LEARNING TO SKETCH FOR LOW-RANK MODEL COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer blocks are the cornerstones of modern deep neural networks. While energy consumption used to be dominated by training of the models, widespread adoption and intensive usage shifted it towards model inference - thus highlighting the importance of post-training model compression for transformers. Sketching describes the field of probabilistic compression that use random projections, sampled from a probability distribution. While being widely used for data compression, sketching for model compression remains widely unexplored. One key component of the sketching methods is the choice of probability distribution over sketching matrices. We propose data-aware sketching by learning the sketching distribution through training a variational autoencoder (VAE). To preserve the utility of the low-ranked model on the downstream task, we train the VAE with a carefully tailored loss function. We show that our method outperforms data-oblivious sketching and low-rank compression baselines. *For compression ratios of up to* 30%*, our compressed model outperforms even the uncompressed baseline*. Additionally, we show that for both depth upscaling and domain adaptation, a fine-tuning step of the VAE decoder is sufficient, highlighting the applicability of the data-aware sketching.

1 INTRODUCTION

Large language models (LLMs) have become the core of the digital infrastructure that underpins several areas of our lives. On the one hand, with the increased usage of those models in our everyday lives, the inference costs have overtaken the training costs for LLMs - highlighting the importance of model compression techniques to reduce carbon emissions. On the other hand, the increasing size of those models widens the gap between big companies with large computational resources for both training and inference and smaller companies and universities with restricted access to computing. Particularly for private and sensitive data, running small-scale versions of those larger models becomes more and more important. While a lot of focus within the research community has been put on post-training quantisation and pruning methods for model compression, sketching in the context of model compression remains widely unexplored.

Sketching methods describe a family of probabilistic methods using random projections sampled from a probability distribution with the aim of dimensionality reduction. The core idea of sketching methods is to compress a large object such as a matrix or a stream of updates into a much smaller representation, the so-called sketch, that preserves the information needed for reconstructing the original object. Computation then proceeds on the sketch rather than on the higher dimensional data, often with large savings in terms of runtime, storage and communication costs. What distinguishes sketching from other compression methods is its reliance on randomness. A sketch is typically computed by applying a randomly drawn linear map $S \in \mathbb{R}^{d\times o}$ (also called the sketching matrix) to the input data such that an input matrix $A \in \mathbb{R}^{o\times m}$ is replaced by the sketch $SA \in \mathbb{R}^{d\times m}$ with $d \ll o$. Sketching methods are inherently probabilistic. First, through sampling of sketching matrices from a probability distribution, the algorithm itself is randomised. Its output is a random variable whose distribution is induced by the random choice of S, in contrast to other approaches that make use of a stochastic model of the data itself. Second, the inherent theoretical guarantees are probabilistic. Sarlos (2006); Woodruff (2014) have shown that the randomness in the output depends on the algorithm designer's choices of parameters such as target accuracy rather than the input, making sketching a powerful tool with input data-independent theoretical guarantees. This

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sets standard sketching apart from analyses that depend on distributional assumptions about the data. While being widely used in the field of data compression, sketching methods in relation to low-rank model compression remain largely unexplored. As transformers are the dominant building blocks of modern LLMs, in this work we focus on compression techniques for transformer-based architectures.

Low-rank model compression methods take a distinctly linear-algebraic view, assuming that most of the model's weight matrices carry considerable redundancy. Large weight matrices $W \in \mathbb{R}^{o \times m}$ can then be replaced by the product of smaller low-rank approximations, i.e., $W \approx UV^T$ with $U \in \mathbb{R}^{o \times k}, V \in \mathbb{R}^{m \times k}$ and $k \ll \min(o, m)$. Factoring out the weight matrices reduces the number of parameters and thus computation and memory costs. By the Eckart-Young theorem (Eckart & Young, 1936), the best rank-k approximation with respect to reconstruction fidelity of the weights, i.e., $\|W - \hat{W}\|_F$ for a reconstruction $\hat{W}$, is given by the truncated singular value decomposition (SVD) of W. However, *the best approximation of the weights is not necessarily the best approximation of the model*. The Eckart-Young theorem focusses on the reconstruction error $\|W - \hat{W}\|_F$, which treats all directions in weight space equally. In practice, directions in weight space are assigned varying importances based on the layer's input, leading to the question of finding the best approximation in activation space, i.e., $\|WX - \hat{W}X\|_F$ for layer inputs X.

To bridge the gap between sketching and low-rank model compression, we propose a data-aware sketching method by learning the sketching distribution through training of a variational autoencoder (VAE) (Kingma & Welling, 2014). As we aim to successfully compress transformer blocks as computationally efficient as possible, we deliberately opt to train a simple VAE rather than an expensive generative model. We first generate a training dataset of sketching matrices by generalising the Eckart-Young theorem from the weight to the activation space, finding optimal $\hat{W}$. Given $\hat{W}$, the corresponding sketching matrix S for training is computed via solving the linear least squares problem $\hat{W}X = S\ WX$ for each layer *in parallel*. Using this training dataset of sketching matrices, we then learn the distribution of sketching matrices by training the VAE. At inference time, we sample from the VAE to obtain sketching matrices S. We evaluate the proposed approach on LLM compression tasks for GPT-2 Small (Radford et al., 2019), GPT-2 Medium (Radford et al., 2019) and the Gemstone-256x23 (McLeish et al., 2026) models and show that the proposed method outperforms data-oblivious sketching methods. To highlight the applicability of the approach, we show that for depth upscaling and domain adaptation, finetuning the VAE decoder is sufficient and no retraining from scratch is needed.

Our contributions are the following: **1)** We propose a new sketching method that learns the distribution of sketching matrices for low-rank model compression. **2)** We show empirically that in contrast to traditional sketching methods, our approach performs well for LLM compression tasks - outperforming data-oblivious sketching methods including randomised SVD (RSVD) (Halko et al., 2011) and CUR decomposition (Mahoney & Drineas, 2009). **3)** We highlight the applicability of the proposed approach through empirical experiments on depth upscaling and domain adaptation task - showing that VAE finetuning rather than retraining is sufficient.

2 Related work

While there is a line of work on post-training model compression within the machine learning community, sketching methods for model compression are scarce. Empirical evidence has shown that traditional sketching approaches such as randomised singular value decomposition (RSVD) (Halko et al., 2011) and CUR decomposition (Mahoney & Drineas, 2009) might not lead to an effective low-rank model due to insufficient downstream performance. In the field of learning sketching matrices, Indyk et al. (2019) proposed to learn a specific instance of a sketching matrix by solving an optimisation problem to determine the values of the non-zero entries of the matrix. A follow-up paper (Indyk et al., 2021) showed that a few data points are sufficient to learn a specific instance of the sketching matrix. Other approaches for learning sketching matrices include those of Li et al. (2023a) where in addition to learning the non-zero values of the sketching matrix, the position of the zero entries is learned. Other methods for learning a specific instance of a sketching matrix include those of Liu et al. (2023) and Chen & Tao (2022).

