Journal of Data Science 0 (0), 1–21
??? 2024
DOI: 10.6339/24-JDS1156
Computing in Data Science

Magnitude Pruning of Large Pretrained Transformer Models with a Mixture Gaussian Prior

Mingxuan Zhang[1], Yan Sun[2], and Faming Liang[1,*]

[1]*Department of Statistics, Purdue University, West Lafayette, IN 47907, USA*
[2]*Department of Biostatistics, Epidemiology, and Informatics, University of Pennsylvania, Pennsylvania, PA 19104, USA*

Abstract

Large pretrained transformer models have revolutionized modern AI applications with their state-of-the-art performance in natural language processing (NLP). However, their substantial parameter count poses challenges for real-world deployment. To address this, researchers often reduce model size by pruning parameters based on their magnitude or sensitivity. Previous research has demonstrated the limitations of magnitude pruning, especially in the context of transfer learning for modern NLP tasks. In this paper, we introduce a new magnitude-based pruning algorithm called mixture Gaussian prior pruning (MGPP), which employs a mixture Gaussian prior for regularization. MGPP prunes non-expressive weights under the guidance of the mixture Gaussian prior, aiming to retain the model's expressive capability. Extensive evaluations across various NLP tasks, including natural language understanding, question answering, and natural language generation, demonstrate the superiority of MGPP over existing pruning methods, particularly in high sparsity settings. Additionally, we provide a theoretical justification for the consistency of the sparse transformer, shedding light on the effectiveness of the proposed pruning method.

Keywords *consistency; large language model; sparsity; stochastic transformer; transformer*

1 Introduction

Large pretrained transformer models have emerged as powerful tools for a variety of downstream natural language processing tasks, from natural language generation to question answering (Radford et al., 2019; Brown et al., 2020). These pretrained models have grown exponentially in size, often comprising hundreds of millions, or even billions, of parameters (Devlin et al., 2019; He et al., 2021; Lewis et al., 2019; Touvron et al., 2023). While their capabilities are undeniably impressive, the computational and storage requirements for such large models are becoming increasingly prohibitive (Strubell et al., 2020).

Score-based pruning, a technique that involves removal of non-expressive parameters based on their importance score rankings, presents a promising avenue for model compression. It has the potential to significantly reduce model size with minimal impact on performance.

Based on the definition of pruning scores, the pruning methods can be classified into distinct categories, such as magnitude-based (zeroth-order) pruning methods (Han et al., 2015a,b; Zhu and Gupta, 2017; Louizos et al., 2017; Wang et al., 2020) and sensitivity-based (higher-order) pruning methods (Molchanov et al., 2019; Ding et al., 2019; Sanh et al., 2020; Liang et al., 2021;

*Corresponding author. Email: fmliang@purdue.edu.

Received July 11, 2024; Accepted October 6, 2024

Zhang et al., 2022; Kurtic et al., 2022; Li et al., 2023). On the other hand, if the classification is made based on the strategies employed, the methods fall into categories such as one-shot pruning (Lee et al., 2018; Frankle and Carbin, 2018; Chen et al., 2020; Liang et al., 2021; Zafrir et al., 2021) and iterative pruning (Han et al., 2015a; Zhu and Gupta, 2017; Louizos et al., 2017; Sanh et al., 2020; Zhang et al., 2022; Li et al., 2023).

It has long been argued and experimentally demonstrated that magnitude-based pruning methods struggle to retain expressive parameters, particularly in high-sparsity settings. Furthermore, when it comes to transfer learning with large pretrained models, which are now the benchmark for state-of-the-art downstream NLP tasks, their effectiveness is reduced. As a result, models pruned using magnitude-based methods often exhibit diminished generalization performance (Sanh et al., 2020).

Recently, Bayesian sparse deep learning has made significant progress through a series of works (Liang et al., 2018b; Sun et al., 2022b,a, 2021; Zhang et al., 2023), demonstrating its potential in deep learning for both statistical inference and model sparsification. By adopting the mixture Gaussian prior (MGP) for the parameters of the neural network, they developed a magnitude-based one-shot pruning method and achieved state-of-the-art performance in pruning both convolutional neural networks (Sun et al., 2022a, 2021) and recurrent neural networks (Zhang et al., 2023). These early experimental results on small-scale models have once again sparked hope for magnitude-based pruning methods. However, their methods have not yet been evaluated on large transformer models across different tasks and datasets. Moreover, as we will discuss in Section 2.3, several key challenges prevent us from directly adopting their methods for pruning larger models.

In this work, we introduce MGPP, a magnitude-based iterative pruning algorithm that is both simple and effective. To validate its performance, we conducted extensive experiments across three key downstream tasks: natural language understanding, question answering, and natural language generation. Our evaluations span three types of pretrained transformer-based language models, $\text{DeBERTaV3}_{\text{base}}$ (He et al., 2021), $\text{BERT}_{\text{base}}$ (Devlin et al., 2019), and $\text{BART}_{\text{large}}$ (Lewis et al., 2019). Our results indicate that when guided by an appropriate prior, magnitude-based methods can outperform existing state-of-the-art pruning methods. Additionally, we provide a loose justification for the consistency of the sparse transformer, shedding light on its effectiveness.

The remaining part of the paper is organized as follows. Section 2 provides preliminary descriptions for related concepts and methods in the literature. Section 3 describes the proposed method and justifies its validity. Section 4 reports numerical experiments. Section 5 concludes the paper with a brief discussion.

2 Preliminaries

2.1 Pruning Scores

An essential component of effective pruning is accurately identifying non-expressive parameters through their importance score rankings. Consider a model defined by a set of parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^T \in \mathbb{R}^d$, each associated with an importance score. Let $\boldsymbol{S} = (S_1, \ldots, S_d)^T \in \mathbb{R}^d$ denote the corresponding score vector. Score-based pruning methods eliminate parameters based on these scores, with parameters assigned lower scores being prioritized for removal. As outlined in the Introduction, score-based pruning methods fall into two primary categories, namely, magnitude-based methods and sensitivity-based methods.

Magnitude-Based (Zeroth-Order) Methods (Zhu and Gupta, 2017; Wang et al., 2020; Chen et al., 2020; Zafrir et al., 2021), which determine the parameters to prune based on their magnitudes. For a given parameter θ_j, the score is defined as $S_j = |\theta_j|$. Among these methods, gradual magnitude pruning (GMP), introduced by Zhu and Gupta (2017), is particularly notable for its effectiveness and simplicity. This widely adopted pruning baseline has inspired the development of numerous subsequent methods, see e.g., Chen et al. (2020) and Zafrir et al. (2021).

Sensitivity-Based Methods (Sanh et al., 2020; Zhang et al., 2022; Kurtic et al., 2022; Li et al., 2023), which incorporate higher-order information, such as gradients and Hessian, to assess the impact of pruning on the loss function $\mathcal{L}$. The first-order pruning methods utilize gradient-based information. Notable examples include movement pruning (MvP) (Sanh et al., 2020) and PLATON (Zhang et al., 2022). The former removes model parameters that are moving towards zero, and the latter is designed to capture the uncertainty of model parameters' importance scores during the pruning process. In downstream pruning scenarios, particularly for BERT-like language models, PLATON is recognized as state-of-the-art, significantly outperforming other baselines, including MvP. The second-order pruning methods (LeCun et al., 1989; Singh and Alistarh, 2020; Frantar et al., 2021; Kurtic et al., 2022) utilize Hessian-based information. To circumvent the costly approximation of the inverse Hessian, Singh and Alistarh (2020) introduced the WoodFisher method, and Frantar et al. (2021) introduced the M-FAC method. However, Kurtic et al. (2022) showed that the WoodFisher method is computationally infeasible at the scale of BERT, and while the M-FAC method scales effectively, it yields inferior pruning results. In response, they proposed a general second-order pruning method, Optimal BERT Surgeon (oBERT), which achieves state-of-the-art performance in upstream pruning scenarios.

While the zeroth-order methods are simple, scalable, and often serve as standard baselines, they are consistently outperformed by higher-order methods, particularly in downstream pruning scenarios and at high sparsity levels. However, as discussed above, the performance gains from utilizing higher-order information come at the cost of additional memory and computational resources. For a model with d parameters, PLATON requires an extra $O(3d)$ memory to maintain three additional states: the average importance scores between consecutive pruning operations, and the exponential average of both the importance scores and the corresponding upper confidence bound. For the $\text{BERT}_{\text{BASE}}$ model, with $d = 85$ million parameters, managing these states is feasible. However, scaling to larger models, such as Llama-7b/70b (Touvron et al., 2023), necessitates approximately an additional 84GB/840GB of GPU memory.

The memory requirement for oBERT is $O(Bd)$, where B is a hyperparameter representing the width of the diagonal block-wise approximation of the empirical Fisher matrix. For the $\text{BERT}_{\text{BASE}}$ model, setting $B = 50$ results in an additional memory requirement of about 17GB. This demand is manageable for $\text{BERT}_{\text{BASE}}$ but becomes unscalable for larger models. The runtime complexity of oBERT is $O(mBd)$, where m denotes the number of gradient outer products used to approximate the Hessian; for the $\text{BERT}_{\text{BASE}}$ model, m is set to 1024.

