	Dynamic Tying (Hay & Wolf, 2024)	83.0 / 82.7	67.1	87.9	84.0 / 88.2	90.4 / 86.8	90.6	47.1	87.5 / 87.5
	NSNet (Ours)	**84.4 / 83.8**	**68.6**	**88.9**	**85.2 / 89.4**	**90.8 / 87.8**	**91.6**	**48.6**	**88.8 / 88.6**
10%	ITP (Molchanov et al., 2019)	79.7 / 79.6	N.A.	82.3	78.5 / 84.3	88.3 / 84.4	88.3	38.0	86.3 / 86.0
	LoSparse (Li et al., 2023)	81.7 / 81.8	66.0	86.1	82.3 / 87.4	89.5 / 86.0	89.2	40.0	87.2 / 87.0
	Dynamic Tying (Hay & Wolf, 2024)	81.5 / 80.9	66.3	86.8	82.5 / 87.4	89.8 / 86.2	89.4	40.6	87.4 / 87.5
	NSNet (Ours)	**82.8 / 82.6**	**68.0**	**87.5**	**84.0 / 88.6**	**90.4 / 87.0**	**90.5**	**42.0**	**88.3 / 88.2**

by NS by BLEU on the text generation task and prediction accuracy on the classification tasks. In Section A.9, we compare NSNet with models compressed by the baseline methods when fine-tuned under the same settings. Section A.10 provides an ablation study on regression-based initialization and knowledge distillation. In Section A.11, we demonstrate the scalability of NS on compressing larger models. A theoretical result analyzing the approximation error for the NS-Linear layer is in Section A.12. A case study about the comparison between the texts generated by NSNet and those generated by the models compressed by the baseline methods is in Section A.13. Throughout all the experiments, instead of directly compressing the original model by NSNet, we first apply NS to compress the low-rank weight matrices of the models compressed by the low-rank compression method, such as LoSparse (Li et al., 2023) or ASVD (Yuan et al., 2023). The compression ratio of the LoSparse or ASVD used by NSNet is set to be the same as the compression ratio of the baseline compression methods. Then NS is applied to compress the low-rank matrices produced by LoSparse or ASVD with a compression ratio of 90%. For example, the NSNet in Table 2, where the baseline compression ratio is 50%, actually has a compression ratio of $90\% \times 50\% = 45\%$. In Section 4.1, LoSparse (Li et al., 2023) is used as the low-rank compression method. In Section 4.2, ASVD is used as the low-rank compression method. The experiments in this paper are conducted on the Nvidia A100 GPU.

4.1. Compressing the Bert Models

4.1.1. Experiment Settings

Following the existing works on compressing the Bert models (Louizos et al., 2017; Sanh et al., 2020; Zhang et al., 2022; Li et al., 2023), we do not compress the LayerNorm and classification head. Throughout all the experiments in our paper, we set the scale factor λ from $\mathcal{L}_{\text{fine-tune}}$ = as 0.5, and the temperature τ from $\mathcal{L}_{\text{distill}}$ as 2. The compression ratio r is set to the same value for all the linear layers in the proposed NSNet. To evaluate the effectiveness of NS in compressing BERT models, we compare it against Iterative Pruning (ITP) (Molchanov et al., 2019), LoSparse (Li et al., 2023), and Dynamic Tying (Hay & Wolf, 2024).

4.1.2. Natural Language Understanding

Implementation Details. Following existing works (Li et al., 2023; Molchanov et al., 2019; Sanh et al., 2020), we evaluate the performance of the NSNet when compressing the pre-trained DeBERTaV3-base model (He et al., 2021) for the natural language understanding task on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). The GLUE dataset consists of two single-sentence classification tasks, SST-2 (Socher et al., 2013) and CoLA (Warstadt et al., 2019), three similarity and paraphrase tasks, MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017), and QQP, and four natural language inference tasks, MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016a), RTE (Dagan et al., 2007; 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), and WNLI (Levesque et al., 2012). Following previous works (Li et al., 2023), we exclude WNLI in the experiments due to its small dataset size, specialized nature requiring complex pronoun disambiguation, and performance anomalies where simpler models may outperform advanced ones. For single-sentence classification tasks such as SST-2 and CoLA, we employ accuracy and the Matthews Correlation Coefficient (MCC), respectively, to

measure model performance. For similarity and paraphrase tasks like MRPC, STS-B, and QQP, we consider both accuracy and the F1 Score to capture the nuances of paraphrase detection and semantic similarity assessment. In natural language inference tasks, including MNLI, QNLI, and RTE, accuracy serves as the primary metric, providing a clear measure of the model's ability to correctly infer relationships between sentences. In MNLI, matched accuracy, denoted as 'm', measures performance on test data from the same domains as the training set, while mismatched accuracy, denoted as 'mm', evaluates generalization to unseen domains.

We select the learning rates of the fine-tuning process for different tasks from $\{1\times10^{-5}, 2.5\times10^{-5}, 5\times10^{-5}, 7.5\times10^{-5}, 1\times10^{-4}\}$. AdamW is used as the optimizer. The batch size is set to 128. The number of fine-tuning epochs is set to 5.

Results. We compare our method with baseline approaches across various compression ratios. The results are shown in Table 1. It is observed that our NS achieves the best performance compared with all competing methods on all the datasets of GLUE under all compression ratios. For example, when the compression ratio of the baseline method is 20%, the NSNet achieves an accuracy of 69.9% on the RTE dataset, which surpasses the best-performing baseline, Dynamic Tying, by 1.5%. In addition, NS also significantly outperforms the competing baselines with the smallest compression ratio of 10%. For example, when the compression ratio of the baseline model is 10%, the NSNet achieves an accuracy of 68.0% on the RTE dataset, which surpasses the best-performing baseline, Dynamic Tying, by 1.7%. In addition, we compare the inference time in milliseconds (ms) per sample measured and the model size (millions of parameters) on the GLUE benchmark using one Nvidia A100 GPU in Figure 3 of the appendix. It is observed that NSNet always exhibits faster inference speed and smaller model size than the competing baseline models with the same level of performance.