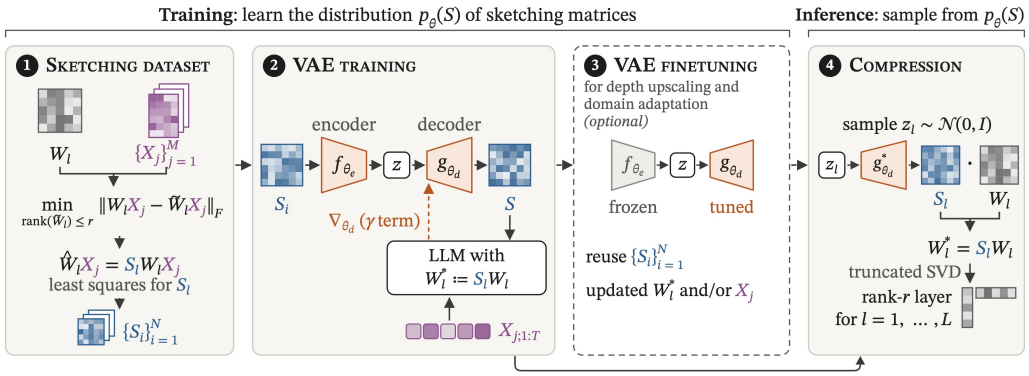


Figure 1: Overview of proposed method for learning the distribution of sketching matrices for LLM compression. First, a sketching training dataset is computed to train the VAE. Depending on the task, VAE can be finetuned before compressing the model by sampling from the trained VAE.

In addition to learning sketching matrices for data compression, Li et al. (2021) proposed learning-augmented sketches for Hessians. Another line of work similar to learning of sketching matrices focusses on learning projections for optimisation (Sakaue & Oki, 2024; Iwata & Sakaue, 2025; Iwata & Futami, 2026). For sketching of data streams, methods such as those of Cao et al. (2023; 2024); Feng et al. (2025) have been proposed. In the learning theory literature, sketching methods have been studied to establish generalisation bounds for neural networks (see e.g., (Bartlett et al., 2022; Sakaue & Oki, 2023)).

While current sketching methods typically rely on sampling from a given distribution of sketching matrices rather than reusing a specific sketching matrix (Ahfock et al., 2023), to the best of our knowledge, no approach has been proposed yet for learning the distribution of sketching matrices rather than a specific instance. A more detailed list of related work can be found in Appendix A.1.

3 Learning the Distribution of Sketching Matrices for Model Compression

In the following, we first give an overview of the proposed method before describing both the training and the inference steps in detail.

Overview. Figure 1 provides an overview of the proposed method. The overall aim is to learn the optimal distribution of sketching matrices where optimal refers to the least loss in performance on a downstream task under a given compression factor. Generative models have proven to be effective tools for approximating unknown underlying probability distributions given a set of training data. We thus aim to train a generative model that enables efficient sampling of sketching matrices via learning the distribution of sketching matrices for transformer blocks. To do so, we need to have access to training sketches. This is a major technical challenge since such training dataset is not known in advance. We therefore phrase this as an approximation problem where the aim is to approximate an unknown linear operator, i.e., the sketching matrix.

For given weight matrices W_l of a transformer trained on the downstream task and an optimal compressed version of these weight matrices $\hat{W}_l$, the problem can be expressed as $\hat{W}_l X_j = S_i W_l X_j$ where $\{X_j\}_{j=1}^M$ denote input samples on the downstream task, and S_i is the sketching matrix we aim to compute. Since both S_i and $\hat{W}_l$ are unknown, an optimal compressed $\hat{W}_l$ for any given uncompressed W_l has to be computed first. For low-rank approximation of matrices, the Eckart-Young theorem Eckart & Young (1936) states that the truncated SVD of W is the best rank-r approximation with respect to reconstruction fidelity. As previous work has shown that optimal reconstruction fidelity in weight space does not necessarily relate to sufficient performance on the downstream tasks (Yu & Wu, 2023), we aim to take the layer's inputs into account and optimise $\tilde{W}_l$, which is a rank-r approximation of the original weight matrix W_l, by minimising the discrepancy in activation

space, i.e., $\|W_l X_j - \tilde{W}_l X_j\|_F$. In this case, we cannot apply Eckart-Young directly but rather have to find a way of satisfying the condition on the rank of $\tilde{W}_l$ while minimising in activation space. We show that this is possible by computing a compact SVD of our input samples X_j, which we use to compute training sketches S_i. In the following, we detail each of the steps required for training of the generative model and inference.

3.1 Training of the generative model

The training of the generative model consists of three distinct steps: finding an optimal $\hat{W}_l$ by solving a minimisation problem in activation space. We then compute training sketches S_i, and train the generative model. In the following, we will detail each of those three steps.

Computing training sketches S_i. To learn the distribution of sketching matrices p_θ, we compute training sketches $\{S_i\}_{i=1}^N$ to incur the least performance degradation on a given downstream task. We can formulate the problem of finding training sketches S_i as approximating an unknown linear operator. In contrast to previous approaches, we aim to incorporate a subset of the training data for the downstream task to find the optimal distribution of sketching matrices. The problem is as follows

$$\hat{W}_l X_j = S_i W_l X_j, \tag{1}$$

for a weight matrix W_l, compressed weight matrix $\hat{W}_l$ and training data samples $\{X_j\}_{j=1}^M$. We compute training sketches S_i by first finding optimal compressed matrices $\hat{W}_l$ with corresponding weight matrices W_l of the transformer trained for the downstream task. The Eckart-Young theorem (Eckart & Young, 1936) states that given a matrix $A \in \mathbb{R}^{m\times n}$ and a desired rank $r \in [0, \min\{m, n\}]$, the best rank-r approximation in terms of reconstruction fidelity, i.e.,

$$\hat{A} = \arg \min_{\text{rank}(\tilde{A})\leq r} \|A - \tilde{A}\|_F, \tag{2}$$

which can be obtained by computing its truncated singular value decomposition. We however aim to incorporate a subset of the training data for the downstream task to find the optimal distribution of sketching matrices. Hence, rather than formulating the goal as achieving the best reconstruction fidelity on the weight matrices itself, we aim to minimise the distance between the uncompressed and compressed activations, i.e.,

$$\min_{\text{rank}(\tilde{W}_l)\leq r} \|W_l X_j - \tilde{W}_l X_j\|_F.$$

Finding optimal $\hat{W}_l$ for minimisation problem in activation space. In contrast to Equation (2) where both the minimisation problem and the condition on the rank are defined on the weights directly, we instead aim to minimise the Frobenius norm in activation space while restricting the rank of the weight matrix. The training data samples for the downstream task X_j, can be seen as a weighting factor and the challenge shifts to preserving the condition on the rank of $\tilde{W}_l$, i.e., $\text{rank}(\tilde{W}_l) \leq r$. As we cannot assume that $X_j \in \mathbb{R}^{m\times n}$ is full rank, we perform a compact SVD on X_j such that $X_j = U\Sigma V^T$ with $\text{rank}(X_j) = k$, $U \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$ and $V^T \in \mathbb{R}^{k\times n}$. As the Frobenius norm is unchanged by orthonormal-column factors, we have

$$\min_{\text{rank}(\tilde{W}_l)\leq r} \|(W_l - \tilde{W}_l)U\Sigma V^T\|_F = \min_{\text{rank}(\tilde{W}_l)\leq r} \|(W_l - \tilde{W}_l)U\Sigma\|_F.$$