2.2 Pruning Strategies

Pruning strategies can be classified into one-shot pruning and iterative pruning. In one-shot pruning, the sparsity pattern is predetermined using the scores of a fully-trained, dense model. A sparse model is then trained with pruned parameters fixed, a technique often termed "rewinding." However, choosing which parameters to prune based on a fully-trained model overlooks the

complex dynamics of training. As a result, parameters that are expressive may be unfairly eliminated at the early stage of training.

On the other hand, iterative pruning jointly performs training and pruning. The sparsity pattern is dynamically updated, offering the model an opportunity to recover from previous pruning decisions. Additionally, the sparsity level can be gradually increased during training through sparsity schedulers, such as the cubic sparsity scheduler (Zhu and Gupta, 2017; Sanh et al., 2020; Zafrir et al., 2021; Zhang et al., 2022; Kurtic et al., 2022; Li et al., 2023) given as follows:

$$v^{(t)} = \begin{cases} 0 & t < t_i, \\ v^{(T)} - v^{(T)} \left(1 - \frac{t - t_i}{t_f - t_i}\right)^3 & t_i \leqslant t \leqslant t_f, \\ v^{(T)} & t_f < t \leqslant T, \end{cases} \tag{1}$$

where $v^{(t)}$ is the sparsity level at the t-th training step, increasing from an initial value 0 to a final level $v^{(T)}$ over a period of $T - t_i - t_f$ steps following a warm-up of t_i steps. In practice, instead of performing pruning at every training step before reaching the target sparsity level, one can also choose to prune every Δt steps (Zhang et al., 2022; Li et al., 2023).

2.3 Mixture Gaussian Priors in Bayesian Sparse Deep Learning

The mixture Gaussian prior (MGP) has recently attracted significant attention in the field of Bayesian sparse deep learning (Sun et al., 2022a). Formally, it models each parameter of the network using a mixture Gaussian prior, defined as follows:

$$\theta_j \sim \lambda \cdot \mathcal{N}(0, \sigma_1^2) + (1 - \lambda) \cdot \mathcal{N}(0, \sigma_0^2), \tag{2}$$

where $\lambda \in (0, 1)$ is the mixture proportion, σ_0^2 is typically set to a very small value, whereas σ_1^2 is usually assigned a relatively larger value. In what follows, we denote the prior density function of θ_j as $\pi(\theta_j; \lambda, \sigma_0^2, \sigma_1^2)$. Furthermore, we assume that all model parameters are *a priori* independent, i.e., $\pi(\boldsymbol{\theta}; \lambda, \sigma_0^2, \sigma_1^2) = \prod_{j=1}^{d} \pi(\theta_j; \lambda, \sigma_0^2, \sigma_1^2)$.

This particular prior has been shown to offer several theoretical advantages under the Bayesian framework. These include posterior consistency, structure selection consistency, and asymptotic normality of predictions for both i.i.d. data (Sun et al., 2022a, 2021) and time-series data (Zhang et al., 2023). These properties make it useful in various applications like variable/model selection, uncertainty quantification, and model sparsification.

Next, we will discuss how this prior is used to prune models. Essentially, the prior serves as a form of regularization, imposing penalty on model parameters. During training, the learning objective becomes

$$\mathcal{L}(D_n, \boldsymbol{\theta}) - \frac{1}{n} \log(\pi(\boldsymbol{\theta}; \lambda, \sigma_0^2, \sigma_1^2)), \tag{3}$$

where n denotes the size of the training dataset D_n, and $\mathcal{L}(D_n, \boldsymbol{\theta})$ represents the negative log-likelihood function of the deep neural network. To facilitate the application of gradient-based optimization algorithms for minimizing (3), we provide the following numerically stable expression of the gradient of log-prior, despite its straightforward derivation:

$$\nabla_{\theta_j} \log(\pi(\theta_j; \lambda, \sigma_0^2, \sigma_1^2)) = -\left(\frac{\theta_j}{\sigma_0^2} g(\theta_j) + \frac{\theta_j}{\sigma_1^2} \left[1 - g(\theta_j)\right]\right), \tag{4}$$

where

$$g(\theta_j) = \left(\exp\{c_2 \theta_j^2 + c_1\} + 1\right)^{-1},$$

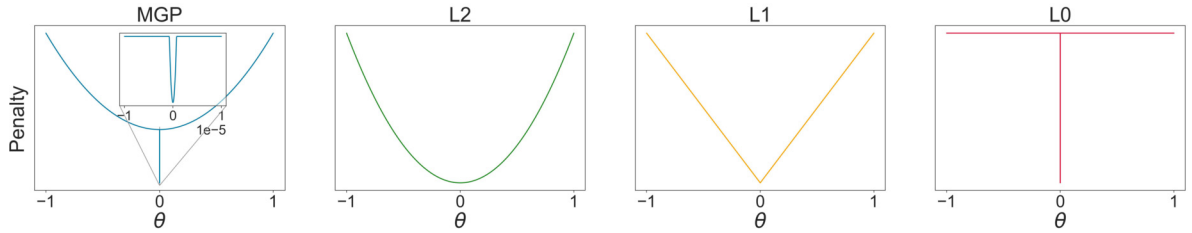

Figure 1: Visualization of the penalty functions across various regularization methods: the MGP displayed in the plot corresponds to $-\log(\pi(\theta; \lambda = 1 \times 10^{-6}, \sigma_0^2 = 1 \times 10^{-7}, \sigma_1^2 = 0.1))$, where a zoomed-in view for the region near zero is provided. Unlike L_0 regularization, which is not differentiable and requires the use of gradient estimators (Louizos et al., 2017), the MGP is differentiable across the entire parameter space.

with $c_1 = \ln(\lambda) - \ln(1-\lambda) + 0.5\ln(\sigma_0^2) - 0.5\ln(\sigma_1^2)$ and $c_2 = 0.5/\sigma_0^2 - 0.5/\sigma_1^2$.

The guidance from the MGP is conveyed through the gradients. The degree of penalty, which serves as the force pushing the model parameters toward zero, can be quantified by the absolute value of the gradient. A comparison between L_0, L_1, L_2, and MGP is presented in Figure 1.

The MGP acts as a piece-wise L_2 regularization, imposing different penalties across various regions of the parameter space. On a larger scale, the MGP applies penalties to parameters in a manner similar to L_2 regularization. In contrast, near zero in the small-scale region, the MGP imposes a more substantial penalty, setting it apart from L_2 regularization. In Section S1 of the Supplement, we illustrate and visualize how λ, σ_0^2, and σ_1^2 affect the landscape of the MGP.

According to the definitions provided in Sections 2.1 and 2.2, previous methods employing MGP (Sun et al., 2022a, 2021, 2022b; Zhang et al., 2023) can be categorized as magnitude-based, one-shot pruning methods. These methods train the model using the learning objective specified in Equation (3). Upon convergence, they perform one-shot pruning based on a pruning threshold determined by the values of λ, σ_0^2, and σ_1^2. Then, the pruned model is retrained using only the loss function $\mathcal{L}$ with the pruned parameters fixed to 0. The detailed algorithm is provided in Section S2 of the Supplement.

Their algorithms have set new standards in performance for pruning smaller models like ResNet-20, ResNet-32, and LSTMs. This highlights the effectiveness of MGP. However, translating these successes to larger, transformer-based models introduces challenges due to the many sensitive hyperparameters involved. Additionally, the retraining stage requires further tuning of hyperparameters such as learning rate and batch size, adding another layer of complexity. This is a particular concern given the computational resources required to train larger models.

Besides the aforementioned challenges, achieving a target sparsity level $v^{(T)}$ adds further complexity. In their approach, σ_1^2 and λ are held constant, while σ_0^2 is initialized to a large value, denoted as $(\sigma_0^{\text{init}})^2$. This initial setting is designed to closely align the proportion of pretrained model parameters that fall below the initial pruning threshold with $v^{(T)}$. A linear scheduler then gradually reduces σ_0^2 from $(\sigma_0^{\text{init}})^2$ to $(\sigma_0^{\text{end}})^2$. Throughout this prior-annealing (PA) process, the MGP penalty on these parameters increases, effectively driving them toward zero, so that they can be one-shot pruned in the end. However, as explained in Section 2.2, this pruning strategy may hurt the performance of the sparsified model. We provide experimental evidence in Section 4.6 to support our arguments.

3 The MGPP Method

To overcome these challenges, we introduce the MGPP method, summarized in Algorithm 1. Instead of relying on annealing the MGP to shrink the parameter down to zero, which tends to fix the sparsity pattern too early in the training process, we take a different approach. We keep the MGP fixed during training and utilize the cubic sparsity scheduler to gradually prune model parameters. When a set of parameters is pruned (i.e., set to zero), they receive gradients only from the loss function in the subsequent training iteration, as the gradients from the MGP become zero. This can serve as a remedy for false selection, thereby overcoming premature pruning of critical parameters. Parameters with large gradients from the loss are more likely to escape the region with large penalties, giving them a chance to be reconsidered for pruning later. Conversely, parameters that receive smaller gradients from the loss function will likely remain within the penalized region, making them candidates for future pruning.

