ESPACE: Dimensionality Reduction of Activations for Model Compression

Charbel Sakr
NVIDIA Research
csakr@nvidia.com

Brucek Khailany
NVIDIA Research
bkhailany@nvidia.com

Abstract

We propose ESPACE[1], an LLM compression technique based on dimensionality reduction of activations. Unlike prior works on weight-centric tensor decomposition, ESPACE projects activations onto a pre-calibrated set of principal components. The activation-centrality of the approach enables retraining LLMs with no loss of expressivity; while at inference, weight decomposition is obtained as a byproduct of matrix multiplication associativity. Theoretical results on the construction of projection matrices with optimal computational accuracy are provided. Experimentally, we find ESPACE enables 50% compression of GPT3, Llama2, and Nemotron4 models with small accuracy degradation, as low as a 0.18 perplexity increase on GPT3-22B. At lower compression rates of 20% to 40%, ESPACE drives GPT3 models to outperforming their baseline, by up to a 0.38 decrease in perplexity for GPT3-8B. ESPACE also reduces GEMM execution time and prefill inference latency on existing hardware. Comparison with related works on compressing Llama2-7B via matrix factorization shows that ESPACE is a first step in advancing the state-of-the-art in tensor decomposition compression of LLMs.

1 Introduction

Capabilities of large language models (LLMs) have recently soared in natural language understanding and generative power. It is appreciated that there exists a correlation between model size and achievable accuracy. Indeed, as LLMs consume trillions of tokens during their training, a large parameter volume is required to capture intricate linguistic features [1]. This leads to a trade-off in LLMs: larger parameter counts improve accuracy but come with increased serving cost.

However, it is also appreciated that the computational requirements of inference may be lower than those of training [2]. To that end, numerous studies have investigated compression of LLMs to reduce inference cost. The most popular LLM compression techniques are quantization [3] and pruning [4]. A less explored, but powerful technique is *tensor decomposition*, and in our work, we propose a novel, *activation-centric* way to decompose LLM tensors. Our proposal is to project activations onto a static set of components optimizing fidelity. The projection reduces activation dimensionality and leads to weight compression at inference as a byproduct of matrix multiplication associativity.

1.1 Related work and motivation for activation-centric tensor decomposition

Recent research has proposed many **quantization** and **pruning** techniques for compressing LLMs. Examples of advances in LLM quantization include SmoothQuant [3], AWQ [5], and GPTQ [6]; while notable LLM pruning works include SparseGPT [4], LLM-Pruner [7], and ReLU-based masking [8]. These methods are conceptually orthogonal to our proposal for activation projection which can be

[1]We use the french pronunciation "espace", which means "space".

implemented in low precision or sparse formats. Nevertheless, compression fundamentally introduces noise, and an open problem is to study the impact of combining different methods, e.g., quantization and matrix factorization. This is beyond the scope of our paper, but a good direction for future work.

Our work is also orthogonal to **non-compressive** LLM serving acceleration such as continuous batching [9] or speculative decoding [10], and attention optimizations such as PagedAttention [11], RadixAttention [12], and FlashAttention [13]. Our study is on matrix multiplication layers involving weights and activations, and hence is mutually exclusive to works improving cross multiplications of activation tensors in attention. In fact, all our experiments use FlashAttention.

Finally, we turn to **tensor decomposition**, also known as **factorization**. Thus far, compression for LLM inference using factorization has been focused on **weight decomposition**. KnGPT [14] uses the Kronecker transform to pack a large matrix into two smaller ones. TSVD [15] performs iterative singular value decomposition (SVD) on weight matrices to produce high rank ternary components. TensorGPT [16] and HEAT [17] compress weight matrices into a cascade product of small matrices using the tensor-train algorithm. SVD-LoRa [18] uses a truncated SVD on weights and finetunes the model using LoRa [19]. The LoRa adapters are then merged to the main branch using bounds on the rank of sum of low rank matrices. ASVD [20] performs a truncated SVD on the weights after re-scaling them by a diagonal matrix and their inverse encapsulating activation statistics. This work realizes the importance of activation-awareness but still uses weight-centric compression. SliceGPT [21] extracts principal components in normalization layers to guide the deletion of rows and columns in weight matrices. The compression is achieved using a factorization made implicit via computational invariance. The statistical method employed by sliceGPT shares similarities with one of our results, but our problem formulation and solution are different.