4.1.3. Question Answering

Implementation Details. We evaluate the performance of the NSNet for performing question answering on SQuADv1.1, (Rajpurkar et al., 2016a). In SQuADv1.1, question answering is formulated as a sequence labeling task, where each token is assigned a probability of being the beginning or end of the answer span. Exact match (EM) and F1 are used as the evaluation metric for this task. Similar to the experiments on the GLUE, we do not compress the LayeNorm and the classification layer. Following (Yuan et al., 2023) for the fine-tuning of the models, we select the best learning $\{1\times10^{-5}, 3\times10^{-5}, 5\times10^{-5}, 8\times10^{-5}\}$. The fine-tuning is performed by using AdamW as the optimizer and the batch size is set to 32.

Results. We compare the performance of the NSNet with models compressed by the competing baselines under different compression ratios. The results are shown in Table 2 of Section A.1 of the appendix. It is observed that NSNet consistently outperforms the competing baseline methods under all compression ratios. Similar to our results on GLUE, NS is especially effective with low compression ratios. For example, when the compression ratio of the baseline model is 10%, NSNet outperforms the best-performing baseline, Dynamic Tying, by 1.5% in exact match. The superior performance of the NSNet is maintained when compressing the BERT-Basse model. For example, when the compression ratio of the baseline model is 10%, NSNet outperforms the best-performing baseline, LoSparse, by 1.3% in exact match.

4.2. Compressing Llama Models

Following existing works (Wang et al., 2025), we evaluate the effectiveness of Neuron Summary (NS) for compressing the Llama-2 models (Touvron et al., 2023b;d), which include configurations with 7 billion and 13 billion parameters. The evaluation is performed on two datasets, which are the Wikitext-2 and the Penn Treebank (PTB) (Marcus et al., 1993). Perplexity is used as the metric to evaluate the language generative capability of the LLMs. The detailed implementation details and experimental results are deferred to Section A.2 of the appendix.

5. Conclusion

We introduce Neuron Summary (NS), a novel compression method that replaces linear layers with NS-Linear layers, reducing model size while maintaining performance. The compressed model, NSNet, initializes via regression-based optimization and fine-tunes using knowledge distillation. Experiments on various NLP tasks show that NS outperforms existing compression methods, especially at high compression ratios. In addition, NS can be combined with existing LLM compression methods, such as quantization, leading to even more efficient LLM deployment.

Impact Statement

This paper presents work whose goal is to advance the field of model compression for Large Language Models by learnable weight sharing.

References

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-

based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2357–2367. Association for Computational Linguistics, 2019.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Bhojanapalli, S., Chakrabarti, A., Veit, A., Lukasik, M., Jain, H., Liu, F., Chang, Y.-W., and Kumar, S. Leveraging redundancy in attention with reuse transformers. *arXiv preprint arXiv:2110.06821*, 2021.

Bisk, Y., Zellers, R., Le Bras, R., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In Quiñonero-Candela, J., Dagan, I., Magnini, B., and d'Alché Buc, F. (eds.), *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pp. 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33428-6.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2007.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. In *International Conference on Learning Representations*, 2019.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337, 2023.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, Prague, June 2007. Association for Computational Linguistics.

Golub, G. H., Hoffman, A., and Stewart, G. W. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88: 317–327, 1987.

Gu, Y., Dong, L., Wei, F., and Huang, M. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Hay, T. D. and Wolf, L. Dynamic layer tying for parameter-efficient transformers. In *The Twelfth International Conference on Learning Representations*, 2024.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2021.

Hsu, Y.-C., Hua, T., Chang, S., Lou, Q., Shen, Y., and Jin, H. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Huang, Y., Chen, Y., Yu, Z., and McKeown, K. In-context learning distillation: Transferring few-shot learning ability of pre-trained language models. *arXiv preprint arXiv:2212.10670*, 2022.

Jiang, Y., Chan, C., Chen, M., and Wang, W. Lion: Adversarial distillation of proprietary large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3134–3154, 2023.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Levesque, H., Davis, E., and Morgenstern, L. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., and Zhao, T. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20336–20350. PMLR, 23–29 Jul 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

Lin, S., Hilton, J., and Askell, A. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3214–3253. Association for Computational Linguistics, 2022.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Lv, X., Zhang, P., Li, S., Gan, G., and Sun, Y. Lightformer: Light-weight transformer using svd-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10323–10335, 2023.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023a.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023b.

Magister, L. C., Mallinson, J., Adamek, J. D., Malmi, E., and Severyn, A. Teaching small language models to reason. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391. Association for Computational Linguistics, 2018.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Muralidharan, S., Sreenivas, S. T., Joshi, R., Chochowski, M., Patwary, M., Shoeybi, M., Catanzaro, B., Kautz, J., and Molchanov, P. Compact language models via pruning and knowledge distillation. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on*

Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024.

Qiu, R., Eldebiky, A., Li Zhang, G., Yin, X., Zhuo, C., Schlichtmann, U., and Li, B. Oplixnet: Towards area-efficient optical split-complex networks with real-to-complex data assignment and knowledge distillation. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2024.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016a. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016b.

Reid, M., Marrese-Taylor, E., and Matsuo, Y. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. *arXiv preprint arXiv:2101.00234*, 2021.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Sanh, V., Wolf, T., and Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Takase, S. and Kiyono, S. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*, 2021.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023b.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023c.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023d.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Wang, J., Chen, Y.-G., Lin, I.-C., Li, B., and Zhang, G. L. Basis sharing: Cross-layer parameter sharing for large language model compression. In *International Conference on Learning Representations*, 2025.