We now define $B := W_l U\Sigma$ and $\tilde{B} := \tilde{W}_l U\Sigma$. To ensure that the rank of $\tilde{W}_l$ is at most r, we have to show that $\text{rank}(\tilde{W}_l) = \text{rank}(\tilde{B}\Sigma^{-1}U^T) \leq r$. For simplicity, we define $C := \Sigma^{-1}U^T$ and note $W_l = BC$. It then follows that

$$\text{rank}(BC) \leq \text{rank}(B) = \text{rank}(BC(U\Sigma)) = \text{rank}((BC)U\Sigma) \leq \text{rank}(BC).$$

Therefore, $\text{rank}(BC) = \text{rank}(B)$. The same also holds for $\tilde{W}_l$, i.e., for $\text{rank}(\tilde{B}C) = \text{rank}(\tilde{B})$. It then follows that

$$\min_{\text{rank}(\tilde{W}_l)\leq r} \|(W_l - \tilde{W}_l)U\Sigma\|_F = \min_{\text{rank}(\tilde{B})\leq r} \|(B - \tilde{B})\|_F.$$

Using the Eckart-Young theorem, the optimal compression for B can be computed as the truncated SVD, i.e.,

$$\hat{B} = \tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T.$$

Knowing $\hat{B}$, we can now solve $\hat{W}_l U \Sigma = \hat{B}$ for $\hat{W}_l$, i.e.,

$$\hat{W}_l = [\tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T] \Sigma^{-1} U^T. \tag{3}$$

Computing training sketches S_i for given $\hat{W}_l$. By creating optimal $\hat{W}_l$ in Equation (3), the only remaining unknown in Equation (1) is S_i since the weight matrix W_l is known. Assuming the set of downstream task training samples $\{X_j\}_{j=1}^M$ is large enough, we solve the resulting linear least squares problem for S_i and by that construct training samples for the generative model. It should be noted here that the choice of $\hat{W}$ induces a bias on the sketching matrices and thus on the distribution the generative model learns.[1] As a subset of downstream task training samples $\{X_j\}_{j=1}^M$ is used, our approach is highly adaptive and incorporates prior knowledge about the downstream task in the sketching distribution. This is in contrast to existing methods which learn a specific sketching matrix based on weight matrices only.

Constructing the loss function for training a VAE. Rather than learning a concrete instance of a sketching matrix, we propose to learn an "optimal" distribution of sketching matrices to sample from where optimal refers to the performance on a downstream task.

Concretely, we train a generative model $p_\theta(S, z)$ with model parameters θ, sketches $S \in \mathcal{S}$ and latent variables $z \in Z$. The marginal likelihood of the data can be described as

$$p_\theta(S) = \int_Z p_\theta(S, z) dz = \int_Z p_\theta(S \mid z) p(z) dz.$$

We then need to define the prior over the latents, i.e., $z \sim p(z)$ which is typically set to $\mathcal{N}(0, I)$ (Kingma & Welling, 2014; Higgins et al., 2017). Additionally, to generate data samples, we need to define $S \mid z \sim p_\theta(S \mid z)$, the so-called decoder. For most generative models, this is typically defined as a deterministic function $g_\theta : Z \to \mathcal{S}$, for example a trained neural network. Sampling data points from the generative model thus consists of drawing $z \sim p(z)$ and computing $S = p_\theta(S \mid z)$.

Training the model is done by maximising the likelihood, i.e.,

$$\theta^* = \arg\max \sum_{i=1}^{N} \log p_\theta(S_i),$$

where $\{S_i\}_{i=1}^N \sim p_{\text{data}}$ denote the samples of training sketches. This is equivalent to minimising the KL divergence $D_{KL}(p_{\text{data}} || p_\theta)$ up to a constant.

Variational autoencoders (VAEs) (Kingma & Welling, 2014) are neural networks that combine an encoder-decoder architecture with probabilistic modelling and thus are well suited for learning probability distributions. The encoder $f_{\theta_e} : \mathcal{S} \to Z$ encodes an input sketch S_i into a compact latent representation z_i. In the next step, the decoder $g_{\theta_d} : Z \to \mathcal{S}$ takes the latent code and maps it back into the space of sketching matrices, i.e., learning $p_{\theta_d}(S \mid z)$. The latent distribution is usually chosen to be a multivariate Gaussian, making it easy and cheap to sample from the VAE. There exist a number of VAE variants. One of the most commonly used variants is the β-VAE (Higgins et al., 2017). β-VAEs are trained incorporating a β weighting the reconstruction loss term against a regularisation term. The standard β-VAE loss function does not take any performance on the downstream task into account. Our ultimate goal, however, is not to accurately generate samples from the distribution of training sketches obtained by solving a linear operator (cf. Equation (1)). Our target distribution of sketching samples leads to low-rank weight matrices that maximise the probability assigned to the correct next token in a transformer. We thus add an additional cross-entropy loss term[2] that ensures sufficient performance on the downstream task. We propose the following loss

[1] While in this example we make use of the Eckart-Young theorem to construct optimal $\hat{W}_l$ with respect to low-rank approximations, this method is not restricted to a specific low-rank approximation method.

[2] Perplexity itself is not suitable as part of a loss function as it scales exponentially. As perplexity is $\exp(CE)$ where CE is the cross-entropy, we choose a cross-entropy loss term instead.

function for training VAE:

$$\mathcal{L}(\theta_d, \theta_e, S) = -\mathbb{E}_{q_{\theta_e}(z|S)}\big[\log p_{\theta_d}(S \mid z)\big] + \beta D_{KL}\big(q_{\theta_e}(z \mid S) \,||\, p(z)\big) - \gamma \frac{1}{T-1} \sum_{t=1}^{T-1} \log p_{\text{LLM}}\big(X_{j;t+1} \mid X_{j;\leq t}; \{W_l^*\}_{l=1}^{L}\big),$$

for layers $l \in \{1, \ldots, L\}$, a token window $X_{j;1:T}$ drawn from a calibration corpus and $p_{\text{LLM}}\big(X_{j;t+1} \mid X_{j;\leq t}; \{W_l^*\}_{l=1}^{L}\big)$ being the conditional probability distribution over the next element of the sequence of the LLM. W_l^* is computed via sampling a z from the current latent space of the VAE, computing the corresponding S_i via the decoder and setting $W_l^* := S_i W_l$. During backpropagation, the gradients with respect to the γ-loss term thus are only propagated through the decoder and the cross-entropy loss term penalises large variance in the output distribution of the decoder only.

3.2 Model compression via sampling from learned sketching distribution

At inference time, we use the trained VAE for model compression. The model compression pipeline looks as follows: First, for every layer l, we sample a $z_i \sim \mathcal{N}(0, I)$ and secondly use the trained VAE to compute a corresponding S_i as $S_i = g^*_{\theta_d}(z_i)$ where $g^*_{\theta_d}$ is the trained decoder. This S_i can then be used to compress layer l's weight matrix by cheaply computing ${W_l}^* = S_i W_l$ once. Once trained, the VAE thus provides an efficient way of sampling from the distribution of sketching matrices that optimise the performance of the low-rank transformer on the downstream task. The proposed approach thus enables cheap compression of the model. While at training time, we ensure the rank of W_l^* to be within the desired range, at inference time, we satisfy the rank condition by performing a truncated SVD on W_l^*.