Factorization can also streamline LLM training and finetuning. For instance, LoRa [19] finetunes pre-trained models using residual low rank adapters which are then absorbed into the main branch. Similarly, GaLore [22] applies a low rank approximation to gradients in back-propagation. These works do not modify inference parameter and operation count, and are hence orthogonal to ours. Our method could be applied in tandem with LoRa or GaLore, but this is beyond the scope of this paper.

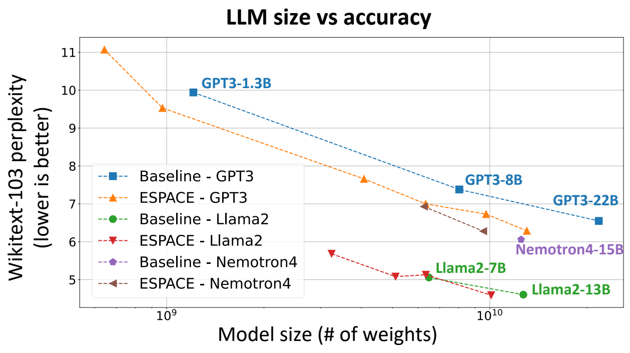

Figure 1: Perplexity[2] versus model size for GPT3 and Llama2 models and comparison to compressed models using ESPACE.

Since factorization increases the number of LLM tensors, achieving high compression rates requires the intermediate dimensions to be much smaller than that of original dot product. This breakage in computation usually necessitates a retraining or finetuning stage to be healed. Unfortunately, this healing process is impeded because factorized LLMs have fewer learnable parameters which decreases expressivity [17, 22].

To our knowledge, no prior art has explored **activation decomposition**. Indeed, applying factorization solvers (e.g., SVD) dynamically incurs large inference runtime overheads. Yet, activation decomposition has several desired features which we examine in Section 2 and motivate via the following insights: (a) weights stay uncompressed during retraining, preventing the aforementioned loss of expressivity; (b) large activation tensors contain inherent redundancies making them prime candidates for compression; and (c) since most LLM computation comprises multiplications of weights and activations, decomposing the latter can lead to compressing the former at inference.

1.2 Contributions

We propose Eigen Static Principal Activation Component Estimation (ESPACE), an LLM compression technique based on activation dimensionality reduction. Our contributions are as follows:

- We project activation tensors onto a static and pre-calibrated orthonormal matrix. The projection lowers activation dimensionality but keeps weight matrices intact and fully available for training.

At inference, leveraging matrix multiplication associativity, model compression is achieved through pre-computation of the product of weight and projection matrices.

- We theoretically derive optimal constructions for activation dimensionality reduction. Specifically, the projection matrix is calibrated in a manner to minimize activation decomposition mean squared error and forward propagated noise metrics. The solution is based on an eigenvalue decomposition of activation auto-correlation and yields multiple candidate projections for each activaton tensor.
- We empirically study compression of models in the GPT3, Llama2, and Nemotron4 families evaluated on the Wikitext-103 dataset for perplexity and the LM evaluation harness for downstream task accuracy. The amelioration in size versus perplexity[2] trade-offs is summarized in Figure 1.
- We show that ESPACE can compress LLMs by $\sim$50% at the cost of a small accuracy loss, as low as 0.18 increase in perplexity on GPT3-22B.
- At lower compression rates, we find encouraging empirical evidence that ESPACE filters out noise and improves accuracy; e.g., $\sim$20% compressed GPT3-8B lowers its baseline perplexity by 0.38.
- As an additional benefit of ESPACE, tangible latency reduction of 35%-to-45% is obtained in matrix multiplication layers. This speed-up translates to up to $\sim$ 40% faster prefill inference latency metricized by the time to first token and measured on existing hardware.
- By comparison to existing works on tensor decomposition, we determine that ESPACE is a first step in pushing the frontier of compression rate versus accuracy retention (see Figure 4).