Wang, X., Zheng, Y., Wan, Z., and Zhang, M. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.

Wu, Y., Kan, S., Zeng, M., and Li, M. Singularformer: Learning to decompose self-attention to linearize the complexity of transformer. In *International Joint Conference on Artificial Intelligence*, pp. 4433–4441, 2023.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=09iOdaeOzp.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099, 2023.

Xiao, T., Li, Y., Zhu, J., Yu, Z., and Liu, T. Sharing attention weights for fast transformer. *International Joint Conference on Artificial Intelligence*, 2019.

Yuan, Z., Shang, Y., Song, Y., Wu, Q., Yan, Y., and Sun, G. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Zhang, Q., Zuo, S., Liang, C., Bukharin, A., He, P., Chen, W., and Zhao, T. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pp. 26809–26823. PMLR, 2022.

Zhao, Y., Lin, C.-Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., and Kasikci, B. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

A. Additional Experiment Results

A.1. Question Answering

Table 2: Results of Compressed DeBERTaV3-base and BERT-base on SQuAD v1.1. The exact match (EM) and F1 are reported in the table below. For example, A/B represents EM/F1.

Ratio	10%	20%	30%	40%	50%	100%
DeBERTaV3-base	-	-	-	-	-	87.7 / 93.5
ITP (Molchanov et al., 2019)	70.9 / 80.3	75.0 / 83.9	78.2 / 86.2	80.5 / 87.5	81.5 / 89.6	-
LoSparse (Li et al., 2023)	72.9 / 82.8	76.8 / 85.8	80.2 / 88.0	82.1 / 89.4	82.3 / 90.3	-
Dynamic Tying (Hay & Wolf, 2024)	72.5 / 82.6	76.4 / 85.3	80.4 / 88.0	82.2 / 89.2	82.1 / 90.0	-
NSNet (Ours)	**74.0 / 83.6**	**77.6 / 86.4**	**81.4 / 88.8**	**82.8 / 89.8**	**83.0 / 90.8**	-
BERT-base	-	-	-	-	-	80.9 / 88.2
ITP (Molchanov et al., 2019)	62.5 / 74.2	66.8 / 78.0	72.3 / 82.4	74.5 / 84.2	76.0 / 85.1	-
LoSparse (Li et al., 2023)	65.2 / 76.8	69.7 / 80.4	73.0 / 82.9	74.6 / 84.2	75.8 / 85.1	-
Dynamic Tying (Hay & Wolf, 2024)	64.7 / 76.3	69.9 / 80.3	72.6 / 82.4	74.8 / 84.0	76.0 / 85.0	-
NSNet (Ours)	**66.5 / 77.6**	**70.9 / 81.4**	**73.9 / 83.5**	**75.6 / 84.6**	**76.9 / 85.6**	-

A.2. Compressing Llama Models

Following existing works (Wang et al., 2025), we evaluate the effectiveness of Neuron Summary (NS) for compressing the Llama-2 models (Touvron et al., 2023b;d), which include configurations with 7 billion and 13 billion parameters. The evaluation is performed on two datasets, which are the Wikitext-2 and the Penn Treebank (PTB) (Marcus et al., 1993). Perplexity is used as the metric to evaluate the language generative capability of the LLMs.

Implementation Details. Following existing LLM compression methods (Ma et al., 2023b), we use LoRA fine-tuning to fine-tune the weights of the compressed LLMs. Details on the LoRA fine-tuning are deferred to Section A.4 of this appendix. We apply NS to compress the Llama models by replacing all their linear layers with the NS-Linear layers without SVD compression. The compression ratio r is set to the same value for all the linear layers in the proposed NSNet. To evaluate the effectiveness of NS in compressing Llama models, we compare it against Adaptive SVD (ASVD) (Yuan et al., 2023), SVD-based Layer-wise Model Compression (SVD-LLM) (Wang et al., 2024), and Basis Sharing (Wang et al., 2025).

Results. We evaluate the performance of Llama-2-7B and Llama-2-13b compressed by NS and the competing baselines, with the compression ratios of the baseline models ranging from 60% to 90% on all the datasets with a step size of 10%. The results are shown in Table 3. It is observed that NSNet compressed from both the Llama-2-7B and Llama-2-13b consistently outperforms all the competing baselines across all the compression ratios, with even faster inference speed. More importantly, NSNet exhibits significant advantages over the competing baselines when fewer parameters are preserved in the compressed models. For instance, when the compression ratio of the baseline model is 60%, NSNet outperforms the best-performing baseline, Basis Sharing, by 15.37 in perplexity on the PTB dataset, with even faster inference speed. In addition, we compare the inference time in milliseconds (ms) per sample measured and the model size (billions of parameters) on the entire GLUE benchmark using one Nvidia A100 GPU in Figure 2. It is observed that NSNet always exhibits faster inference speed and smaller model size than the competing baseline models with the same level of performance.

Table 3: Comparison of the perplexity of the models compressed from the Llama-2-7b and the Llama-2-13b on Wikitext-2 and PTB.