3.3 Finetuning the trained VAE for depth upscaling and domain-adaptation

In practice, a small and carefully designed transformer block is often used for depth upscaling, i.e., increasing the depth of the neural network by increasing the number of layers. Depth upscaling provides a computationally efficient way of increasing model size while reusing the pretrained weights of a comparable smaller transformer. Similarly, in practice, especially for large models, a pretrained model is adapted to a specific domain via finetuning of the pretrained model, rather than training the model from scratch. As highlighted in Subsection 3.2, training the VAE from scratch is the computationally expensive part of the method. Once trained, the transformer can be compressed efficiently by sampling from $\mathcal{N}(0, I)$, computing a forward pass of the decoder $g_{\theta_d} : Z \to \mathcal{S}$ and a matrix-matrix multiplication $S_i W_l$ per layer l of the model. To avoid retraining the VAE from scratch for depth upscaling and domain-adaptation, we propose an efficient finetuning stage. We reuse the training sketches $\{S_i\}_{i=1}^{N}$ for finetuning, which eliminates the costs for recomputing new samples. In addition, we fix the encoder's weights θ_e and only finetune the decoder's weights θ_d. The finetuning loss function is given by

$$\mathcal{L}(\theta_d, \theta_e, S) = -\mathbb{E}_{q_{\theta_e}(z|S)}\big[\log p_{\theta_d}(S \mid z)\big] - \gamma \frac{1}{T-1} \sum_{t=1}^{T-1} \log p_{\text{LLM}}\big(X_{j;t+1} \mid X_{j;\leq t}; \{W_l^*\}_{l=1}^{L}\big),$$

where W_l^* and/or X_j are updated for depth upscaling and domain-adaptation, accordingly. As the weights of the encoder are fixed, we omit the β loss term for finetuning.

4 Experimental evaluation

In the following, we first detail the experimental setup before highlighting the results of the proposed method on LLM compression, depth upscaling experiments and domain-adaptive finetuning experiments. Further experimental results and a detailed depiction of all hyperparameters can be found in Appendix A.2 and A.3, respectively.

Table 1: Overview of perplexity for LLM compression for the proposed method (VAE), SVD, RSVD and CUR decomposition. *For compression ratios of $\leq 30\%$, our method outperforms even the uncompressed baseline for GPT-2 Small and GPT-2 Medium models*. For GPT-2 Medium and Gemstone-256x23 models, the proposed approach outperforms the baselines for all compression ratios.

Model	Compression ratio	SVD	RSVD	CUR	VAE
GPT-2 Small	0.1	25.09	25.03	1.9×10^7	**22.18** $\pm$**0.02**
	0.2	25.98	25.98	4.4×10^7	**22.91** $\pm$**0.08**
	0.4	**31.19**	31.65	2.2×10^8	33.93 $\pm$0.61
	Uncompressed: 24.36				
GPT-2 Medium	0.1	18.26	18.23	1.1×10^9	**15.73** $\pm$**0.01**
	0.2	18.46	18.55	1.9×10^{10}	**16.65** $\pm$**0.00**
	0.3	22.04	20.75	1.2×10^8	**17.90** $\pm$**0.03**
	0.4	179.05	102.58	1.5×10^9	**20.84** $\pm$**0.03**
	0.5	1794.19	1137.70	3.1×10^8	**53.01** $\pm$**2.69**
	Uncompressed: 18.00				
Gemstone-256$\times$23	0.1	36.09	36.09	3.4×10^6	**34.68** $\pm$**0.04**
	0.2	41.87	42.82	2.1×10^7	**39.41** $\pm$**0.05**
	0.3	57.31	57.29	1.9×10^6	**44.93** $\pm$**0.17**
	0.4	89.27	98.12	6.4×10^6	**55.14** $\pm$**0.04**
	0.5	171.42	195.11	9.5×10^6	**77.36** $\pm$**0.21**
	Uncompressed: 32.53				

4.1 Experimental setup

We evaluate the proposed sketching method for model compression. We train and evaluate on a variety of transformer-based models and datasets. For comparing against baselines, we use a pretrained GPT-2 Small model (Radford et al., 2019) and GPT-2 Medium model (Radford et al., 2019), both pretrained on the OpenWebText dataset (Gokaslan & Cohen, 2019). In addition to that, we train and evaluate for the Gemstone-256x23 model (McLeish et al., 2026), pretrained on the Dolma dataset (Soldaini et al., 2024). The VAE is trained for 1000 epochs on the training set of the Wikitext-2 dataset (Merity et al., 2017) and all models are evaluated on the Wikitext-2 test dataset. To stabilise VAE training, we apply gradient clipping as well as warm starts for β and γ. For depth upscaling, the Gemstone models are pretrained on the Dolma dataset and evaluated on the Wikitext-2 test dataset. The domain adaptation experiments are performed through finetuning both the GPT-2 Small and GPT-2 Medium model on the Wikitext-2 training dataset for three epochs over the full dataset. The VAE is finetuned for 200 epochs on the Wikitext-2 validation dataset instead. For the model compression step, we restrict our compression to the hidden dimensions of the transformers for all methods to ensure comparability. All results are given as mean $\pm$ standard deviation over 3 distinct runs with different seeds. A full list of hyperparameter settings for training and evaluation is provided in Appendix A.3.

Training the VAE requires finding appropriate ranges for both regularisation parameters β and γ. Through numerical experiments, we narrowed down that a $\beta \in [10, 100]$ and $\gamma \in [100, 1000]$ achieves best results. While smaller values of β lead to a loss in performance and instabilities during training, larger values do not seem to affect the training of the VAE as much. It is therefore recommended to start with a larger choice of β if undecided. The only effect of a γ value that has been chosen as too small, however, is a loss in performance. The larger the γ in relation to β, the more the VAE loses its ability to approximate the underlying distribution of sketching matrices and the higher the chances are for a latent collapse of the VAE to occur – turning the VAE into a standard autoencoder instead. While an appropriate choice of the regularisation parameters thus seems crucial for training the VAE, the suitable ranges for both β and γ are sufficiently large – enabling training without an extensive need for finetuning of those hyperparameters.