2 Dimensionality Reduction & Projections

In this section, we introduce notation for matrix multiplication, review weight decomposition, and introduce our proposed mechanism of dimensionality reduction via activation projections.

2.1 Matrix Multiplication and Weight Decomposition

We consider general matrix multiplications (GEMMs) described in Figure 2(a) of the form

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X} \tag{1}$$

where $\mathbf{W}$ is a weight matrix of size $K \times N$ and $\mathbf{X}$ is an input activation tensor of size $K \times M$ so that the output activation tensor $\mathbf{Y}$ is of size $N \times M$. Typically, K and N are defined by network topology and layer instance, they are commonly referred to as *embedding* or *hidden* size. In contrast, M stacks multiple dimensions in an activation tensors to obtain a 2D matrix view. Generally, these are the *sequence* and *batch* dimensions.

Transformer-based LLMs have four GEMM layers per block: query-key-value (QKV), projection (Proj), fully connected 1 (FC1), and fully connected 2 (FC2) layers. Our study is concerned with these layers, while cross activation multiplication and embedding layers are untouched. For notational simplicity, in this paper, *we do not include layer indices in our equations*.

The matrix $\mathbf{W}$ in (1) stores layer parameters and dictates the model's inference accuracy. To improve convergence of these parameters, an optimizer state is stored alongside weights during training and tracks historical values of gradients and updates [23, 24]. On the other hand, the activation tensor $\mathbf{X}$ depends on the input stimulus to the network, and is therefore generated on the fly.

Thus, at inference, weights are fixed but activations are dynamic. As a consequence, prior work on tensor decomposition has focused on compressing frozen weight matrices. One way of doing so is breaking $\mathbf{W}^T$ into a low-rank approximation using some form of truncated SVD [20, 18], which is described in Figure 2(b). Specifically, (1) is approximated as:

$$\mathbf{Y} \approx \mathbf{U}\mathbf{V}\mathbf{X} \tag{2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are matrices of size $N \times L$ and $L \times K$, respectively, with L being the factorization rank. For the decomposition procedure to be useful, two conditions need to be met: (a) $L << \min(K, N)$ for compression, and (b) the approximation $\mathbf{W}^T \approx \mathbf{U}\mathbf{V}$ should be accurate. However, achieving both conditions simultaneously may be challenging because a very low rank factorization usually leads to significant accuracy drop [22].

[2]Perplexity depends on tokenizer so that comparisons across LLM families (GPT3 vs Llama2) are not useful.

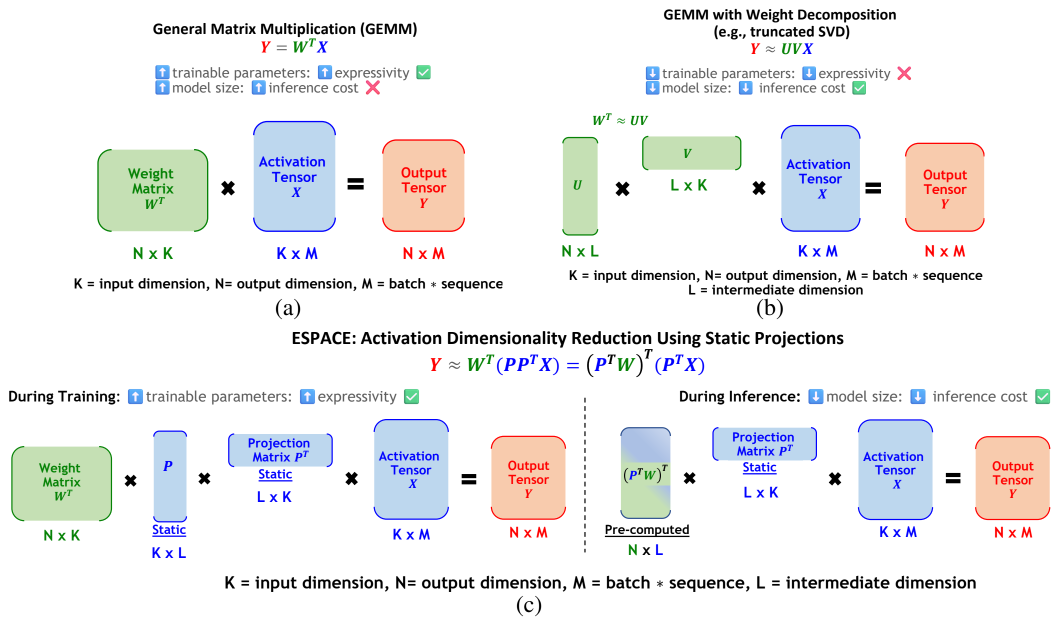
General Matrix Multiplication (GEMM)
Y = W^T X
trainable parameters: ⬆ expressivity ✓
model size: ⬆ inference cost ✗
Weight Matrix W^T
Activation Tensor X
Output Tensor Y
N x K
K x M
N x M
K = input dimension, N= output dimension, M = batch ∗ sequence
(a)
GEMM with Weight Decomposition (e.g., truncated SVD)
Y ≈ UVX
trainable parameters: ⬇ expressivity ✗
model size: ⬇ inference cost ✓
W^T ≈ UV
U
V
N x L
L x K
K x M
N x M
K = input dimension, N= output dimension, M = batch ∗ sequence
L = intermediate dimension
(b)
ESPACE: Activation Dimensionality Reduction Using Static Projections
Y ≈ W^T(PP^T X) = (P^T W)^T (P^T X)
During Training: ⬆ trainable parameters: ⬆ expressivity ✓
During Inference: ⬇ model size: ⬇ inference cost ✓
P
Static
K x L
Projection Matrix P^T
Static
L x K
(P^T W)^T
Pre-computed
N x L
K = input dimension, N= output dimension, M = batch ∗ sequence, L = intermediate dimension
(c)