Our proposed algorithm introduces only three additional hyperparameters beyond the standard ones: λ, σ_0^2, and σ_1^2. A comprehensive hyperparameter-sensitivity analysis is given in Section 4.8. Briefly, we find λ to be robust and set it universally to 10^{-7}. Preliminary experiments suggest that, when combined with a sparsity scheduler, it is preferable to set σ_0^2 to very small values such as 1×10^{-10}, in contrast to previous works that often set σ_0^2 (specifically, $(\sigma_0^{\text{end}})^2$) to larger values, i.e., $[1 \times 10^{-7}, 1 \times 10^{-5}]$. We limit σ_0^2 and σ_1^2 to the sets $\{1 \times 10^{-9}, 1 \times 10^{-10}\}$ and $\{0.05, 0.1\}$, respectively. Our approach significantly reduces the computational burden associated with hyperparameter tuning, especially when it comes to training large transformer models. Despite these restrictions, our method outperforms other baselines, as demonstrated in the experimental results section (see Section 4).

We also found that gradually incorporating the MGP improves the performance of the sparsified model. This 'prior warm-up' can be seamlessly integrated into the warm-up phase of the sparsity scheduler, denoted by Equation 1. Let $\eta^{(t)}$ be the prior coefficient at training step t, we have

$$v^{(t)}, \eta^{(t)} = \begin{cases} 0, \dfrac{t}{t_i} & t < t_i \\ v^{(T)} - v^{(T)} \left(1 - \frac{t - t_i}{t_f - t_i}\right)^3, 1 & t_i \leqslant t \leqslant t_f \\ v^{(T)}, 1 & t_f < t \leqslant T. \end{cases} \tag{5}$$

In the Appendix A, we provide a loose justification for the parameter estimation consistency of the sparse transformer model with the mixture Gaussian prior, drawing upon the established theory from Liang et al. (2022) and Liang et al. (2018a). This justifies the use of the mixture Gaussian prior for sparsifying the transformer model as proposed in the paper, while the MGPP method proposed above is mainly for locating a maximum *a posteriori* (MAP) solution for the complex transformer model. Additionally, we note that the parameter estimation consistency for the sparse transformer model is subject to loss-invariant transformations. That is, the model is assumed to be unique up to loss-invariant transformations, e.g., reordering the hidden neurons of the same hidden layer or simultaneously changing the signs or scales of certain connection weights and biases. The same assumption has often been used in studying theoretical properties of deep neural network models, see e.g., Liang et al. (2018b) and Sun et al. (2022a).

Algorithm 1 MGPP.

Input: training dataset D_n, pretrained model $\boldsymbol{\theta}^{(0)}$, number of training epochs E, mini-batch size m, λ, σ_0^2, σ_1^2, t_i, t_f, Δt
Initialize: $t = 1$, $T = \lceil En/m \rceil$, optimizer (e.g., AdamW (Loshchilov and Hutter, 2019))
for epoch from 1 to E **do**
  for each mini-batch $\mathcal{B}$ sampled from D_n **do**
    Calculate gradients of the loss through backpropagation

$$\nabla_{\boldsymbol{\theta}^{(t-1)}} \mathcal{L}(\mathcal{B}, \boldsymbol{\theta}^{(t-1)})$$

    Calculate $v^{(t)}$ and $\eta^{(t)}$ based on Eq. 5
    Calculate gradients of MGP based on Eq. 4

$$-\eta^{(t)} \frac{1}{n} \nabla_{\boldsymbol{\theta}^{(t-1)}} \log(\pi(\boldsymbol{\theta}^{(t-1)}; \lambda, \sigma_0^2, \sigma_1^2))$$

    Update $\boldsymbol{\theta}^{(t-1)} \rightarrow \boldsymbol{\theta}^{(t)}$ by an optimization step
    Calculate scores $\boldsymbol{S}^{(t)} = (|\theta_1^{(t)}|, \ldots, |\theta_d^{(t)}|)$
    if t mod $\Delta t = 0$ or $t > t_f$ **then**

$$\theta_j^{(t)} = \begin{cases} \theta_j^{(t)} & \text{if } S_j^{(t)} \text{ in top } v^{(t)}\% \\ 0 & \text{otherwise} \end{cases}$$

    end if
    Set $t = t + 1$
  end for
end for

4 Experiments

4.1 Experimental Setup

The performance of the final pruned models can be influenced by various factors unrelated to the pruning methodology, including the number of training epochs, the maximum input sequence length, maximum gradient norm, and the number of beams used for evaluation in natural language generation tasks, among others. To control for these variables and ensure a fair comparison with different baselines, we follow the guidelines established in the recent works (Zhang et al., 2022; Kurtic et al., 2022; Li et al., 2023). We set all these methodology-unrelated factors to match those used in (Zhang et al., 2022; Kurtic et al., 2022; Li et al., 2023). We only tune methodology-related factors, such as λ, σ_1^2, and σ_0^2, along with standard hyperparameters like learning rate and batch size, which are also tuned in the baseline methods. Additional details are provided below and in the Section S3 of the Supplement.

We evaluate the proposed method, MGPP, across three downstream NLP tasks: natural language understanding, question answering, and natural language generation, as well as in the upstream pruning scenario. Specifically, we apply MGPP to three pretrained transformer-based language models: DeBERTaV3$_{\text{base}}$ (180 million parameters), BERT$_{\text{base}}$ (110 million parameters), and BART$_{\text{large}}$ (400 million parameters).

Following the prior works (Louizos et al., 2017; Sanh et al., 2020; Zhang et al., 2022; Kurtic

et al., 2022; Li et al., 2023), we prune all weight matrices, except for embeddings, LayerNorm, and the final prediction module. Our implementation is based on the publicly available Hugging Face Transformers library (Wolf et al., 2020). All performance metrics reported for MGPP are derived from the mean of five independent runs, each using a different random seed.

We compare MGPP with the following baselines:

- `Gradual Magnitude Pruning (GMP)` (Zhu and Gupta, 2017) is a simple yet strong magnitude-based iterative pruning baseline, widely recognized as one of the best magnitude-based pruning methods.
- `Movement Pruning (MvP)` (Sanh et al., 2020) is a sensitivity-based (first-order) iterative pruning method that prunes parameters based on their movement away from zero.
- `Iterative pruning (ITP)` (Molchanov et al., 2019) is a sensitivity-based (first-order) iterative pruning method that prunes parameters at each iteration if their importance scores fall below a hard threshold.
- `PLATON` (Zhang et al., 2022) is a sensitivity-based (first-order) iterative pruning method designed to capture the uncertainty of model parameters' importance scores during the pruning process.
- `oBERT` (Kurtic et al., 2022) is a sensitivity-based (second-order) iterative pruning method that utilizes a diagonal block-wise approximation of the empirical Fisher matrix.
- `LoSparse` (Li et al., 2023) is a sensitivity-based (first-order) iterative pruning method for transformer-based language models that integrates low-rank and sparse matrices to prune weight matrices effectively.

4.2 Natural Language Understanding

We assess the pruning performance of MGPP on $\text{BERT}_{\text{base}}$ (Devlin et al., 2019) and $\text{DeBERTaV3}_{\text{base}}$ models (He et al., 2021) by conducting experiments on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which includes a variety of tasks. Specifically, GLUE features two single-sentence classification tasks, SST-2 (Socher et al., 2013) and CoLA (Warstadt et al., 2019), as well as three tasks focused on similarity and paraphrasing: MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), and QQP. Additionally, the benchmark includes four natural language inference tasks: MNLI (Williams et al., 2017), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005), and WNLI (Levesque et al., 2012). In accordance with prior studies, we omit WNLI from our experiments. Additional details regarding the datasets can be found in the Section S3.1 of the Supplement.

A table containing training details, such as learning rate, batch size, the number of training epochs, σ_0^2, and σ_1^2 for each dataset, is presented in the Section S3.1 of the Supplement.

The results on the GLUE development set are summarized in Tables 1 and 2; all baseline results are directly taken from Zhang et al. (2022); Li et al. (2023). MGPP consistently achieves equal or superior performance compared to existing approaches across most datasets and sparsity levels. Notably, as the amount of training data increases, our method performs even better relative to other baselines. For instance, as shown in Table 1, at a target sparsity level of 90%, MGPP achieves 85.2/84.2% accuracy on the MNLI dataset—3.5/2.4% higher than the best-performing baseline, LoSparse. Remarkably, our results at 90% sparsity for MNLI even surpass LoSparse's performance at 80% sparsity, demonstrating the effectiveness of our approach with more data and higher sparsity levels. Similarly, Table 2 shows that our method, while using less memory, achieves better or comparable results to PLATON.

Table 1: Comparison of different pruning methods for the DeBERTaV3$_{\text{base}}$ model on the GLUE development sets, where "N.A." indicates non-convergence of the model, "m/mm" denotes the accuracy for the matched and mismatched development sets of the MNLI task, and other metrics (i.e., Acc, F1, Mcc, P/S Corr) are defined in Table S1 (in the Supplement). The highest-performing results for each dataset are highlighted in bold.