Model	Dataset	Ratio	SVD-LLM (Wang et al., 2024)	ASVD (Yuan et al., 2023)	Basis Sharing (Wang et al., 2025)	**NSNet (Ours)**
Llama-2-7b	Wikitext-2	90%	7.27	5.74	6.52	**5.60**
		80%	8.38	6.86	7.77	**6.14**
		70%	10.67	10.62	9.69	**9.51**
		60%	16.14	19.12	13.62	**13.25**
	PTB	90%	43.82	32.63	25.14	**19.67**
		80%	75.55	114.70	60.00	**55.32**
		70%	110.28	140.96	97.40	**80.26**
		60%	204.34	272.65	195.95	**180.58**
Llama-2-13b	Wikitext-2	90%	5.94	5.03	5.47	**4.93**
		80%	6.66	5.77	5.90	**5.62**
		70%	8.00	7.82	7.96	**7.21**
		60%	10.78	13.18	11.25	**10.12**
	PTB	90%	35.85	34.03	20.96	**19.25**
		80%	52.06	59.68	42.05	**38.42**
		70%	86.40	79.53	74.36	**72.28**
		60%	110.85	110.65	102.45	**95.79**

A.3. Ablation Study

A.3.1. Integrating NS with Quantization

The design of NS is orthogonal to other widely used LLM compression methods, such as quantization. Following existing works on compressing the Llama models (Yuan et al., 2023), we integrate NS into quantization-based LLM compression methods that are widely recognized by the community to examine how NS could further enhance their performance. The evaluation is performed for combing NS with two quantization methods, which are the Round-To-Nearest (RTN) and the Activation-aware Weight Quantization (AWQ) (Lin et al., 2023). We note that our study focuses on establishing the orthogonal property of NS to these basic quantization methods. Future work could extend this investigation to more advanced quantization techniques and other LLM compression approaches. The results are shown in Table 4. It is observed that NSNet quantized by both methods can still maintain a reasonably good performance with even less memory consumption, showing the potential of NS to complement existing quantization-based compression techniques. In addition, we compare the performance of NSNet combined with different quantization methods with Basis Sharing combined with different quantization methods. It is observed that NSNet sacrifices much less performance than Basis Sharing when combined with quantization. For example, AWQ increases the perplexity of the model compressed by Basis Sharing with a compression ratio of 60% by 1.10. In contrast, AWQ only increases the perplexity of the NSNet with a compression ratio of 60% by 0.70.

Table 4: Comparison of the perplexity between Basis Sharing and NSNet when combined with quantization methods for compressing Llama-2-7b. The evaluation is performed on the Wikitext-2.

Ratio	FP16 (Original)		INT 8 (RTN)		INT 4 (AWQ)	
	Basis Sharing (Wang et al., 2025)	NSNet	Basis Sharing (Wang et al., 2025)	NSNet	Basis Sharing (Wang et al., 2025)	NSNet
90%	6.52	**5.60**	6.70	**5.62**	6.92	**5.79**
80%	7.77	**6.14**	7.95	**6.18**	8.05	**6.36**
70%	9.69	**9.51**	9.84	**9.53**	10.39	**9.92**
60%	13.62	**13.25**	13.90	**13.32**	14.72	**13.95**

A.3.2. Ablation Study on the Performance of the NSNet of Different Model Sizes

We perform a comparative study evaluating the effect of model size on the performance of NS in compressing LLMs. We evaluate how models compressed by NSNet and the competing baselines perform as the number of parameters decreases. The comparison is performed on the MNLI task in the GLUE dataset, with compressed models ranging from 20M to 150M parameters. The NSNet is directly compressed from the DeBERTaV3-base by replacing all the linear layers with the NS-Linear layers. The evaluation results are illustrated in Figure 3 (a). It is observed that NSNet outperforms models compressed by other methods more significantly as the model size decreases.

A.3.3. Ablation Study on the Initialization of the NSNet and the Fine-Tuning of the NSNet

To verify the effectiveness of the proposed novel initialization method and the fine-tuning method of the NSNet, we perform an ablation study by evaluating the performance of the initialized NSNet without fine-tuning. The ablation study is performed on the MNLI task in the GLUE benchmark. We perform the evaluation for NSNet compressed from the DeBERTaV3-base for compression ratios of 10%, 15%, and 20%. The results are shown in Table 5. It is observed that the initialized NSNet still achieves a reasonable performance on the downstream task without fine-tuning. Moreover, fine-tuning with the knowledge distillation significantly reduces the performance gaps between the initialized compressed models and the pre-trained LLMs.

Table 5: Ablation study on the initialization of the NSNet and the fine-tuning of the NSNet. The NSNet without Fine-tuning denotes the NSNet initialized with the proposed NS initialization method in Section 3.2. The matched accuracy and the mismatched accuracy are reported in the table below. For example, A/B represents matched/mismatched accuracy.

Ratio	10%	15%	20%	100%
DeBERTaV3-base	-	-	-	90.5 / 90.6
NSNet without Fine-tuning	83.0 / 82.4	81.8 / 82.2	81.0 / 81.2	-
NSNet	85.7 / 84.4	84.4 / 83.8	82.8 / 82.6	-

A.4. LoRA Fine-Tuning Settings for Models Compressed from the Llama-2 Models

LoRA (Hu et al., 2021) optimizes the fine-tuning of LLMs by adding a small set of trainable parameters while the original model parameters remain frozen. This is achieved by constraining the update of a pre-trained weight matrix W_0 to a low-rank structure, represented as $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank r is significantly smaller than the minimum of d and k. During training, W_0 is kept static, and gradient updates are applied only to A and B. The adaptation is applied as $h = W_0x + \Delta Wx = W_0x + BAx$, ensuring that W_0 and $\Delta W = BA$ are multiplied by the same input x, and their output vectors are summed coordinate-wise to yield the modified output h. We set the rank adjustment parameter of LoRA fine-tuning to 8, which determines the rank of the low-rank matrices used for adaptation. The LoRA fine-tuning scaling factor, which decides the adaptation impact of the fine-tuning, is set to 32.

A.5. Comparison with Llama Pruning Methods

We conduct additional experiments to compare NSNet with pruning-based methods, namely ShearedLLaMA (Xia et al., 2024) and Minitron (Muralidharan et al., 2024). To ensure a fair comparison, we adopt the same fine-tuning settings for all methods. Specifically, we apply the same task objective and knowledge distillation from the original model, along with the LoRA fine-tuning strategy, to the models compressed by ShearedLLaMA and Minitron. Table 6 presents the perplexity results on Wikitext-2 and PTB datasets across different sparsity ratios. As shown, NSNet consistently achieves lower perplexity scores than both ShearedLLaMA and Minitron on LLaMA-2-7B and LLaMA-2-13B backbones, under identical fine-tuning conditions.