Figure 2: Decompositions in GEMMs: (a) baseline multiplication of weight matrix and activation tensor, (b) truncated SVD on the weight matrix, and (c) proposed approach of inserting a static matrix to project activations. With ESPACE, all weights are available for training, while inference compression is achieved via per-computation of $\left(\mathbf{P}^T\mathbf{W}\right)$.

As with other compression techniques, e.g., quantization and pruning, retraining of the compressed model may be employed to recover accuracy. However, the decomposition in (2) introduces two training-related hurdles: (a) the effective number of trainable parameters has decreased significantly which reduces model expressivity, and (b) the breakage of spatial weight structure prevents the retraining procedure from loading the original optimizer state. Retraining a model without its optimizer state is known to introduce significant difficulty in convergence [17].

2.2 Activation Decomposition via Static Projection

Since weight decomposition poses the above hurdles, we motivate the need for an activation-centric solution. In some measure, activation compression may be more achievable due to the large stack dimension M comprising batches and sequences. Statistically, the Central Limit Theorem claims that stacking data is likely to exhibit redundancies [25]. In the case of LLMs, such redundancies are further pronounced due to the likelihood of repeated tokens and information in natural language.

Therefore, activations should be prime candidates for tensor decomposition. Nevertheless, prior arts have not explored activation decomposition due to one fundamental limitation: unlike weights, activations are generated on the fly; meaning that tensors must be compressed during inference, potentially incurring large runtime penalties.

We propose to apply *static* dimensionality reduction on the activation tensor $\mathbf{X}$ in (1). Concretely, our proposal is to project $\mathbf{X}$ onto a pre-computed static orthonormal matrix $\mathbf{P}$ of size $K \times L$, where crucially $L << K$. Reconstructing $\mathbf{X}$ requires a re-expansion using the transpose of the projection matrix, i.e., $\mathbf{X} \approx \mathbf{P}\mathbf{P}^T\mathbf{X}$. While $\mathbf{P}^T\mathbf{P} = \mathbf{I}_{L\times L}$, we note that $\mathbf{P}\mathbf{P}^T \neq \mathbf{I}_{K\times K}$ since $L << K$. Thus, the proposed activation transformation is noisy, and in Section 3, we derive optimal conditions on the calibration of $\mathbf{P}$ to minimize the effects of this noise.