Sparsity	**Method**	**MNLI** m/mm	**RTE** Acc	**QNLI** Acc	**MRPC** Acc/F1	**QQP** Acc/F1	**SST-2** Acc	**CoLA** Mcc	**STS-B** P/S Corr
0%	DeBERTaV3$_{\text{base}}$	90.5/90.6	82.0	94.0	89.5/93.3	92.4/89.8	95.3	69.2	91.6/91.1
80%	MvP	N.A.	61.2	86.0	79.2/85.0	N.A.	89.4	N.A.	84.3/84.3
	ITP	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	84.5/83.8	68.0	88.6	85.0/89.4	90.6/87.2	91.7	50.0	88.8/88.5
	MGPP	**87.2/86.9**	**70.0**	**91.7**	**85.5**/89.4	**91.3/88.3**	**93.2**	**56.3**	**88.9/88.5**
85%	MvP	N.A.	59.0	N.A.	78.5/84.3	N.A.	89.0	N.A.	83.9/83.9
	ITP	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	83.3/82.9	66.9	87.6	83.6/88.0	90.3/87.0	90.4	46.8	87.7/87.3
	MGPP	**86.0/85.9**	**68.3**	**90.9**	**84.3/88.7**	**91.1/87.9**	**92.3**	**50.9**	**88.0/87.5**
90%	MvP	N.A.	N.A.	N.A.	77.0/83.4	N.A.	88.0	N.A.	N.A.
	ITP	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	81.7/81.8	66.0	86.1	82.3/**87.4**	89.5/86.0	89.2	40.0	87.2/**87.0**
	MGPP	**85.2/84.2**	**66.2**	**88.8**	**82.6**/87.1	**91.1/88.0**	**90.2**	**48.0**	87.2/86.7

Table 2: Comparison of different methods for the BERT$_{\text{base}}$ model on the GLUE development sets in downstream tasks, where "m/mm" denotes the accuracy for the matched and mismatched development sets of the MNLI task. Refer to Table S1 (in the Supplement) for the other metrics used in the table.

Sparsity	**Method**	**MNLI** m/mm	**QQP** Acc/F1	**QNLI** Acc	**SST-2** Acc
0%	BERT$_{\text{base}}$	84.6 / 83.4	91.5 / 88.5	91.3	92.7
80%	GMP	81.5 / 82.9	86.0 / 83.8	89.2	84.3
	MvP	81.6 / 82.1	90.6 / 87.5	88.3	89.0
	PLATON	83.1 / 83.4	90.7 / 87.5	90.1	91.5
	MGPP	83.1 / 83.4	**90.8** / **87.6**	**90.2**	**91.9**
90%	GMP	78.8 / 79.0	78.8 / 77.0	86.6	80.7
	MvP	80.7 / 81.1	90.2 / 86.7	86.6	87.4
	PLATON	82.0 / 82.2	90.2 / 86.8	88.9	90.5
	MGPP	**82.1** / 82.2	**90.4** / **87.1**	**89.2**	**90.8**

Table 3: Comparison of MGPP, ITP, and LoSparse for the DeBERTaV3$_{\text{base}}$ model on the SQuADv1.1 validation set, where the best results for each dataset are highlighted in bold.

Dataset	**SQuADv1.1** EM/F1					
Sparsity	**95%**	**90%**	**80%**	**70%**	**60%**	**50%**
DeBERTaV3$_{\text{base}}$	87.7/93.5					
- ITP	65.2/76.1	70.9/80.3	75.0/83.9	78.2/86.2	78.2/86.2	81.5/89.6
- LoSparse	69.3/79.1	72.9/82.8	76.8/85.8	80.2/88.0	82.1/89.4	82.3/90.3
- MGPP	**73.7/82.9**	**78.0/86.2**	**80.2/88.6**	**81.1/89.5**	82.1/**90.1**	**82.5**/90.3

4.3 Question Answering

We assess the performance of MGPP on the DeBERTaV3$_{\text{base}}$ model (He et al., 2021) by conducting experiments on a standard question answering dataset SQuADv1.1 (Rajpurkar et al., 2016). SQuADv1.1 is a reading comprehension benchmark consisting of questions derived from Wikipedia articles, with 88k training samples and 10k validation samples.

For all sparsity levels, the number of training epochs and batch sizes is set to 10 and 16, respectively. We set the learning rate to 5×10^{-5}, and for the MGP, we specify $\sigma_0^2 = 1 \times 10^{-10}$ and $\sigma_1^2 = 0.05$. More details are given in the Section S3.2 of the Supplement.

The results on the SQuADv1.1 validation set are summarized in Table 3 using two performance metrics: exact match (EM) and F1. All baseline results are taken directly from (Li et al., 2023). MGPP demonstrates performance that is either superior to or on par with existing methods across all sparsity levels.

Consistent with our findings on the GLUE benchmark, our method is especially effective in high sparsity regimes. For example, at the 90% sparsity level, MGPP outperforms LoSparse (the best-performing baseline) by 5.1% in terms of EM.

4.4 Natural Language Generation

We assess the pruning performance of MGPP on the BART$_{\text{large}}$ model (Lewis et al., 2019) by conducting experiments on two natural language generation datasets: XSum (Narayan et al., 2018) and CNN/DailyMail (Hermann et al., 2015). The objective is to generate either a concise summary or a highlight that captures the main point of a document. Refer to the Section S3.3 of the Supplement for more detailed information about the datasets.

For all sparsity levels and both datasets, we set the number of training epochs to 12 and the batch size to 32. The beam search length is fixed at 8, and the learning rate is set to 2×10^{-5}. For the MGP, we set $\sigma_0^2 = 1 \times 10^{-10}$ and $\sigma_1^2 = 0.1$. Additional details can be founded in the Section S3.3 of the Supplement.

The results on the test sets of both datasets are summarized in Table 4, using three performance metrics: ROUGE 1/2/Lsum scores (Lin, 2004). All baseline results are directly adopted from (Li et al., 2023). The comparison indicates that MGPP outperforms existing approaches across all sparsity levels on both datasets. Notably, the larger the performance gap between the fully fine-tuned dense model and its sparsified counterpart, the greater the extent to which MGPP outperforms the baselines. This trend is especially pronounced for the XSum dataset, where the higher task complexity leads to a more significant gap.

Table 4: Comparison of MGPP, ITP, and LoSparse for the $\text{BART}_{\text{large}}$ model on the datasets: XSum and CNN/DailyMail, where "Lead-3" represents choosing the first 3 sentences as the summary, and the best results for each dataset are highlighted in bold.

Sparsity	Method	XSum	CNN/DailyMail
0%	Lead-3	16.30/1.60/11.95	40.42/17.62/36.67
	$\text{BART}_{\text{large}}$	45.14/22.27/37.25	44.16/21.28/40.90
50%	ITP	38.42/16.32/31.43	40.76/18.30/37.65
	LoSparse	39.18/16.91/31.62	41.54/19.04/38.58
	MGPP	**42.92/19.70/34.80**	**42.59/19.90/39.57**
60%	ITP	36.71/14.96/29.86	40.52/18.10/37.31
	LoSparse	38.30/16.02/30.72	41.42/19.00/38.47
	MGPP	**41.69/18.75/33.69**	**42.27/19.63/39.26**
70%	ITP	34.42/13.15/27.99	40.35/17.98/37.15
	LoSparse	37.41/15.42/30.02	41.21/18.84/38.21
	MGPP	**40.20/17.33/32.34**	**41.93/19.21/38.88**

4.5 Upstream Pruning

Upstream pruning (Zafrir et al., 2021) provides an alternative to the conventional downstream pruning approach. In upstream pruning, the model is pruned during the semi-supervised pre-training phase and then fine-tuned sparsely on specific downstream tasks. Models pruned in this manner generally exhibit improved generalization capabilities (Chen et al., 2020; Zafrir et al., 2021) and require significantly fewer computational resources for fine-tuning, as only the remaining parameters need to be adjusted. However, upstream pruning typically demands a considerably larger dataset compared to downstream pruning. Currently, oBERT (Kurtic et al., 2022) stands as the state-of-the-art method for upstream pruning on BERT-like models and serves as the primary baseline for comparison in this section.

Following the guidelines established by oBERT, we use the $\text{BERT}_{\text{base}}$ model fine-tuned on two upstream datasets: BookCorpus and English Wikipedia. We then apply MGPP to prune the model on the same datasets for 3 epochs. Finally, we sparse-fine-tune the pruned model on the GLUE benchmark for 8 epochs. Detailed hyperparameters are provided in Section S3.4 of the Supplement.

The results are presented in Table 5. We adopt all baseline results directly from Kurtic et al. (2022). Notably, MGPP outperforms oBERT, despite the latter's additional memory requirement of $O(50d)$ and its greater computational complexity of $O(mBd)$.