Table 6: Perplexity comparison between NSNet and pruning-based methods for LLaMA-2-7B and LLaMA-2-13B on WikiText-2 and PTB.

Model	Dataset	Ratio	ShearedLLaMA	Minitron	NSNet (Ours)
LLaMA-2-7B	WikiText-2	90%	6.10	6.24	**5.60**
		80%	6.80	6.99	**6.14**
		70%	10.20	10.44	**9.51**
		60%	14.10	13.95	**13.25**
	PTB	90%	20.45	20.18	**19.67**
		80%	56.24	57.90	**55.32**
		70%	88.10	85.23	**80.26**
		60%	185.92	187.14	**180.58**
LLaMA-2-13B	WikiText-2	90%	5.78	5.48	**4.93**
		80%	6.24	6.32	**5.62**
		70%	7.80	7.95	**7.21**
		60%	10.79	10.64	**10.12**
	PTB	90%	19.95	19.76	**19.25**
		80%	40.42	39.10	**38.42**
		70%	75.20	77.90	**72.28**
		60%	98.74	99.31	**95.79**

A.6. Comparison with LoSparse

To ensure a fair comparison with LoSparse (Li et al., 2023), we compress NSNet directly from the original pretrained models instead of reusing models compressed by LoSparse. We report Exact Match (EM) and F1 scores on the SQuAD dataset, with each cell formatted as EM/F1. The results are presented in Table 7. It is observed that NSNet outperforms models compressed by LoSparse consistently across various compression ratios. Furthermore, combining NSNet with LoSparse yields additional performance gains, achieving the best results in all settings.

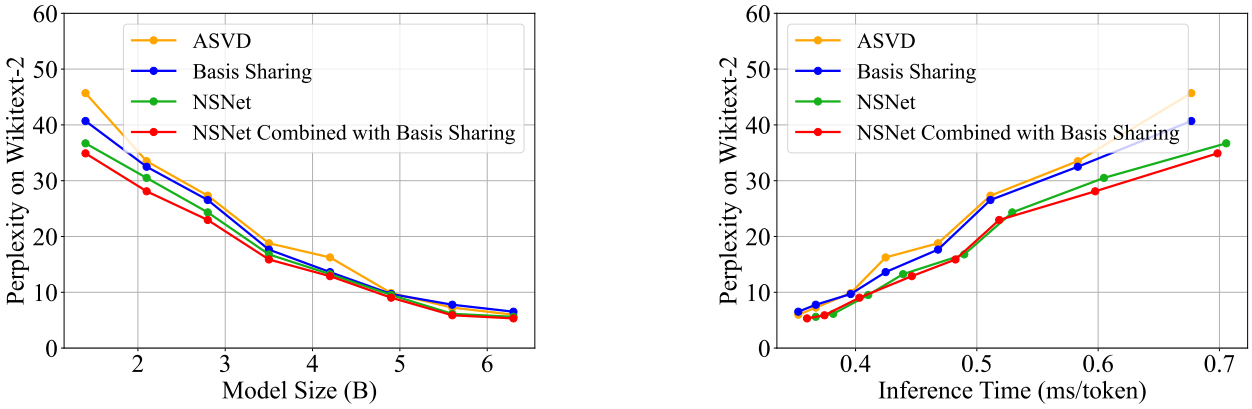
A.7. Combining NS with the Compression Method Sharing Parameters across Different Layers

Most existing parameter sharing methods for compressing LLMs reduce the model size by reusing weight matrices across different layers. Although NS is designed for parameter sharing among neurons within a single layer, it can be seamlessly integrated with cross-layer parameter-sharing methods to achieve further compression. In this Section, we study the per-

Table 7: Comparison between NSNet, LoSparse, and their combination on SQuAD under different sparsity levels. Each cell shows EM/F1 score.

Model	10%	20%	30%	40%	50%
LoSparse (DeBERTaV3-base)	72.9/82.8	76.8/85.8	80.2/88.0	82.1/89.4	82.3/90.3
NSNet (DeBERTaV3-base)	73.7/83.4	77.5/86.2	81.1/88.5	82.6/89.9	82.8/90.8
NSNet + LoSparse (DeBERTaV3-base)	**74.0/83.6**	**77.6/86.4**	**81.4/88.8**	**82.8/89.8**	**83.0/90.8**
LoSparse (BERT-base)	65.2/76.8	69.7/80.4	73.0/82.9	74.6/84.2	75.8/85.1
NSNet (BERT-base)	66.2/76.3	70.5/81.0	73.7/83.2	75.3/84.5	76.5/85.4
NSNet + LoSparse (BERT-base)	**66.5/77.6**	**70.9/81.4**	**73.9/83.5**	**75.6/84.6**	**76.9/85.6**

formance of NSNet when combined with Basis Sharing. To combine NS with the Basis Sharing, we apply NS to compress the weight matrices of the models compressed by Basis Sharing. To obtain a compact model with a compression ratio of r_0 by integrating Basis Sharing and NSNet, we begin by applying Basis Sharing to reduce the model size by half of the total intended reduction, resulting in an intermediate compression ratio of $0.5 + r_0/2$. Next, we apply NS to compress the weight matrices in the Basis Sharing model to reach the target compression ratio of r_0. For example, to obtain the NSNet combined with Basis Sharing with a compression ratio of 80%, we first compress the Llama-3 model using Basis Sharing with a compression ratio of 90% and then compress the weight matrices in the compressed Basis Sharing model with NS. In our experiments, we use Llama-3 7B as the base model. The comparative performance in terms of perplexity on the Wikitext-2 dataset is depicted in Figure 2, where models compressed by various methods are compared across compression ratios from 0.2 to 0.9, with a step size of 0.1. It is observed that combining NSNet with Basis Sharing further enhances the performance of compressed models of the same size as those compressed by NS or Basis Sharing alone, demonstrating the promise of combining NS with cross-layer parameter-sharing methods. The performance improvement from the combination of NS and Basis Sharing becomes increasingly significant as the target model size is further reduced.