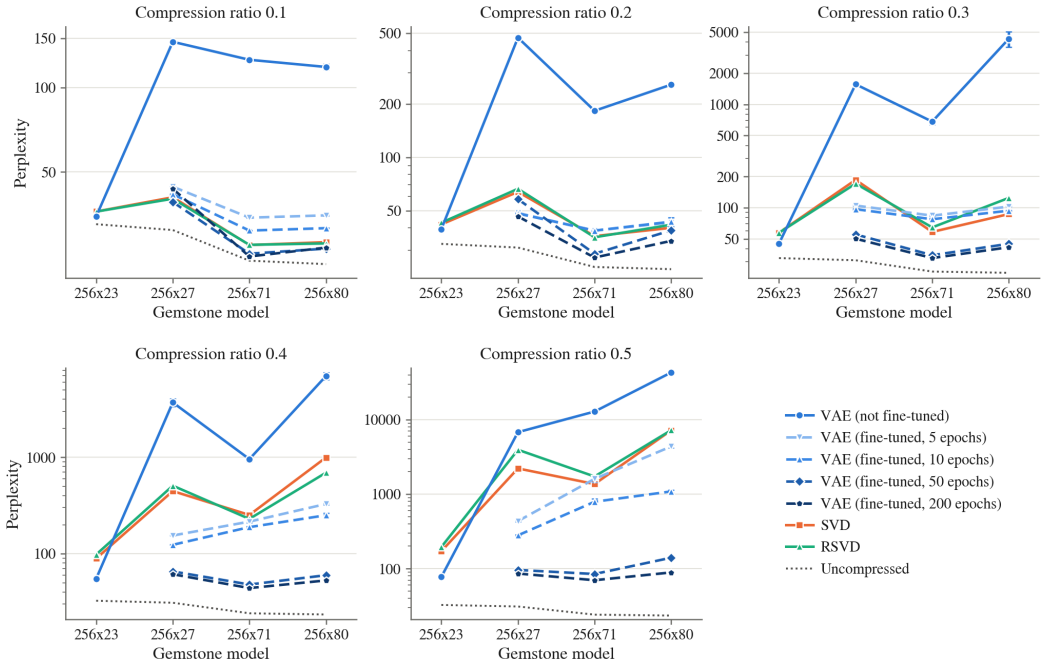


Figure 2: Gemstone depth upscaling for the proposed method (VAE) and the proposed method with finetuning for a varying number of finetuning epochs vs. SVD, RSVD and the uncompressed baseline. For finetuning of the VAE of ≥ 50 epochs, our method with finetuning consistently outperforms the other baselines for all compression ratios. For larger compression ratios (≥ 0.4), finetuning for 10 epochs achieves perplexity values smaller than the SVD and RSVD baselines.

4.2 Learned sketching for LLM compression

We apply the proposed method to LLM compression, comparing the perplexity of the learned sketching approach against truncated singular value decomposition (SVD) (Golub & Van Loan, 2013) as an optimal low-rank approximation for the reconstruction baseline, as well as randomised singular value decomposition (RSVD) and CUR decomposition as data-oblivious sketching baselines. Table 1 highlights the model perplexity for a GPT-2 Small, GPT-2 Medium and Gemstone-256x23 model. The proposed method outperforms the uncompressed baseline model for compression ratios of up to 30% for the GPT-2 Small and GPT-2 Medium models, highlighting the effectiveness of the proposed approach. For the GPT-2 Medium and Gemstone-256x23 models, our method additionally outperforms SVD significantly, as well as both data-oblivious sketching methods.

4.3 Depth upscaling through VAE finetuning

To highlight the applicability of the proposed approach, we evaluate it for depth upscaling. Depth upscaling is a computationally efficient way of increasing the model size by increasing the depth of the model. As the width of the layers is usually kept to enable reusage of pretrained weights of a comparable smaller model, we test reusing the VAE trained for the shallow model. Figure 2 highlights the performance of the proposed approach both with and without finetuning the VAE for a varying number of epochs for the Gemstone-256 series. *Even though the VAE was trained for the shallowest Gemstone model with* 23 *layers only, we show that finetuning the VAE for as little as* $10 - 50$ *epochs is sufficient to outperform the SVD, RSVD and CUR baselines.* Particularly for high compression ratios, the proposed approach outperforms the baselines by several orders of magnitude.

Table 2: Overview of perplexity for domain-adaptation experiments for GPT-2 Small and GPT-2 Medium models. While the proposed method (VAE) with finetuning achieves similar results to truncated SVD for the GPT-2 Small model, our method applied on the GPT-2 Medium model (with and without finetuning) outperforms the SVD, RSVD and CUR baselines.

Model	Compression ratio	SVD	RSVD	CUR	VAE (not finetuned)	VAE (finetuned)
GPT-2 Small	0.1	**18.90**	18.91	3.5×10^8	22.28 $\pm$0.02	18.92 $\pm$0.01
	0.2	**19.58**	19.63	2.3×10^7	24.45 $\pm$0.20	19.62 $\pm$0.02
	0.4	**22.89**	23.18	4.9×10^7	35.36 $\pm$0.59	23.52 $\pm$0.05
	Uncompressed: 18.56					
GPT-2 Medium	0.1	14.16	14.19	6.5×10^6	15.86 $\pm$0.01	**14.09 $\pm$0.00**
	0.2	**14.58**	14.65	5.7×10^8	16.06 $\pm$0.02	14.61 $\pm$0.00
	0.3	15.82	15.79	2.4×10^8	16.90 $\pm$0.01	**15.62 $\pm$0.01**
	0.4	45.45	27.02	2.1×10^9	**19.75 $\pm$0.02**	20.56 $\pm$0.13
	0.5	654.72	428.94	7.1×10^7	**41.61 $\pm$1.13**	134.94 $\pm$2.56
	Uncompressed: 13.98					

4.4 Domain-adaptive finetuning

Another important computationally efficient application is domain-adaptive finetuning. Instead of retraining a large language model from scratch, a pretrained version of the model is adapted for the desired domain through finetuning. Table 4.4 highlights the perplexity of a GPT-2 Small and GPT-2 Medium model for domain-adaptive finetuning on OpenWebText. While for the shallower GPT-2 Small model, the finetuned VAE achieves similar perplexity values as truncated SVD, the best performing baseline and the proposed method with and without finetuning outperforms all baselines for nearly all compression ratios for the deeper GPT-2 Medium model. Interestingly, VAE finetuning seems to be mostly relevant for small compression ratios of up to 30%, whereas the non-finetuned VAE outperforms the finetuned version for larger compression ratios. *This highlights the applicability of the proposed approach for domain-adaptation and shows that for large compression ratios even the VAE finetuning step may not be needed.*

5 Conclusion and future work

We presented a data-aware sketching method for model compression. In contrast to existing data-aware sketching methods, we learn the distribution of sketching matrices rather than only a specific instance by training a VAE. Through empirical experiments on several LLMs and datasets we show that the proposed approach performs well for low-rank model compression and outperforms data-oblivious sketching methods as well as truncated SVD. For compression ratios $\leq 30\%$, we even outperform the uncompressed baseline model in terms of perplexity for both the GPT-2 Small and GPT-2 Medium models. We highlight the applicability of our method through depth upscaling and domain-adaptation experiments. Using the Gemstone model series, we highlight that for depth upscaling finetuning of the VAE for as little as 10 epochs is sufficient for high compression ratios $\geq 40\%$. For smaller compression ratios a finetuning step for 50 epochs suffices to outperform the baseline methods. Similarly, we show that for domain-adaptation on the GPT-2 Medium model, finetuning the VAE leads to better results for compression ratios up to 30%. Reusing the VAE without finetuning outperforms the baselines for higher compression ratios.