Table 5: Comparison of MGPP and oBERT on development sets for the upstream-pruned model $\text{BERT}_{\text{BASE}}$ at the 90% sparsity level, where "m/mm" indicates the accuracy for the matched and mismatched development sets of the MNLI task.

Sparsity	Method	MNLI m/mm	QNLI Acc	QQP Acc/F1	SST-2 Acc
0%	$\text{BERT}_{\text{BASE}}$	84.6 / 83.4	91.3	91.5 / 88.5	92.7
90%	oBERT	82.2 / 82.5	89.3	90.4 / 87.1	92.0
	MGPP	**82.4** / **82.6**	**89.8**	**90.5** / **87.3**	**92.3**

Table 6: Comparison of MGPP with two ablation variants, PA and L_2, on the MNLI, MRPC, and SST-2 datasets, where the results of MGPP are taken from Table 1.

Sparsity	**Method**	**MNLI** m/mm	**MRPC** Acc/F1	**SST-2** Acc
80%	PA	83.8/82.9	83.1/88.3	90.1
	L_2	86.0/85.6	82.4/87.3	91.5
	MGPP	**87.2/86.9**	**85.5/89.4**	**93.2**
85%	PA	81.6/81.4	78.4/85.7	88.5
	L_2	83.8/84.6	76.5/82.0	90.5
	MGPP	**86.0/85.9**	**84.3/88.7**	**92.3**
90%	PA	79.5/78.9	77.6/83.8	87.2
	L_2	81.6/81.2	71.8/82.1	87.1
	MGPP	**85.2/84.2**	**82.6/87.1**	**90.2**

4.6 Ablation Study

To justify the contributions of various components and design choices in our method, we conduct an ablation study in this section. We compare our approach to Prior-Annealing (PA) (Sun et al., 2021; Zhang et al., 2023), which provides an effective implementation for sparsifying deep neural network models with the MGP prior in the one-shot pruning style.

Additionally, we replace the MGP in our method with an L_2 penalty (denoted as L_2) to confirm the significance of this particular prior. As we have previously discussed, the MGP imposes penalties on parameters in a way that is similar to L_2 regularization on a larger scale of the parameter space.

The ablation study is carried out on the DeBERTaV3$_{\text{base}}$ model on three datasets from the GLUE benchmark: MNLI, MRPC, and SST-2. These datasets represent diverse task categories, including single-sentence classification, similarity and paraphrasing, and natural language inference. They also vary in training set size, ranked from large to small: MNLI, SST-2, MRPC.

The results are summarized in Table 6. Notably, we performed extensive hyperparameter search for PA to ensure a fair comparison (details are given in Section 3.5 of the Supplement). Despite this effort, MGPP consistently outperforms PA on all three datasets and across all sparsity levels. When compared to L_2, the advantage of MGPP becomes increasingly pronounced as sparsity increases. For example, on the MNLI dataset, at 80% sparsity, MGPP surpasses L_2 by 1.2/1.3%. At 90% sparsity, this margin grows significantly, with MGPP outperforming L_2 by 3.6/3.0%.

4.7 Algorithm Analysis

To better illustrate the impact of MGP, Figure 2 depicts the distribution of remaining nonzero parameters and the evolution of pruning thresholds during training for a 90% sparsified DeBERTaV3$_{\text{base}}$ model on the MNLI dataset, comparing our method against the L_2 variant. We observe that both MGPP and L_2 tend to prune parameters that are close to zero. However, as shown in Figure 2(b), the spike component in the MGP more effectively drives parameters toward zero, resulting in a lower pruning threshold. In contrast, L_2 fails to similarly reduce the pruning threshold, leading to a performance gap in generalization.

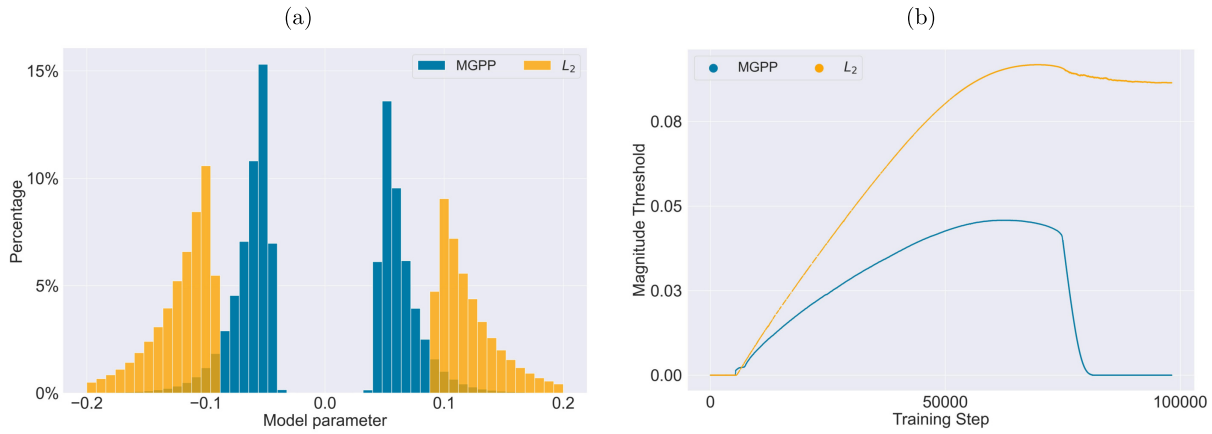

Figure 2: A comparative analysis of MGPP and L_2: (a) distributions of remaining nonzero parameters, (b) magnitude pruning thresholds during training.

4.8 Hyperparameter Sensitivity Analysis

The proposed method, MGPP, introduces six hyperparameters: three from the Mixture Gaussian Prior (MGP) and three from the cubic sparsity scheduler. It is important to note that the cubic sparsity scheduler is also employed by the baselines considered in this work, so no additional hyperparameters are introduced when comparing to these baselines.

In this section, we focus on the sensitivity of the three MGP-specific hyperparameters:

- λ: MGPP is robust to this hyperparameter, which we fix at 1×10^{-7} in all experiments. Changing λ only slightly adjusts the width of the spike component (see Section S1 of the Supplement). Preliminary experiments show that its impact on performance is negligible, and a value below 0.1 is generally sufficient.
- σ_0^2: A general guideline is to use a smaller value when more training samples are available. We limited our selection to the set $\{1 \times 10^{-9}, 1 \times 10^{-10}\}$. Values in the range $1 \times 10^{-12} \leqslant \sigma_0^2 \leqslant 1 \times 10^{-8}$ do not significantly affect performance.
- σ_1^2: Similar to σ_0^2, smaller values are recommended for larger datasets. We restricted our choices to the set $\{0.1, 0.05\}$. Although this hyperparameter has more impact on performance, the suggested values work well across all experiments.

For more detailed discussion on how these hyperparameters shape the prior landscape, please refer to Section S1 of the Supplement.

5 Conclusion

In this paper, we have developed MGPP, a novel magnitude-based iterative pruning method designed to sparsify large-scale transformer models. Extensive experimental results on various natural language processing tasks and two transformer-based language models demonstrate the effectiveness and efficiency of MGPP, particularly in settings with abundant training data or high sparsity. Additionally, we provided a theoretical justification for the consistency of MGPP, offering insights into its strong performance.

Transformer model pruning is an ongoing research area. Besides the pruning scores discussed in Section 2.1, more complex pruning scores have also been proposed in the literature. For instance, the Platon method (Zhang et al., 2022) prunes the model based on the upper confidence bound of the weight importance, while the WoodFisher (Singh and Alistarh, 2020), M-FAC (Frantar et al., 2021), and oBERT (Kurtic et al., 2022) methods utilize Hessian-based information for pruning. These pruning scores can also be computed with the mixture Gaussian prior, leading to new variants of the proposed method. Notably, the consistency property of the MGPP method can be extended to these new variants, providing a theoretical guarantee for their validity. In contrast, existing methods often lack such theoretical support for their performance. Additionally, we note that the calculation of complex pruning scores often requires higher GPU memory than that needed for magnitude-based pruning scores.

While model compression often involves other strategies like knowledge distillation and quantization, these are not mutually exclusive with pruning. For instance, one could enhance the performance of a pruned model through knowledge distillation and further reduce storage requirements by quantizing the remaining parameters. We leave such extensions for future work.

Supplementary Material

The supplementary material includes (i) a brief description for the prior annealing algorithm, (ii) detailed experimental settings, and (iii) a folder (code) which contains all the code for the proposed algorithm `MGPP` as well as the code to reproduce the experiments.

Appendix

A Consistency of Sparse Transformer with the MGP Penalty

The appendix is organized as follows. Section A provides a loose justification for the consistency of the proposed MGPP method. In Section A.1, we introduce an auxiliary stochastic transformer model; and in Section A.2, we provide a constructive proof for the consistency of the sparse stochastic transformer estimator under the theoretical framework of the imputation regularized-optimization (IRO) algorithm (Liang et al., 2018a). With this preparation, we then establish the consistency of the sparse transformer model with a mixture Gaussian prior/penalty.