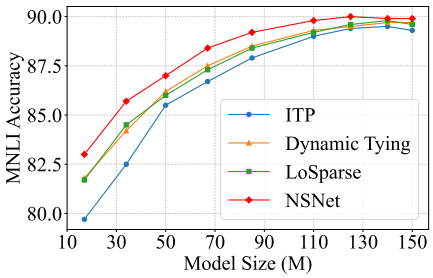


(a) Perplexity vs. Model Size (B).

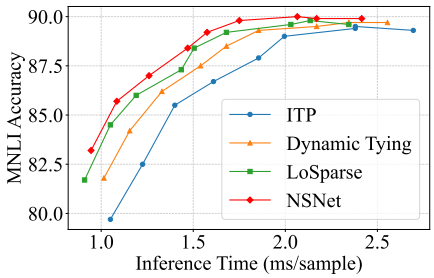
(b) Performance vs. Inference Time (ms/token).

Figure 2: Comparison of NSNet models compressed from Llama-7B against competing baselines. Figure (a) illustrates the performance under different model sizes, while Figure (b) illustrates the trade-off between the performance and the inference time.

In addition, we compare the inference time in milliseconds (ms) per sample measured and the model size (millions of parameters) on the GLUE benchmark using one Nvidia A100 GPU in Figure 3. It is observed that NSNet always exhibits faster inference speed and smaller model size than the competing baseline models with the same level of performance.



(a) Accuracy vs. Model Size (M).



(b) Performance vs. Inference Time (ms/sample).

Figure 3: Comparison of NSNet models compressed from DeBERTaV3-base against competing baselines. Figure (a) illustrates the performance under different model sizes, while Figure (b) illustrates the trade-off between the performance and the inference time.

A.8. Evaluation on Text Generation and Classification Tasks

Following prior works such as ASVD (Yuan et al., 2023), Basis Sharing (Wang et al., 2025), and SVD-LLM (Wang et al., 2024), we have evaluated the language modeling capabilities of compressed LLaMA models on the Wikitext-2 and PTB datasets using perplexity in Table 3. To further demonstrate the effectiveness of NSNet, we conduct additional experiments on two other tasks, which are text generation and classification. For text generation, we evaluate the compressed LLaMA-2-7B models on the TruthfulQA dataset (Lin et al., 2022), using BLEU score as the evaluation metric. For classification, we evaluate model performance on six benchmark datasets, which are OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), ARC-E (Clark et al., 2018), PIQA (Bisk et al., 2020), and MathQA (Amini et al., 2019). We report the average accuracy across these six datasets as the final classification score. All methods are fine-tuned with the same settings as NSNet, using a combination of task-specific objectives and knowledge distillation from the original model.

Table 8: Evaluation of compressed LLaMA-2-7B models on TruthfulQA (BLEU) and average classification accuracy across six benchmark datasets.

Task	Ratio	SVD-LLM	ASVD	Basis Sharing	NSNet (Ours)
TruthfulQA (BLEU)	90%	0.29	0.25	0.26	**0.30**
	80%	0.28	0.24	0.25	**0.29**
	70%	0.25	0.21	0.23	**0.27**
	60%	0.24	0.20	0.20	**0.25**
Classification	90%	0.55	0.52	0.51	**0.56**
	80%	0.54	0.48	0.49	**0.55**
	70%	0.47	0.44	0.47	**0.51**
	60%	0.42	0.39	0.42	**0.45**

A.9. Comparison Baseline Models Fine-Tuned under the Same Settings as NSNet

To ensure a fair comparison with baseline methods, we have conducted additional experiments where all models, including NSNet and the baselines, are fine-tuned under identical settings. We apply both the task objective and knowledge distillation (KD) to fine-tune each model for the same number of epochs. As shown in Table 9, NSNet significantly outperforms the baseline methods across all compression ratios for both DeBERTaV3-base and BERT-base backbones, under the same fine-tuning settings. The results for compressing LLaMA-2-7b and LLaMA-2-13b on WikiText-2 and PTB are shown in Table 10. It is observed that NSNet consistently outperforms baseline methods across different compression ratios on both datasets and model sizes.

A.10. Ablation Study on Regression-based Initialization and Knowledge Distillation

We conduct an ablation study to evaluate the impact of the proposed regression-based initialization and knowledge distillation in NSNet. The experiments are conducted on compressing the LLaMA-2-7B model at a compression ratio of 60%. For

Table 9: Exact Match / F1 scores comparison for the BERT models compressed under the same fine-tuning settings.

Model	10%	20%	30%	40%	50%
ITP (DeBERTaV3-base)	71.8 / 81.5	75.9 / 84.8	79.6 / 87.2	81.3 / 88.4	81.8 / 89.8
LoSparse (DeBERTaV3-base)	73.1 / 83.0	77.0 / 86.0	80.4 / 88.1	82.2 / 89.4	82.4 / 90.3
Dynamic Tying (DeBERTaV3-base)	72.9 / 82.9	76.8 / 85.6	80.5 / 88.1	82.2 / 89.3	82.3 / 90.2
NSNet (DeBERTaV3-base)	**74.0 / 83.6**	**77.6 / 86.4**	**81.4 / 88.8**	**82.8 / 89.8**	**83.0 / 90.8**
ITP (BERT-base)	63.1 / 75.1	68.0 / 79.5	72.8 / 82.7	74.8 / 84.3	76.2 / 85.2
LoSparse (BERT-base)	65.5 / 77.0	69.9 / 80.6	73.1 / 83.0	74.7 / 84.4	76.0 / 85.2
Dynamic Tying (BERT-base)	65.2 / 76.7	70.0 / 80.7	72.9 / 82.7	74.9 / 84.3	76.1 / 85.1
NSNet + LoSparse (BERT-base)	**66.5 / 77.6**	**70.9 / 81.4**	**73.9 / 83.5**	**75.6 / 84.6**	**76.9 / 85.6**

Table 10: Perplexity comparison for the LLaMA models compressed under the same fine-tuning settings.