A central point of training the VAE to learn the distribution of sketching matrices is how to generate the training dataset of sketching matrices. In this paper, we focus on low-rank approximations and thus made use of the Eckart-Young theorem to compute optimal $\hat{W}$ to then infer the corresponding S, as it provides theoretical optimality guarantees. Equation (1) however only requires us to have access to input samples X_j and compressed weight matrices $\hat{W}$. The proposed method can thus be used for other model compression methods as well. The overall training, inference and finetuning pipeline remains unchanged, and the only difference is the way of computing $\hat{W}_l$ to infer the training sketches S_i. Another interesting future line of work is the extension from model compression back to traditional data compression applications.

AI USE STATEMENT

In this work, we have not used generative AI tools for help in developing theoretical models or conceptual frameworks, formulation of mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning and reformatting of datasets or interpreting results, and assisting with translation, generation of synthetic datasets, supporting qualitative and thematic data analysis are not applicable to this work.

We used generative AI tools for creating scientific figures, creating and editing software code, summarising existing literature and identifying relevant literature. We have reviewed all AI-assisted work. We verified LLM-generated code and tested it for correctness. We checked that the relevant literature identified by the LLM was not hallucinated and made sure that literature summaries were accurate. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the full experimental protocol including models, hyperparameters as well as training and evaluation datasets to enhance reproducibility in Subsection 4.1 and Appendix A.3. While Subsection 4.1 provides the main setup, Appendix A.3 further details the experimental setup for the generation of the sketching dataset, VAE training, VAE finetuning and model compression for all the models evaluated as part of this work. The code for reproducing all experiments will be made available upon acceptance.

REFERENCES

Daniel Ahfock, William J Astle, and Sylvia Richardson. On randomized sketching algorithms and the tracy–widom law. *Statistics and Computing*, 33(1):34, 2023.

Peter Bartlett, Piotr Indyk, and Tal Wagner. Generalization bounds for data-driven numerical linear algebra. In *Conference on Learning Theory*, pp. 2013–2040. PMLR, 2022.

Yukun Cao, Yuan Feng, and Xike Xie. Meta-sketch: A neural data structure for estimating item frequencies of data streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6916–6924, 2023.

Yukun Cao, Yuan Feng, Hairu Wang, Xike Xie, and S Kevin Zhou. Learning to sketch: A neural approach to item frequency estimation in streaming data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(11):7136–7153, 2024.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

Arnav Chavan, Nahush Lele, and Deepak Gupta. RETHINKING COMPRESSION: REDUCED ORDER MODELLING OF LATENT FEATURES IN LARGE LANGUAGE MODELS. In *The Second Tiny Papers Track at ICLR 2024*, 2024. URL https://openreview.net/forum?id=BfVccaZiEv.

Tiejin Chen and Yicheng Tao. Learning sparsity and randomness for data-driven low rank approximation. *arXiv preprint arXiv:2212.08186*, 2022.

Shabarish Chenakkod, Michał Dereziński, and Xiaoyu Dong. Optimal subspace embeddings: Resolving nelson-nguyen conjecture up to sub-polylogarithmic factors. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 4501–4559. SIAM, 2026.

Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Michał Dereziński and Michael W Mahoney. Recent and upcoming developments in randomized numerical linear algebra for machine learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6470–6479, 2024.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Yuan Feng, Yukun Cao, Hairu Wang, Xike Xie, and S Kevin Zhou. Lego sketch: A scalable memory-augmented neural network for sketching data streams. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 16589–16603. PMLR, 13–19 Jul 2025.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `https://skylion007.github.io/OpenWebTextCorpus`, 2019.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh Tahaei, Omar Mohamed Awad, and Yang Liu. Compressing pre-trained language models using progressive low rank decomposition. *Advances in Neural Information Processing Systems*, 35:6–14, 2021.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53 (2):217–288, 2011.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=uPv9Y3gmAI5`.

Xing Hu, Dawei Yang, Yuan Cheng, Zhixuan Chen, and Zukang Xu. SAES-SVD: Self-adaptive suppression of accumulated and local errors for SVD-based LLM compression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=KMAYsQO8pU`.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1404–1416, 2022.

Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. *Advances in Neural Information Processing Systems*, 32, 2019.

Piotr Indyk, Tal Wagner, and David Woodruff. Few-shot data-driven algorithms for low rank approximation. *Advances in Neural Information Processing Systems*, 34:10678–10690, 2021.

Tomoharu Iwata and Futoshi Futami. Data-driven projection generation for efficiently solving heterogeneous quadratic programming problems. *Machine Learning*, 115(5):123, 2026.

Tomoharu Iwata and Shinsaku Sakaue. Learning to generate projections for reducing dimensionality of heterogeneous linear programming problems. In *ICML*, 2025.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR 2014)*, 2014.

Yi Li, Honghao Lin, and David P Woodruff. Learning-augmented sketches for hessians. *arXiv preprint arXiv:2102.12317*, 2021.

Yi Li, Honghao Lin, Simin Liu, Ali Vakilian, and David Woodruff. Learning the positions in countsketch. In *The Eleventh International Conference on Learning Representations*, 2023a. URL `https://openreview.net/forum?id=iV9Cs8s8keU`.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pp. 20336–20350. PMLR, 2023b.

Cuiyu Liu, Xiao Chuanfu, Mingshuo Ding, and Chao Yang. Tensor-based sketching method for the low-rank approximation of data streams. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=rOFKmzNTbC.

Defu Liu, Yixiao Zhu, Zhe Liu, Yi Liu, Changlin Han, Jinkai Tian, Ruihao Li, and Wei Yi. A survey of model compression techniques: past, present, and future. *Frontiers in Robotics and AI*, 12: 1518965, 2025.

Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Mehdi Makni, Kayhan Behdin, Zheng Xu, Natalia Ponomareva, and Rahul Mazumder. A unified framework for sparse plus low-rank matrix decomposition for LLMs. In *The Second Conference on Parsimony and Learning (Proceedings Track)*, 2025. URL https://openreview.net/forum?id=hyN75SAJTI.

Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

Sean McLeish, John Kirchenbauer, David Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws. *Advances in Neural Information Processing Systems*, 38:123419–123454, 2026.

Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 91–100, 2013.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLim: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4UfRP8MopP.

Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. Randomized numerical linear algebra: A perspective on the field with an eye to software. LAPACK Working Note LAWN 296, University of Tennessee, 2023.

Jeff M. Phillips. Coresets and sketches. In *Handbook of Discrete and Computational Geometry*. CRC Press, 3rd edition, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajarshi Saha, Varun Srivastava, and Mert Pilanci. Matrix compression via randomized low rank and low precision factorization. *Advances in Neural Information Processing Systems*, 36:18828–18872, 2023.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Shinsaku Sakaue and Taihei Oki. Improved generalization bound and learning of sparsity patterns for data-driven low-rank approximation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1–10. PMLR, 2023.

Shinsaku Sakaue and Taihei Oki. Generalization bound and learning methods for data-driven projections in linear programming. *Advances in Neural Information Processing Systems*, 37:12825–12846, 2024.

Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pp. 143–152. IEEE, 2006.

Pratyusha Sharma, Jordan Ash, and Dipendra Kumar Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *International Conference on Learning Representations*, volume 2024, pp. 17632–17651, 2024.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, 2024.