A.1 Asymptotic Equivalence Between the Transformer and Stochastic Transformer Models

The asymptotic equivalence between the transformer and stochastic transformer has been studied in Kim et al. (2024). To make the paper self-contained, we provided a brief review as follows.

Transformer Model Following Thickstun (2020), we define a transformer block as follows. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T \in \mathbb{R}^{n\times d}$ denote an input matrix to a transformer block, which trans-

forms $\boldsymbol{x}$ to $\boldsymbol{z} \in \mathbb{R}^{n\times d}$ with the detail specified as follows:

$$\begin{aligned}
&Q^{(h)}(\boldsymbol{x}_i) = W_{h,q}^T \boldsymbol{x}_i, \quad K^{(h)}(\boldsymbol{x}_i) = W_{h,k}^T \boldsymbol{x}_i, \\
&V^{(h)}(\boldsymbol{x}_i) = W_{h,v}^T \boldsymbol{x}_i, \quad W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d\times k}, \\
&\alpha_{i,j}^{(h)} = \mathrm{softmax}_j\left(\frac{\langle Q^{(h)}(\boldsymbol{x}_i), K^{(h)}(\boldsymbol{x}_j)\rangle}{\sqrt{k}}\right), \ i, j = 1, 2, \ldots, n, \\
&\boldsymbol{u}_i = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(\boldsymbol{x}_j), \quad W_{c,h} \in \mathbb{R}^{k\times d}, \\
&\tilde{\boldsymbol{u}}_i = \mathrm{LayerNorm}(\boldsymbol{x}_i + \boldsymbol{u}_i; \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1), \quad \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1 \in \mathbb{R}^d, \\
&\tilde{\boldsymbol{z}}_i = W_2^T \mathrm{ReLU}(W_1^T \tilde{\boldsymbol{u}}_i), \quad W_1 \in \mathbb{R}^{d\times m}, \ W_2 \in \mathbb{R}^{m\times d}, \\
&\boldsymbol{z}_i = \mathrm{LayerNorm}(\tilde{\boldsymbol{u}}_i + \tilde{\boldsymbol{z}}_i; \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2), \quad \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2 \in \mathbb{R}^d,
\end{aligned} \tag{6}$$

where H denotes the number of attention heads, $\langle\cdot,\cdot\rangle$ denotes inner product, and the layerNorm is given by

$$\mathrm{LayerNorm}(\boldsymbol{a}; \boldsymbol{\gamma}, \boldsymbol{\beta}) = \boldsymbol{\gamma} \odot \frac{(\boldsymbol{a} - \bar{a}\mathbf{1}_d)}{s_a} + \boldsymbol{\beta},$$

where $\odot$ is the element-wise multiplication operator, $\boldsymbol{a} = (a_1, a_2, \ldots, a_d)^T \in \mathbb{R}^d$, $\mathbf{1}_d$ is an d-vector of ones, $\bar{a} = \frac{1}{d}\sum_{i=1}^d a_i$, and $s_a = \sqrt{\frac{1}{d}\sum_{i=1}^d (a_i - \bar{a})^2}$. For convenience, we denote the transformer block by the function $f_{\boldsymbol{\theta}} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, where the parameters $\boldsymbol{\theta}$ consist of $\{W_{h,q}, W_{h,k}, W_{h,v}, W_{c,h} : h = 1, 2, \ldots, H\}$ and $\{\boldsymbol{\gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2, W_1, W_2\}$. A transformer is a composition of L transformer blocks: $f_{\boldsymbol{\theta}_L} \circ \cdots \circ f_{\boldsymbol{\theta}_1}(\boldsymbol{x}) \in \mathbb{R}^{n\times d}$, each block has its own parameters. The common settings of the hyperparameters are $d = 512$, $k = 64$, $m = 2048$, and $H = 8$.

Stochastic Transformer Model The stochastic transformer model is defined as follows:

$$\begin{aligned}
&Q^{(h)}(\boldsymbol{x}_i) = W_{h,q}^T \boldsymbol{x}_i + \boldsymbol{\epsilon}^{h,q}, \quad K^{(h)}(\boldsymbol{x}_i) = W_{h,k}^T \boldsymbol{x}_i + \boldsymbol{\epsilon}^{h,k}, \\
&V^{(h)}(\boldsymbol{x}_i) = W_{h,v}^T \boldsymbol{x}_i + \boldsymbol{\epsilon}^{h,v}, \quad W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d\times k}, \\
&\alpha_{i,j}^{(h)} = \mathrm{softmax}_j\left(\frac{\langle Q^{(h)}(\boldsymbol{x}_i), K^{(h)}(\boldsymbol{x}_j)\rangle}{\sqrt{k}}\right), \ i, j = 1, 2, \ldots, n, \\
&\boldsymbol{u}_i = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(\boldsymbol{x}_j) + \boldsymbol{\epsilon}^{i,u}, \quad W_{c,h} \in \mathbb{R}^{k\times d}, \\
&\tilde{\boldsymbol{u}}_i = \mathrm{LayerNorm}(\boldsymbol{x}_i + \boldsymbol{u}_i; \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1), \quad \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1 \in \mathbb{R}^d, \\
&\tilde{\boldsymbol{z}}_i = W_2^T \mathrm{ReLU}(W_1^T \tilde{\boldsymbol{u}}_i) + \boldsymbol{\epsilon}^{i,\tilde{z}}, \ W_1 \in \mathbb{R}^{d\times m}, \ W_2 \in \mathbb{R}^{m\times d}, \\
&\boldsymbol{z}_i = \mathrm{LayerNorm}(\tilde{\boldsymbol{u}}_i + \tilde{\boldsymbol{z}}_i; \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2), \quad \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2 \in \mathbb{R}^d,
\end{aligned} \tag{7}$$

where the noise variables $\boldsymbol{\epsilon}^{h,q} \sim N(0, \sigma_q^2 I_k)$, $\boldsymbol{\epsilon}^{h,k} \sim N(0, \sigma_k^2 I_k)$, $\boldsymbol{\epsilon}^{h,v} \sim N(0, \sigma_v^2 I_k)$, $\boldsymbol{\epsilon}^{i,u} \sim N(0, \sigma_u^2 I_d)$, and $\epsilon^{i,\tilde{z}} \sim N(0, \sigma_{\tilde{z}}^2 I_d)$ are mutually independent. Note that σ_q^2, σ_k^2, σ_v^2, σ_u^2, and $\sigma_{\tilde{z}}^2$ are all known, pre-specified by user. As a consequence of introducing the noise variables, we can treat $Q^{(h)}$'s, $K^{(h)}$'s, $V^{(h)}$'s, $\boldsymbol{u}_i$'s, and $\tilde{\boldsymbol{z}}_i$'s as latent variables, and decompose the model as

$$\begin{aligned}
\pi_{\boldsymbol{\theta}}(z, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}}|\boldsymbol{x}) &= \prod_{h=1}^{H} \pi(Q^{(h)}|\boldsymbol{x}, \boldsymbol{\theta}^{(1)}) \prod_{h=1}^{H} \pi(K^{(h)}|\boldsymbol{x}, \boldsymbol{\theta}^{(2)}) \prod_{h=1}^{H} \pi(V^{(h)}|\boldsymbol{x}, \boldsymbol{\theta}^{(3)}) \\
&\times \pi(\boldsymbol{U}|\boldsymbol{x}, \boldsymbol{\theta}^{(4)}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \pi(\tilde{\boldsymbol{Z}}|\boldsymbol{x}, \boldsymbol{\theta}^{(5)}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}) \pi(z|\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}}, \boldsymbol{x}, \boldsymbol{\theta}^{(6)}),
\end{aligned} \tag{8}$$

where $\boldsymbol{Q} = \{Q^{(1)}, Q^{(2)}, \ldots, Q^{(H)}\}$, $\boldsymbol{K} = \{K^{(1)}, K^{(2)}, \ldots, K^{(H)}\}$, $\boldsymbol{V} = \{V^{(1)}, V^{(2)}, \ldots, V^{(H)}\}$, $\boldsymbol{U} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$, $\tilde{\boldsymbol{Z}} = \{\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, \ldots, \tilde{\boldsymbol{z}}_n\}$, $\boldsymbol{\theta}^{(1)} = \{W_{h,q} : h = 1, 2, \ldots, H\}$, $\boldsymbol{\theta}^{(2)} = \{W_{h,k} : h = 1, 2, \ldots, H\}$, $\boldsymbol{\theta}^{(3)} = \{W_{h,v} : h = 1, 2, \ldots, H\}$, $\boldsymbol{\theta}^{(4)} = \{W_{c,h} : h = 1, 2, \ldots, H\}$, $\boldsymbol{\theta}^{(5)} = \{W_1, W_2, \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1\}$, and $\boldsymbol{\theta}^{(6)} = \{\boldsymbol{\gamma}_2, \boldsymbol{\beta}_2\}$.

The asymptotic equivalence between the transformer and stochastic transformer models have been established in Kim et al. (2024), where it was shown that the two models have asymptotically the same loss function under appropriate conditions.