Our proposal, described in Figure 2(c) is to approximate the GEMM in (1) using the following:

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X} \approx \mathbf{W}^T\mathbf{P}\mathbf{P}^T\mathbf{X} = \mathbf{W}^T\left(\mathbf{P}\mathbf{P}^T\mathbf{X}\right) = \left(\mathbf{P}^T\mathbf{W}\right)^T\left(\mathbf{P}^T\mathbf{X}\right) \tag{3}$$

where we used associativity of matrix multiplication to highlight key aspects of our approach.

During training/finetuning: we view our GEMM as $\mathbf{W}^T\left(\mathbf{P}\mathbf{P}^T\mathbf{X}\right)$ where $\mathbf{X}$ has been replaced by its approximation. We emphasize that $\mathbf{P}$ is static and does not get updated during training.

Meanwhile, $\mathbf{W}^T$ is fully available for adaptation to the activation approximation. The availability of all learnable weights elides losing model expressivity. The structure of $\mathbf{W}^T$ is also unchanged and can be mapped to the baseline's optimizer state. Thus, the proposed approach does not suffer from the same limitations as weight decomposition techniques. We do note that introducing $\mathbf{P}$ induces a small storage overhead at train time. However, when $L << K$, and the order of computation is properly compiled, the number of operations per iteration is lower than baseline training; and, though not central to our contribution, we did observe up to 15% reduction in training iteration time for 50% compressed models.

During inference: we view our GEMM as $\left(\mathbf{P}^T\mathbf{W}\right)^T\left(\mathbf{P}^T\mathbf{X}\right)$ where the required matrices are $\mathbf{P}$ of size $K \times L$ and $\left(\mathbf{P}^T\mathbf{W}\right)$ of size $N \times L$, which is pre-computed before deployment. Thus, per-layer parameter count required for inference has decreased from KN to $L(K+N)$, which, provided $L << \{K, N\}$ presents an opportunity for significant model compression. For instance, if $N = K$, i.e., $\mathbf{W}^T$ is square, and $L = K/4$, then our method yields 50% compression at inference time. This is one of the compression rates we target in Section 4.

We emphasize that $\mathbf{P}$ is not shared across GEMM layers; rather, each GEMM layer decomposed according to (3) has its own pre-calibrated matrix $\mathbf{P}$. Furthermore, by virtue of (3) not introducing dependencies across mini-batches, our method is fully compatible with data parallelism.

3 Eigen Static Principal Activation Component Estimation

Our proposed activation decomposition induces an approximation error as $\mathbf{X} \neq \mathbf{P}\mathbf{P}^T\mathbf{X}$. In this section, we first introduce an ergodic estimation of activation auto-correlation. This important statistic is then used for theoretical constructions of $\mathbf{P}$ with guarantees on computational accuracy. Multiple results are presented and then combined in our compression studies in Section 4.

3.1 Activation auto-correlation estimation

Let $\mathbf{x}$ be an arbitrary K-dimensional vector in $\mathbf{X}$; we define the *activation auto-correlation* matrix of size $K \times K$ as $\mathbf{C}_\mathbf{X} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]$ where expectation is taken over activation vectors. This matrix is symmetric positive semi-definite having a *real* eigenvalue decomposition (EVD) $\mathbf{C}_\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ where $\mathbf{V}$ is an orthonormal matrix whose columns are eigenvectors, and $\mathbf{D}$ is a diagonal matrix containing the corresponding *non-negative* eigenvalues, assumed to be sorted in decreasing order. The eigenvector corresponding to the i^{th} largest eigenvalue is called i^{th} *principal* eigenvector.

This autocorrelation matrix can be empirically estimated using an instance of the activation tensor:

$$\mathbf{X} = [\mathbf{x}_1|\ldots|\mathbf{x}_\mathbf{M}] \Rightarrow \mathbf{X}\mathbf{X}^T = \left[\mathbf{x}_1\mathbf{x}_1^T + \ldots + \mathbf{x}_M\mathbf{x}_M^T\right] \Rightarrow \mathbf{C}_\mathbf{X} = \mathbf{X}\mathbf{X}^T/M \tag{4}$$

However, evaluating (4) and its EVD dynamically introduces a prohibitive computational overhead. Thus, we estimate $\mathbf{C}_\mathbf{X}$ in a pre-deployment calibration process. Specifically, during calibration, we sample and forward pass B random input batches, and for each, calculate $\mathbf{C}_\mathbf{X}^{(i)} = \mathbf{X}^{(i)}\mathbf{X}^{(i)T}/M$, where superscript i denotes batch index. Then, we average our estimate of the auto-correlation matrix as $\mathbf{C}_\mathbf{X} = \sum_{i=1}^{B}\mathbf{C}_\mathbf{X}^{(i)}/B$ and use its eigenvalue decomposition for further optimizations.