Jiayi Tian, Ryan Solgi, Jinming Lu, Yifan Yang, Hai Li, and Zheng Zhang. Flat-llm: Fine-grained low-rank activation space transformation for large language model compression. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2988–3002, 2026.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL `https://openreview.net/forum?id=gp32jvUquq`.

Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. In *International Conference on Learning Representations*, volume 2025, pp. 12561–12590, 2025b.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4287–4296, 2025c.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025d.

David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1-2):1–157, 2014.

Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. A survey of resource-efficient llm and multimodal foundation models. *arXiv preprint arXiv:2401.08092*, 2024.

Hao Yu and Jianxin Wu. Compressing transformers: Features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11007–11015, 2023.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 2024.

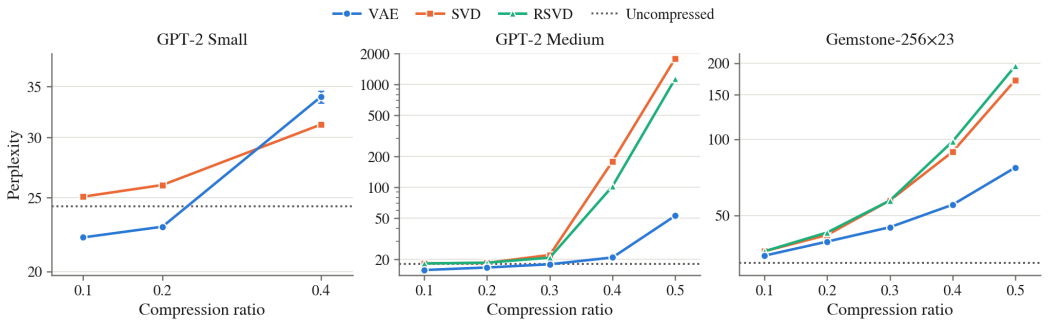


Figure 3: Perplexity comparison of the learned sketching approach vs. SVD, RSVD and the uncompressed baseline for a GPT-2 Small, GPT-2 Medium and Gemstone-256x23 model. For compression ratios $\leq 30\%$, the proposed method outperforms even the uncompressed baseline for the GPT-2 Small and GPT-2 Medium models. For all compression ratios, the learned sketching approach outperforms SVD and RSVD for the GPT-2 Medium and Gemstone model.

A Appendix

A.1 Related work

Sketching methods. Data-oblivious sketching methods have been developed since many years. While Mahoney (2011); Phillips (2017) and Martinsson & Tropp (2020) give an overview of more traditional data-oblivious methods, Murray et al. (2023) and Dereziński & Mahoney (2024) provide a more recent overview of the developments in the randomised numerical linear algebra field for machine learning. Data-oblivious sketching methods with sparse sketches include those of Charikar et al. (2004); Cormode & Muthukrishnan (2005); Clarkson & Woodruff (2017); Meng & Mahoney (2013); Chenakkod et al. (2026). A list of related work for data-aware sketching methods is provided in the main part of the paper.

Low-rank model compression methods. Low-rank model compression for LLMs is a relatively recent field of research. Surveys like those of Zhu et al. (2024), Liu et al. (2025) and Xu et al. (2024) provide an overview of the recent developments in the field. Recent work in data-free LLM compression includes those of Hajimolahoseini et al. (2021); Chavan et al. (2024); Sharma et al. (2024). While previous methods worked in weight space directly, Yu & Wu (2023) highlighted that while features are low rank, the weights are not. This lead to a new line of work working in activation space rather than weight space (see e.g., (Hsu et al., 2022; Hua et al., 2022; Yuan et al., 2023; Wang et al., 2025d;c;b;a; Tian et al., 2026; Hu et al., 2026). In addition to low-rank compression methods, combinations of low-rank approximations and sparsity as well as low-rank approximation and quantisation have been proposed (Li et al., 2023b; Makni et al., 2025; Mozaffari et al., 2025; Saha et al., 2023; 2024).

A.2 Additional experimental results

In the following, we will detail additional experimental results for the experiments on LLM compression A.2.1 and depth upscaling A.2.2. Subsubsection A.3 gives an overview of the experimental settings used for generation of the sketching training dataset, the VAE training, finetuning experiments and LLM compression.

A.2.1 Learned sketching for LLM compression

Figure 3 highlights the perplexities of the learned sketching approach vs. SVD, RSVD and the uncompressed baseline for a GPT-2 Small, GPT-2 Medium and Gemstone-256x23 model. For compression ratios of up to 30%, the proposed method even outperforms the uncompressed model. For all compression ratios, our method performs better than any of the baseline methods for the GPT-2 Medium and Gemstone-256x23 model.

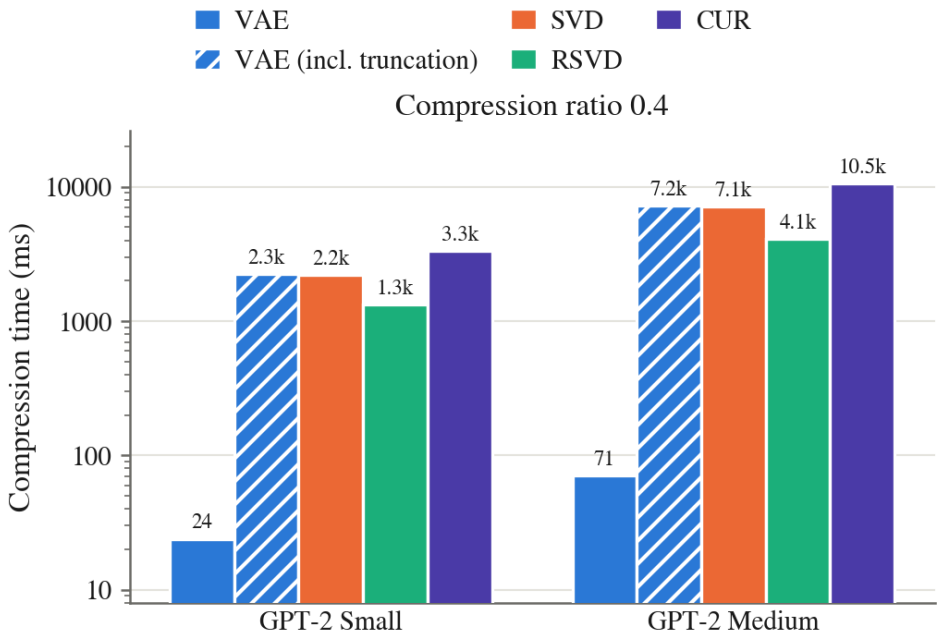


Figure 4: Comparison of runtimes for VAE, VAE including truncation to target rank, SVD, RSVD and CUR decomposition for the GPT-2 Small and GPT-2 Medium models. The proposed method VAE (incl. truncation) requires less time than computing the CUR decomposition and is equally fast as truncated SVD.

In addition to the experiments presented in the main part of the paper, figure 4 compares the runtimes for the proposed method (VAE) without additional truncation to the target rank and the proposed method with additional truncation to the target rank (VAE incl. truncation) to truncated SVD, RSVD and CUR decomposition. Compressing the LLM with the proposed method (VAE incl. truncation) takes less time than compressing via CUR decomposition and is equally fast as truncated SVD.