More precisely, they showed that there exists a small value $\tau(d, k, m, H)$, as a function of d, k, m and H, such that

$$\sup_{\boldsymbol{\theta} \in \Theta} \frac{1}{n} \left| \log \pi_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}} | \boldsymbol{x}) - \log \tilde{\pi}_{\boldsymbol{\theta}}(\boldsymbol{z} | \boldsymbol{x}) \right| \xrightarrow{p} 0, \tag{9}$$

as $n \to \infty$ and $\max\{\sigma_q, \sigma_k, \sigma_v, \sigma_u, \sigma_{\tilde{z}}\} \prec \tau(d, k, m, H)$, where $\pi_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}} | \boldsymbol{x})$ represents the pseudo-complete data likelihood function of the stochastic transformer by treating $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, $\boldsymbol{U}$, and $\tilde{\boldsymbol{Z}}$ as latent variables, $\tilde{\pi}_{\boldsymbol{\theta}}(\boldsymbol{z} | \boldsymbol{x})$ represents the likelihood function of the transformer, and $\xrightarrow{p}$ denotes convergence in probability. We note that similar techniques have been used in Liang et al. (2022) and Sun and Liang (2022) in establishing the asymptotic equivalence between the deep neural network and stochastic neural network (StoNet) models.

A.2 Consistency of Sparse Transformer

By treating $\{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}}\}$ as latent variables, the parameters $\boldsymbol{\theta}$ of the stochastic transformer model can be estimated using a regularization approach as follows:

$$\hat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta}} \Big\{ \log \pi_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}} | \boldsymbol{x}) + \log P_\lambda(\boldsymbol{\theta}) \Big\}, \tag{10}$$

where $P_\lambda(\boldsymbol{\theta})$ denotes the sparsity penalty imposed on $\boldsymbol{\theta}$, and λ is the tuning parameter. With an appropriate choice of $P_\lambda(\boldsymbol{\theta})$, we can provide a constructive proof for the consistency of $\hat{\boldsymbol{\theta}}_n$ based on the IRO algorithm (Liang et al., 2018a).

The IRO algorithm starts with an initial weight setting $\hat{\boldsymbol{\theta}}_n^{(0)}$ and then iterates between the imputation and regularized optimization steps:

- **Imputation:** For each block, conditioned on the current parameter estimate $\boldsymbol{\theta}_{t-1}$, simulate the latent variables $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{U}, \tilde{\boldsymbol{Z}})$ from the predictive distribution

$$\begin{aligned} \pi_{\boldsymbol{\theta}_{t-1}}(\boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t | \boldsymbol{x}, \boldsymbol{z}) \propto & \prod_{h=1}^{H} \pi(Q_t^{(h)} | \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(1)}) \prod_{h=1}^{H} \pi(K_t^{(h)} | \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(1)}) \prod_{h=1}^{H} \pi(V_t^{(h)} | \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(2)}) \\ & \times \pi(\boldsymbol{U}_t | \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(4)}, \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t) \pi(\tilde{\boldsymbol{Z}}_t | \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(5)}, \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t) \pi(\boldsymbol{z} | \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t, \boldsymbol{x}, \boldsymbol{\theta}_{t-1}^{(6)}), \end{aligned} \tag{11}$$

  where t indexes iterations, $\boldsymbol{Q}_t = \{Q_t^{(h)} : h = 1, 2, \ldots, H\}$, $\boldsymbol{K}_t = \{K_t^{(h)} : h = 1, 2, \ldots, H\}$, $\boldsymbol{V}_t = \{V_t^{(h)} : h = 1, 2, \ldots, H\}$, $\boldsymbol{U}_t = \{\boldsymbol{u}_{1,t}, \boldsymbol{u}_{2,t}, \ldots, \boldsymbol{u}_{n,t}\}$, $\tilde{\boldsymbol{Z}}_t = \{\tilde{\boldsymbol{z}}_{1,t}, \tilde{\boldsymbol{z}}_{2,t}, \ldots, \tilde{\boldsymbol{z}}_{n,t}\}$. Here, $\boldsymbol{u}_{i,t}$ and $\tilde{\boldsymbol{z}}_{i,t}$ denote, respectively, the imputed values for $\boldsymbol{u}_i$ and $\boldsymbol{z}_i$ at iteration t.
- **Regularized optimization:** Given the pseudo-complete data $\{\boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t, \boldsymbol{z}, \boldsymbol{x}\}$, update $\hat{\boldsymbol{\theta}}_n^{(t-1)}$ by maximizing the penalized log-likelihood function as follows:

$$\hat{\boldsymbol{\theta}}_n^{(t)} = \arg\max_{\boldsymbol{\theta}} \Big\{ \log \pi_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t | \boldsymbol{x}) + \log P_\lambda(\boldsymbol{\theta}) \Big\}, \tag{12}$$

which, by the decomposition (8), can be reduced to solving for $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(6)}$, separately. The penalty function $P_\lambda(\boldsymbol{\theta})$ should be chosen such that $\hat{\boldsymbol{\theta}}_n^{(t)}$ forms a consistent estimator for the working parameter

$$\begin{aligned}\boldsymbol{\theta}_*^{(t)} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\hat{\boldsymbol{\theta}}^{(t-1)}} \log \pi(\boldsymbol{Z}, \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t | \boldsymbol{\theta}, \boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{\theta}} \int \log \pi(\boldsymbol{z}, \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t | \boldsymbol{\theta}, \boldsymbol{x}) \\ &\quad \times \pi(\boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t, \boldsymbol{U}_t, \tilde{\boldsymbol{Z}}_t | \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}_n^{(t-1)}) \pi(\boldsymbol{z} | \boldsymbol{x}, \boldsymbol{\theta}^*) d\boldsymbol{Q}_t d\boldsymbol{K}_t d\boldsymbol{V}_t d\boldsymbol{z},\end{aligned}$$

where $\boldsymbol{\theta}^*$ is defined by

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E} \log \pi(\boldsymbol{z} | \boldsymbol{\theta}, \boldsymbol{x}), \tag{13}$$

and it corresponds to the true parameters of the underlying sparse transformer model (6).

For the sake of theoretical simplicity, we can assume that the hyperparameters of the transformer, namely, d, k, m and H, can increase with n but at a low order. By standard statistical estimation theory, see e.g., Portnoy (1988), we can achieve consistency of $\hat{\boldsymbol{\theta}}_n^{(t)}$ with the mixture Gaussian prior/penalty at each iteration of the IRO algorithm.

The above assumption can be much relaxed. For example, we may assume that d, k, m and H increase with n exponentially. Under this extended assumption, we can still achieve consistency of $\hat{\boldsymbol{\theta}}_n^{(t)}$ with the mixture Gaussian prior/penalty at each iteration of the IRO algorithm. This is possible by leveraging the theories presented in Song and Liang (2022), Sun et al. (2022a), and Sun et al. (2021). To elaborate, the estimation of $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(4)}$ is reduced to solving a series of high-dimensional linear regressions, for which consistency with the mixture Gaussian prior can be maintained, as per the theory from Song and Liang (2022). The estimation of $\boldsymbol{\theta}^{(5)}$ is reduced to solving a sparse deep neural network model with the ReLU and linear activation functions, ensuring consistency with the mixture Gaussian prior based on the theories outlined in Sun et al. (2022a) and Sun et al. (2021). The case of $\boldsymbol{\theta}^{(6)}$ is similar to $\boldsymbol{\theta}^{(5)}$, it is reduced to solving a sparse deep neural network model when a fully connected neural network is added to connect $\boldsymbol{z}$ and $\boldsymbol{y}$. Otherwise, the parameters $\{\boldsymbol{\gamma}_2, \boldsymbol{\beta}_2\}$ can be uniquely determined. Note that, as mentioned in the main text, the parameters in the LayerNorm transformation are not sparsified.

Furthermore, according to Theorem 4 of Liang et al. (2018a), the estimator $\hat{\boldsymbol{\theta}}_n^{(t)}$ is consistent when both n and t are sufficiently large. In summary, under mild regularity conditions and the mixture Gaussian prior, we can establish that

$$\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \xrightarrow{p} 0, \tag{14}$$

for sufficiently large n and sufficiently large t and almost every dataset $\{\boldsymbol{x}, \boldsymbol{y}\}$ for the stochastic transformer model.

Finally, based on (9) and under certain regularity conditions as given in Liang et al. (2022), we also have

$$\|\tilde{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \xrightarrow{p} 0, \tag{15}$$

where $\tilde{\boldsymbol{\theta}}_n^{(t)}$ is a sparse transformer estimator give by

$$\tilde{\boldsymbol{\theta}}_n^{(t)} = \arg\max_{\boldsymbol{\theta}} \Big\{ \log f_{\boldsymbol{\theta}}(\boldsymbol{z} | \boldsymbol{x}) + \log P_\lambda(\boldsymbol{\theta}) \Big\}. \tag{16}$$

In summary, through the introduction of an auxiliary stochastic transformer model and the utilization of the IRO convergence theory, we have justified the consistency of the sparse

transformer model under mild regularity conditions similar to those given in Liang et al. (2022) and Liang et al. (2018a).