This ergodic approach of estimating activation statistics as part of a calibration process has been employed to great effect in other compression works on quantization [26, 27] and pruning [28].

3.2 Activation decomposition with minimum mean squared error

Let us write $\tilde{\mathbf{X}} = \mathbf{P}\mathbf{P}^T\mathbf{X}$; for a vector $\mathbf{x} \in \mathbf{X}$, its counterpart in $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ is given by:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{L}\langle\mathbf{p}_i, \mathbf{x}\rangle\mathbf{p}_i \tag{5}$$

where $\{\mathbf{p}_i\}_{i=1}^{L}$ are the orthonormal column vectors of $\mathbf{P}$, i.e., $\langle\mathbf{p}_i, \mathbf{p}_j\rangle = \mathbb{1}_{\{i==j\}}, \forall i, j \in 1 \ldots L$.

We define the mean squared error (MSE) of the decomposition as

$$\mathbb{E}\left[\|\mathbf{x} - \tilde{\mathbf{x}}\|^2\right] \tag{6}$$

with the L_2-norm used throughout this paper. Our first result constructs $\mathbf{P}$ minimizing this MSE.

Theorem 1. *For an activation tensor* $\mathbf{X}$ *whose auto-correlation matrix has an eigenvalue decomposition given by* $\mathbf{C_X} = \mathbf{V}\mathbf{D}\mathbf{V}^T$*, the projection matrix* $\mathbf{P}$ *minimizing the mean squared error in* (6) *is given by* $\mathbf{P} = [\mathbf{v}_1| \dots |\mathbf{v}_L]$ *where* $\mathbf{v}_i$ *is the* i^{th} *principal eigenvector in* $\mathbf{V}$.

Proof. See Appendix A.1. The result is readily obtained by substituting $\tilde{\mathbf{x}}$ in (5) into (6) and minimizing the MSE which involves quadratic forms involving the positive semi-definite $\mathbf{C_X}$. □

Theorem 1 shares similarities with the Principal Component Analysis (PCA) algorithm [29]. PCA extracts low dimensional features having maximum correlation with input data. Unlike PCA, we omit input normalization to elide its computational cost. Still, we term the columns of $\mathbf{P}$ in Theorem 1 as Principal Activation Components. Since those are obtained using an EVD on a static estimation of $\mathbf{C_X}$, we call our method Eigen Static Principal Activation Component Estimation (ESPACE).

The MSE in Theorem 1 is a strong indicator of the quality of an approximation technique, e.g., it is often employed in quantization studies [26, 27]. However, empirical data may contain large outliers which can dominate the optimization process; say a few high-magnitude vectors in (6) masking the contribution of small data on the solution. An alternate metric to the MSE can be employed to prevent such artifacts in averaging: the normalized MSE (NMSE) defined as:

$$\mathbb{E}\left[\|\mathbf{x}-\tilde{\mathbf{x}}\|^2 / \|\mathbf{x}\|^2\right] \tag{7}$$

The solution of Theorem 1 can be slightly modified to minimize the NMSE in (7).

Corollary 2. *For an activation tensor* $\mathbf{X}$*, let* $\hat{\mathbf{C}}_\mathbf{X} = \mathbb{E}\left[\left(\mathbf{x}/\|\mathbf{x}\|\right)\left(\mathbf{x}/\|\mathbf{x}\|\right)^T\right]$ *be its input-normalized auto-correlation matrix having an eigenvalue decomposition given by* $\hat{\mathbf{C}}_\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^T$*, the projection matrix* $\mathbf{P}$ *minimizing the normalized mean squared error in* (7) *is given by* $\mathbf{P} = [\mathbf{v}_1| \dots |\mathbf{v}_L]$ *where* $\mathbf{v}_i$ *is the* i^{th} *principal eigenvector in* $\mathbf{V}$.