A.2.2 Depth upscaling through VAE finetuning

For depth upscaling, we additionally report the perplexity values for four Gemstone models with different depths (23, 27, 71 and 80 layers) while keeping the width of the models fixed to 256. An overview of the results can be found Table A.2.2. For the different compression ratios tested ($0.1 - 0.5$), finetuning the VAE for 50 epochs is sufficient to outperform the baseline methods for all depths.

A.3 Experimental details

Generation of sketching matrix dataset. The hyperparameters used for generating the sketching matrix datasets for the different LLMs can be found in Table 4.

VAE training. Table 5 provides an overview of the hyperparameters used for VAE training.

Finetuning experiments. Table 6 provides an overview of the hyperparameters used for the finetuning experiments.

Model compression experiments. Table 7 provides an overview of the hyperparameters used for the model compression experiments.

Table 3: Extensive overview of perplexity values for four different Gemstone models (256×23, 256×27, 256×71 and 256×80) for different compression ratios ($0.1 - 0.5$) and VAE finetuning epochs ($0, 5, 10, 50, 200$) against SVD, RSVD and CUR decomposition as baselines. The finetuned VAE outperforms the baseline methods for all LLM depths for finetuning for at least 50 epochs.

	Compression				VAE, finetuning epochs				
Gemstone	ratio	SVD	RSVD	CUR	0	5	10	50	200
256×23	0.1	36.09	36.09	3.4×10^6	**34.68**	–	–	–	–
	0.2	41.87	42.82	2.1×10^7	**39.41**	–	–	–	–
	0.3	57.31	57.29	1.9×10^6	**44.93**	–	–	–	–
	0.4	89.27	98.12	6.4×10^6	**55.14**	–	–	–	–
	0.5	171.42	195.11	9.5×10^6	**77.36**	–	–	–	–
	Uncompressed: 32.53								
256×27	0.1	40.60	40.10	1.7×10^7	145.68	44.28	41.60	**39.04**	43.55
	0.2	63.85	66.46	9.8×10^5	472.10	48.59	48.24	58.10	**46.43**
	0.3	186.03	171.00	6.1×10^6	1565.60	104.93	96.82	55.17	**50.12**
	0.4	445.94	507.43	2.0×10^6	3704.27	154.42	123.72	65.52	**61.28**
	0.5	2213.20	3941.25	2.7×10^7	6792.75	434.60	279.66	95.94	**85.93**
	Uncompressed: 31.01								
256×71	0.1	27.42	27.46	8.9×10^6	125.69	34.33	30.87	25.53	**24.95**
	0.2	35.98	35.44	1.2×10^7	182.78	38.62	38.64	28.63	**27.24**
	0.3	58.31	64.49	1.1×10^6	681.59	83.81	77.36	34.70	**32.56**
	0.4	252.23	229.66	1.7×10^6	957.45	215.18	188.72	47.92	**44.01**
	0.5	1366.33	1725.40	2.5×10^6	1.3×10^4	1603.35	790.43	84.62	**69.89**
	Uncompressed: 24.11								
256×80	0.1	28.07	27.78	4.6×10^6	118.48	34.99	31.49	**26.62**	26.84
	0.2	40.07	41.71	6.4×10^7	256.61	42.98	43.37	38.76	**33.82**
	0.3	86.80	124.42	6.1×10^6	4316.16	102.52	93.62	44.75	**41.45**
	0.4	996.14	692.39	4.1×10^6	6958.12	328.01	250.63	59.85	**52.86**
	0.5	7188.30	7218.17	9.8×10^6	4.3×10^4	4368.04	1087.29	140.06	**88.85**
	Uncompressed: 23.43								

Table 4: Hyperparameters for generation of sketching matrix dataset.

Setting	GPT-2 Small / Medium	Gemstone-256×23
Calibration corpus	`stas/openwebtext-10k`, train split	`stas/openwebtext-10k`, train split
Tokens per example	32	32
Group sizing margin	2.0	2.0
Max. out-features	4096	4096
Pseudo-inverse rtol	10^{-3}	10^{-3}
Batch size	8	8
S samples pooled	360 (Small), 288–432 (Medium)	3105
Compression ratios	$1 - r$ for $r \in \{0.5, \ldots, 0.9\}$	$1 - r$ for $r \in \{0.5, \ldots, 0.9\}$

Table 5: Hyperparameters for VAE training.

Setting	GPT-2 Small	GPT-2 Medium	Gemstone-256×23
S shape (out-features)	768×768	1024×1024	256×256
Latent dimension	16	16	32
Base / max channels	16 / 128	16 / 128	16 / 128
Bottleneck size	12	12	12
KL weight β	{10, 20, 30, 50}	{10, 20, 30, 50}	{1, 20}
Perplexity weight γ	{100, 1000, 10000}	{100, 1000, 10000}	1000
Epochs	1000	1000	1000
Learning rate	10^{-3}	10^{-3}	10^{-3}
Batch size	32	32	32
Gradient clipping	1.0	1.0	1.0
Validation fraction	0.1	0.1	0.1
KL warmup	10% of epochs	10% of epochs	10% of epochs
γ term start	10% of epochs	10% of epochs	10% of epochs
γ term corpus	WikiText-2 train	WikiText-2 train	WikiText-2 train
γ term window	512 tokens	512 tokens	512 tokens

Table 6: Hyperparameters for finetuning experiments.

Setting	Value
Language-model finetuning	
Corpus	WikiText-2 (raw), train split
Epochs	3
Learning rate	5×10^{-5}, warmup ratio 0.03
Batch size	8, block size 1024
Weight decay	0.01, gradient clip 1.0
VAE finetuning against the finetuned LM	
β / γ	20 / 1000 (Small), 10 / 1000 (Medium)
Epochs	200, best epoch selected, epoch 0 a candidate
Learning rate	10^{-4}
Warmups	none (β and γ full from epoch 1)
Calibration split	WikiText-2 validation
VAE depth finetuning	
Target models	Gemstone-256×27, 256×71
finetuning length	{5, 10, 20, 35, 50, 200} epochs
Encoder	frozen (decoder only)

Table 7: Hyperparameters of model compression experiments.

Setting	Value
Baselines	
SVD	truncated SVD at the same rank fraction
RSVD	10 oversamples, 2 power iterations
CUR	leverage-score column/row sampling at the same rank fraction
Seeds (randomised methods)	$\{0, 42, 312\}$
Perplexity evaluation	
Corpus	WikiText-2 (raw), test split
Sliding window	stride 512, window = model max. length (1024)
OpenWebText check	`stas/openwebtext-10k`, ranks 0.5 and 0.8
Compressed layers	24 of 49 (Small), 48 of 97 (Medium), 69–240 (Gemstones)
Compression timing	
Device / dtype	CUDA, float64
Repeats	10 timed, 3 warmup; per-layer median, summed over layers
Ranks	0.6 and 0.9 (compression ratios 0.4 and 0.1)
VAE time	sampling S plus SW; "incl. truncation" adds the SVD of SW