Finally, we note that the above justification for the consistency of the sparse transformer model is based on the assumption that $\boldsymbol{x} \in \mathbb{R}^{n \times d}$ consists of n i.i.d observations. In practice, the observations might exhibit correlations. Nevertheless, this should not significantly impact the validity of our results, as long as $\boldsymbol{x}$ contains a sufficiently large number of independent samples.

Acknowledgement

The authors thank the editor, associate editor, and referees for their constructive comments which has led to significant improvement of this paper.

Funding

Liang's research is support in part by the NSF grants DMS-2015498 and DMS-2210819, and the NIH grant R01-GM152717.

References

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. (2020). Language models are few-shot learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M-F, Lin H-T (eds) *Advances in Neural Information Processing Systems 33*: 1877–1901.

Cer D, Diab M, Agirre E, Lopez-Gazpio I, Specia L (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint: https://arxiv.org/abs/1708.00055.

Chen T, Frankle J, Chang S, Liu S, Zhang Y, Wang Z, et al. (2020). The lottery ticket hypothesis for pre-trained bert networks. In: Larochelle H, Ranzato M, Hadsell R, Balcan M-F, Lin H-T (eds) *Advances in Neural Information Processing Systems 33*: 15834–15846.

Dagan I, Glickman O, Magnini B (2006). The Pascal recognising textual entailment challenge. In: Quiñonero-Candela J, Dagan I, Magnini B, d'Alché-Buc F (eds) *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, 177–190. Springer.

Devlin J, Chang MW, Lee K, Toutanova K (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Ding X, Zhou X, Guo Y, Han J, Liu J, et al. (2019). Global sparse momentum sgd for pruning very deep neural networks. In: Wallac HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox, EB, and Garnett, R (eds) *Advances in Neural Information Processing Systems, 32*.

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

Frankle J, Carbin M (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint: https://arxiv.org/abs/1803.03635.

Frantar E, Kurtic E, Alistarh D (2021). M-fac: Efficient matrix-free approximations of second-order information. In: Ranzato M, Beygelzimer A, Dauphin YN, Liang P, Vaughan JW (eds) *Advances in Neural Information Processing Systems 34*: 14873–14886.

Han S, Mao H, Dally WJ (2015a). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint: https://arxiv.org/abs/1510.00149.

Han S, Pool J, Tran J, Dally W (2015b). Learning both weights and connections for efficient neural network. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) *Advances in Neural Information Processing Systems, 28*: 1135–1143.

He P, Gao J, Chen W (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. arXiv preprint: https://arxiv.org/abs/2111.09543.

Hermann KM, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, et al. (2015). Teaching machines to read and comprehend. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) *Advances in Neural Information Processing Systems, 28*: 1693–1701.

Kim S, Sun Y, Liang F (2024). Narrow and deep neural networks achieve feature learning consistency.

Kurtic E, Campos D, Nguyen T, Frantar E, Kurtz M, Fineran B, et al. (2022). The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. arXiv preprint: https://arxiv.org/abs/2203.07259.

LeCun Y, Denker J, Solla S (1989). Optimal brain damage. In: Touretzky DS (eds) *Advances in Neural Information Processing Systems, 2*: 598–605.

Lee N, Ajanthan T, Torr PH (2018). Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint: https://arxiv.org/abs/1810.02340.

Levesque H, Davis E, Morgenstern L (2012). The winograd schema challenge. In: Brewka G, Eiter T, McIlraith SA (eds) *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, et al. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint: https://arxiv.org/abs/1910.13461.

Li Y, Yu Y, Zhang Q, Liang C, He P, Chen W, et al. (2023). Losparse: Structured compression of large language models based on low-rank and sparse approximation. arXiv preprint: https://arxiv.org/abs/2306.11222.

Liang C, Zuo S, Chen M, Jiang H, Liu X, He P, et al. (2021). Super tickets in pre-trained language models: From model compression to improving generalization. arXiv preprint: https://arxiv.org/abs/2105.12002.

Liang F, Jia B, Xue J, Li Q, Luo Y (2018a). An imputation-regularized optimization algorithm for high-dimensional missing data problems and beyond. *Journal of the Royal Statistical Society, Series B*, 80(5): 899–926. https://doi.org/10.1111/rssb.12279

Liang F, Li Q, Zhou L (2018b). Bayesian neural networks for selection of drug sensitive genes. *Journal of the American Statistical Association*, 113(523): 955–972. https://doi.org/10.1080/01621459.2017.1409122

Liang S, Sun Y, Liang F (2022). Nonlinear sufficient dimension reduction with a stochastic neural network. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A (eds) *Advances in Neural Information Processing Systems 35*.

Lin CY (2004). Rouge: A package for automatic evaluation of summaries. In: *Text Summarization Branches Out*, 74–81.

Loshchilov I, Hutter F (2019). Decoupled weight decay regularization.

Louizos C, Welling M, Kingma DP (2017). Learning sparse neural networks through l_0 regularization. arXiv preprint: https://arxiv.org/abs/1712.01312.

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

Narayan S, Cohen SB, Lapata M (2018). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. arXiv preprint: https://arxiv.org/abs/1808.08745.

Portnoy S (1988). Asymptotic behavior of likelihood methods for exponential families when the number of parameters tend to infinity. *The Annals of Statistics*, 16(1): 356–366. https://doi.org/10.1214/aos/1176350710

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

Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv preprint: https://arxiv.org/abs/1606.05250.

Sanh V, Wolf T, Rush A (2020). Movement pruning: Adaptive sparsity by fine-tuning. In: Larochelle H, Ranzato M, Hadsell R, Balcan M-F, Lin H-T (eds) *Advances in Neural Information Processing Systems 33*: 20378–20389.

Singh SP, Alistarh D (2020). Woodfisher: Efficient second-order approximation for neural network compression. In: Larochelle H, Ranzato M, Hadsell R, Balcan M-F, Lin H-T (eds) *Advances in Neural Information Processing Systems 33*: 18098–18109.

Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642.

Song Q, Liang F (2022). Nearly optimal Bayesian shrinkage for high-dimensional regression. *Science China Mathematics*, 66: 409–442. https://doi.org/10.1007/s11425-020-1912-6

Strubell E, Ganesh A, McCallum A (2020). Energy and policy considerations for deep learning in nlp. 2019, arXiv preprint: https://arxiv.org/abs/1906.02243.

Sun Y, Liang F (2022). A kernel-expanded stochastic neural network. *Journal of the Royal Statistical Society Series B*, 84(2): 547–578. https://doi.org/10.1111/rssb.12496

Sun Y, Song Q, Liang F (2022a). Consistent sparse deep learning: Theory and computation. *Journal of the American Statistical Association*, 117: 1981–1995. https://doi.org/10.1080/01621459.2021.1895175

Sun Y, Song Q, Liang F (2022b). Learning sparse deep neural networks with a spike-and-slab prior. *Statistics & Probability Letters*, 180: 109246. https://doi.org/10.1016/j.spl.2021.109246

Sun Y, Xiong W, Liang F (2021). Sparse deep learning: A new framework immune to local traps and miscalibration. In: Ranzato M, Beygelzimer A, Dauphin YN, Liang P, Vaughan JW (eds) *Advances in Neural Information Processing Systems 34*: 22301–22312.

Thickstun J (2020). The transformer model in equations. https://johnthickstun.com/docs/transformers.pdf.

Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, et al. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint: https://arxiv.org/abs/2307.09288.

Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint: https://arxiv.org/abs/1804.07461.

Wang H, Qin C, Zhang Y, Fu Y (2020). Neural pruning via growing regularization. arXiv preprint: https://arxiv.org/abs/2012.09243.

Warstadt A, Singh A, Bowman SR (2019). Neural network acceptability judgments. *Transac-*

tions of the Association for Computational Linguistics, 7: 625–641. https://doi.org/10.1162/tacl_a_00290

Williams A, Nangia N, Bowman SR (2017). A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint: https://arxiv.org/abs/1704.05426.

Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. (2020). Transformers: State-of-the-art natural language processing. In: Liu Q, Schlangen D (eds) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Association for Computational Linguistics, Online.

Zafrir O, Larey A, Boudoukh G, Shen H, Wasserblat M (2021). Prune once for all: Sparse pre-trained language models. arXiv preprint: https://arxiv.org/abs/2111.05754.

Zhang M, Sun Y, Liang F (2023). Sparse deep learning for time series: Theory and Applications. In: Oh A, Naumann T, Globerson A, Saenko K, Levine S (eds) *Advances in Neural Information Processing Systems 35*.

Zhang Q, Zuo S, Liang C, Bukharin A, He P, Chen W, et al. (2022). Kamalika Chaudhuri and Stefanie Jegelka and Le Song and Csaba Szepesvári and Gang Niu and Sivan Sabato, Platon: Pruning large transformer models with upper confidence bound of weight importance. In: Chaudhuri K, Jegelka S, Song L, Szepesvári C, Niu G, Sabato S (eds) *International Conference on Machine Learning*: 26809–26823. PMLR.

Zhu M, Gupta S (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint: https://arxiv.org/abs/1710.01878.