Proof. The proof in Appendix A.2 uses equivalence of NMSE and MSE with L_2-normalized vectors. □

Corollary 2 applies to the decomposition in (3) with no activation normalization required at compute time. Rather, normalization is done during calibration, where $\hat{\mathbf{C}}_\mathbf{X}$ is estimated instead of $\mathbf{C_X}$.

Both solutions in Theorem 1 and Corollary 2 are options to be employed in ESPACE, where either may be more suitable on a layer-wise basis. Next we present further options for ESPACE based on the optimization of alternate metrics to the MSE and NMSE.

3.3 Activation decomposition with optimized forward propagated accuracy metrics

While local fidelity metrics, such as the MSE and NMSE above, are good indicators of the quality of an approximation technique, it has been shown that better insights on a neural network's accuracy may be derived via the study of forward propagated noise [30, 31, 32]. In this section, we study the effects of the decomposition in (3) on the output of the GEMM, and the output loss of the model.

At a given layer, let us write an arbitrary scalar in the GEMM output tensor in (1) as $y \in \mathbf{Y}$. Note that $y = \langle \mathbf{w}, \mathbf{x} \rangle$ for some weight vector in $\mathbf{w} \in \mathbf{W}$ and activation vector $\mathbf{x} \in \mathbf{X}$. We also let $\tilde{y}$ be the associated output when the GEMM is approximated by (3), which is given by $\tilde{y} = \langle \mathbf{w}, \tilde{\mathbf{x}} \rangle$ with $\tilde{\mathbf{x}}$ given by (5). We define the GEMM Output-referred MSE (GO-MSE) as $\mathbb{E}\left[(y - \tilde{y})^2\right]$.

Similarly, given an input to the network, we write the output loss function (the vocab cross-entropy) as $\mathcal{L}$. When one arbitrary activation tensor is transformed as per (3), a mismatch in computation is introduced and propagated all the way to the output. We let $\tilde{\mathcal{L}}$ be the resulting new value of the loss function. We define the Network Loss-referred MSE (NL-MSE) as $\mathbb{E}\left[(\mathcal{L} - \tilde{\mathcal{L}})^2\right]$.

A closed form solution for $\mathbf{P}$ in (3) minimizing the GO-MSE and NL-MSE is elusive to us. Therefore, we derive upper bounds on these metrics which we use as a proxy for optimization.

Proposition 3. *For a GEMM in* (1) *and its decomposition in* (3)*, the GO-MSE is upper bounded by:*

$$\mathbb{E}\left[(y - \tilde{y})^2\right] \leq 2\mathbb{E}\left[\|\mathbf{w}\|^2 \cdot \|\mathbf{x}\|^2\right] - 2\mathbb{E}\left[\langle \mathbf{w}, \mathbf{x} \rangle \cdot \langle \mathbf{w}, \tilde{\mathbf{x}} \rangle\right] \tag{8}$$

and the NL-MSE is upper bounded by

$$\mathbb{E}\left[(\mathcal{L}-\tilde{\mathcal{L}})^2\right] \leq 2\mathbb{E}\left[\|\nabla_{\mathbf{x}}\|^2 \cdot \|\mathbf{x}\|^2\right] - 2\mathbb{E}\left[\langle\nabla_{\mathbf{x}},\mathbf{x}\rangle \cdot \langle\nabla_{\mathbf{x}},\tilde{\mathbf{x}}\rangle\right] \tag{9}$$

where a first order Taylor approximation on the loss function is assumed and its gradient with respect to vector $\mathbf{x}$ *is denoted as* $\nabla_{\mathbf{x}}$.

Proof. The proof in Appendix A.3 first shows $\|\tilde{\mathbf{x}}\|^2 < \|\mathbf{x}\|^2$ and then uses the Cauchy Schwarz inequality to establish both bounds. □

Next, we provide closed form solutions for $\mathbf{P}$ in (3) minimizing the bounds in Proposition 3.