Model	Dataset	Ratio	SVD-LLM	ASVD	Basis Sharing	NSNet (Ours)
LLaMA-2-7b	WikiText-2	90%	7.27	5.74	6.52	**5.60**
		80%	8.38	6.86	7.77	**6.14**
		70%	10.67	10.62	9.69	**9.51**
		60%	16.14	19.12	13.62	**13.25**
	PTB	90%	43.82	32.63	25.14	**19.67**
		80%	75.55	114.70	60.00	**55.32**
		70%	110.28	140.96	97.40	**80.26**
		60%	204.34	272.65	195.95	**180.58**
LLaMA-2-13b	WikiText-2	90%	5.94	5.03	5.47	**4.93**
		80%	6.66	5.77	5.90	**5.62**
		70%	8.00	7.82	7.96	**7.21**
		60%	10.78	13.18	10.74	**10.12**
	PTB	90%	35.85	34.03	20.96	**19.25**
		80%	52.06	59.68	42.05	**38.42**
		70%	86.40	79.53	74.36	**72.28**
		60%	90.85	110.65	102.45	**95.79**

the ablation without regression-based initialization, each linear layer is initialized by keeping the first 60% of the original neurons in that layer. As shown in Table 11, the regression-based initialization plays a crucial role in improving the performance of the compressed model. Moreover, fine-tuning with knowledge distillation further enhances the performance of the model initialized with the regression-based method, reducing perplexity from 13.59 to 13.25.

A.11. Scalability of NS for Compressing Larger Models

The proposed NSNet is designed to scale efficiently to larger language models. In addition to our evaluations on 7B and 13B models, we assess the scalability of NSNet by compressing the LLaMA-30B model under the same fine-tuning settings used for baseline methods. As shown in Table 12, NSNet continues to outperform other compression methods on both the WikiText-2 and PTB datasets. These results confirm that NSNet remains effective even at the scale of 30B-parameter models.

A.12. Theoretical Analysis for the Approximation Error of Using NS to Compress a Linear Layer

We denote the neural summary $\mathbf{S} \in \mathbb{R}^K$ which is used to compresses the weight matrix $\mathbf{W} \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ of a linear layer. We use the initialization scheme described in Section 3.2 to find the NS $\mathbf{S}$ by minimizing the NS approximation error, which is the the mean squared error (MSE) between the weight matrix reconstructed from the NS and the original weight matrix $\mathbf{W}$. To this end, we denote by $\{\widehat{\mathbf{w}}_i\}_{i=1}^{D_{\text{out}}}$ the extracted neurons from the NS, and $\{\widehat{\mathbf{w}}_i\}_{i=1}^{D_{\text{out}}}$ the original neurons in

Table 11: Ablation study on regression-based initialization and knowledge distillation for NSNet. Evaluation is based on perplexity on WikiText-2.

Model	WikiText-2
LLaMA-2-7B	5.47
NSNet w/o Regression-Based Initialization	18.45
NSNet w/o KD	13.59
NSNet (Full)	13.25

Table 12: Perplexity comparison on LLaMA-30B using WikiText-2 and PTB.

Dataset	SVD-LLM	ASVD	Basis Sharing	NSNet (Ours)
WikiText-2	5.60	5.75	5.42	**5.14**
PTB	19.64	20.12	19.95	**18.92**

the weight matrix $\mathbf{W}$. We define

$$\Omega_j \coloneqq \{i\colon \mathbf{S}_j \text{ appears in } \widehat{\mathbf{w}}_i\}\,, c_j \coloneqq |\Omega_j|\,. \tag{1}$$

That is, Ω_j is the set of the indices of the reconstructed neurons where the element $\mathbf{S}_j$ of the NS is an element of at least one of these reconstructed neurons. We use $\widehat{\mathbf{w}}_{i,j}$ to denote the element of the reconstructed neuron $\widehat{\mathbf{w}}_i$ which is equal to $\mathbf{S}_j$, and use $\mathbf{w}_{i,j}$ to denote the element of $\mathbf{w}_i$ co-located as $\widehat{\mathbf{w}}_{i,j}$. The NS approximation error used in Section 3.2 is defined as

$$\text{Err}(\mathbf{S}) \coloneqq \sum_{i=1}^{D_{\text{out}}} \|\mathbf{w}_i - \widehat{\mathbf{w}}_i\|_2^2, \tag{2}$$

which is the sum of the squared ℓ^2-distance between all the reconstructed neurons $\{\widehat{\mathbf{w}}_i\}_{i=1}^{D_{\text{out}}}$ and the original neurons $\{\mathbf{w}_i\}_{i=1}^{D_{\text{out}}}$ in the weight matrix $\mathbf{W}$. We have the following theorem stating the optimal NS approximation error and its upper bound.

Theorem A.1. The NS approximation error can be expressed as

$$\text{Err}(\mathbf{S}) = \sum_{j=1}^{K} \sum_{i\in\Omega(j)} \left(\mathbf{w}_{i,j} - \mathbf{S}_j\right)^2. \tag{3}$$

Moreover, the optimal NS $\mathbf{S}^* \in \mathbb{R}^K$ which minimizes $\text{Err}(\mathbf{S})$ is

$$\mathbf{S}_j^* = \frac{\sum_{t\in\Omega_j} \mathbf{w}_{t,j}}{c_j}, \tag{4}$$

with the corresponding minimum NS approximation error being

$$\text{Err}(\mathbf{S}^*) = \sum_{j=1}^{K} \sum_{i\in\Omega(j)} \left(\mathbf{w}_{i,j} - \mathbf{S}_j^*\right)^2. \tag{5}$$

Furthermore, we have $\text{Err}(\mathbf{S}^*) \leq \|\text{vec}\,(\mathbf{W})\|_2^2$, where $\text{vec}\,(\mathbf{W})$ denotes the vectorization of all the elements of the matrix $\mathbf{W}$.

Proof. First of all, it can be verified from (2) that the NS approximation error can be expressed as

$$\text{Err}(\mathbf{S}) = \sum_{j=1}^{K} \sum_{i\in\Omega(j)} \left(\mathbf{w}_{i,j} - \mathbf{S}_j\right)^2, \tag{6}$$

which follows from observing that the element $\mathbf{S}_j$ is equal to all the elements of $\{\widehat{\mathbf{w}}_{i,j}\}_{i\in\Omega_j}$, which in turn are used to approximate $\{\mathbf{w}_{i,j}\}_{i\in\Omega_j}$.

It follows that (6) can be minimized by finding the stationary point of $\text{Err}(\mathbf{S})$, and the optimal NS $\mathbf{S}^* \in \mathbb{R}^K$ which minimizes $\text{Err}(\mathbf{S})$ is

$$\mathbf{S}_j^* = \frac{\sum_{t\in\Omega_j} \mathbf{w}_{t,j}}{c_j}, \quad \forall j \in [K], \tag{7}$$

and the corresponding minimum NS approximation error is

$$\text{Err}(\mathbf{S}^*) = \sum_{j=1}^{K} \sum_{i\in\Omega(j)} \left(\mathbf{w}_{i,j} - \mathbf{S}_j^*\right)^2. \tag{8}$$

It can also be verified that $\text{Err}(\mathbf{S}^*) \leq \|\text{vec}\,(\mathbf{W})\|_2^2$, which follows from the definition of the minimum NS approximation error and (8).

□

A.13. Case Study on the Text Generated by NSNet

Table 13 presents a qualitative case study comparing text generated by NSNet and models compressed by competing compression methods from LLaMA-2-7B at an 80% compression ratio. Given the prompt "**What is the responsibility of the AI assistant?**", each model generates a response for a comparative evaluation of fluency, informativeness, and coherence. The uncompressed LLaMA-2-7B model produces a verbose yet coherent response with detailed implementation options. The Basis Sharing model demonstrates simplicity and ease of understanding but lacks technical depth and precision. The ASVD model performs relatively well, generating content that is both structured and informative, though some redundancy is observed. In contrast, NSNet achieves the best overall quality by maintaining high fluency, clear semantic structure, and rich technical content, closely aligning with the original model while using significantly fewer parameters.

Table 13: Case Study on the Text Generated by NSNet and the Baseline Models Compressed from LLaMA-2-7B. The input prompt is in bold, and the text following the prompt is the generated sentence. The study is performed for models with a compression ratio of 80%.

Models	Generated Text
LLaMA-2-7B	**What is the responsibility of the AI assistant?** The responsibility of an AI assistant is to take care of the basic, repetitive, and administrative tasks. If you don't have time for these things, then your AI assistant can do them for you. They can be used to schedule meetings, set reminders, and even take notes during meetings. There are a few different ways to build an AI assistant. The first is to use a pre-built solution. These are available from various companies, including Google, Microsoft, and IBM. They provide a platform for building an AI assistant and often come with pre-built models. The second way to build an AI assistant is to use a pre-built framework. These are available from various companies, including Amazon Web Services and Google Cloud Platform. They provide a platform for building an AI assistant and often come with pre-built models. The third way to build an AI assistant is to use a custom solution. This means building an AI assistant from scratch. This can be done with various programming languages, including Python, R, and C++.
Basis Sharing	**What is the responsibility of the AI assistant?** An AI assistant is a kind of program that helps with different activities that a person might need during the day. It can do things like setting timers, making quick notes, or answering questions that are not too hard. The main idea is to make life easier by using AI for small actions that take time if done by hand. People can make AI assistants by using some services online. These platforms already have the tools, so it's easier to start. Companies like Amazon and others give these kinds of services. Another choice is to learn programming and try building one with coding. This can be more flexible but also more difficult. It depends on what the assistant should do and how much effort someone wants to put into making it work.
ASVD	**What is the responsibility of the AI assistant?** An AI assistant is designed to support users by automating a range of administrative and repetitive tasks. These include managing calendars, sending reminders, answering basic queries, and organizing information. The goal of an AI assistant is to enhance productivity by reducing the time spent on routine actions. There are several ways to create an AI assistant. One approach is using pre-built solutions provided by companies like Google or Microsoft. These tools often include integrated natural language processing and voice recognition capabilities. Another option is to use cloud-based AI services or SDKs that allow customization, such as Amazon Lex or Google Dialogflow. Developers who need more control can also build assistants from scratch using frameworks and machine learning libraries in Python or other languages.
NSNet	**What is the responsibility of the AI assistant?** An AI assistant is built to help users by handling routine and administrative tasks automatically. Typical functions include scheduling events, sending notifications, responding to simple inquiries, and organizing digital content. By offloading these repetitive duties, AI assistants aim to boost user efficiency and save time. There are multiple ways to develop such assistants. One common method involves leveraging pre-existing platforms from providers like Google or Microsoft, which offer built-in support for natural language understanding and voice interaction. Alternatively, cloud-based APIs and SDKs, such as Amazon Lex or Google Dialogflow, allow for greater customization. For those seeking maximum flexibility, AI assistants can also be developed from the ground up using programming languages like Python and popular machine learning libraries.