Theorem 4. *For a GEMM in* (1) *and its decomposition in* (3)*, the projection matrix minimizing the bounds in Proposition 3 is given by* $\mathbf{P} = [\mathbf{v}_1|\ldots|\mathbf{v}_L]$ *where* $\mathbf{v}_i$ *is the* i^{th} *principal eigenvector in* $\mathbf{V}$ *obtained via eigenvalue decomposition on a matrix* $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ *defined as:*

$$\mathbf{C} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right] \quad \textit{and} \quad \mathbf{C} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^T + \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^T\mathbf{x}\mathbf{x}^T\right] \tag{10}$$

to minimize the upper bounds on GO-MSE in (8) *and NL-MSE in* (9)*, respectively.*

Proof. The proof is included in Appendix A.4, where we also include modifications required in calibration. Specifically, $\mathbf{C}_{\mathbf{X}}$ is reused and left/right multiplied by $\mathbf{w}\mathbf{w}^T/N$ to yield $\mathbf{C}$ in (10) minimizing the bound on GO-MSE. An additional backward pass is needed to properly scale activation vectors and their gradients when calibrating $\mathbf{C}$ in (10) minimizing the bound on NL-MSE. □

Theorem 4 augments Theorem 1 and Corollary 2 with two options for the design of $\mathbf{P}$. Much like Corollary 2, we supplement our new solutions with L_2-normalization to include

$$\hat{\mathbf{C}} = \mathbb{E}\left[\left(\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right)/\|\mathbf{w}\|^2\cdot\|\mathbf{x}\|^2\right] \quad \text{and} \quad \hat{\mathbf{C}} = \mathbb{E}\left[\left(\mathbf{x}\mathbf{x}^T\nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^T + \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^T\mathbf{x}\mathbf{x}^T\right)/\|\nabla_{\mathbf{x}}\|^2\cdot\|\mathbf{x}\|^2\right]$$

as alternate choices for the calibrated matrices $\mathbf{C}$ in (10). Unlike Corollary 2, L_2-normalization in these two matrices does not correspond to a notable optimization. Nevertheless, these options are retained in the spirit of suppressing the influence of large data in calibration.

Thus, overall we have six choices for $\mathbf{P}$. Since each can be obtained as part of a fast and pre-deployment calibration phase, we may simply select the best one for each layer. In our experiments of Section 4, the best candidate is determined via a per-layer validation over all six choices. A sensitivity study on the impact of each of the six candidates is provided in Appendix B.4.

4 Model Compression Studies

In this section, we report on experimental studies investigating LLM compression using ESPACE.

4.1 Experimental setup

We employ three sets of open source LLMs: GPT3 [33], Llama2 [34], and Nemotron4 [35]. Specifically, we experiment on GPT3-{1.3B, 8B, 22B}, Llama2-{7B, 13B}, and Nemotron4-15B. Accuracy is evaluated in two ways: perplexity measured on the Wikitext-103 dataset [36] and zero-shot downstream task accuracy of: BoolQ (BQ) [37], Hellaswag (HS) [38], PIQA (PQ) [39], RACE (RA) [40], and WinoGrande (WG) [41].

The Wikitext-103 dataset is split into train, validation, and test sets. We use 512 random sequences from the training set for calibrating projection matrices required by ESPACE. We use the validation set for layer-wise sensitivity studies. The test set is used to report perplexity results in this section.

Our implementation uses NVIDIA's Megatron LM [33] and downstream task evaluation invokes Eleuther AI's LM evaluation harness [42]. For the latter, we report raw accuracy scores, and their average; we do not post process results or apply normalization to the scores.

When ESPACE is applied, we retrain the models to adapt to the approximation error of activation projection as discussed in Section 2. Retraining simply extends the models' pre-training sessions and uses the 330B-token MTNLG dataset [43], which was used to train GPT3 models. All implementation details are included in Appendix B to help reproducibility of our results.

We metricize model size reduction via inference compression rate. Specifically, for layers decomposed per (3), we count the number of entries in $\mathbf{P}$ and $\left(\mathbf{P}^T\mathbf{W}\right)$; for other layers, we count those in $\mathbf{W}^T$. We also report the latency of executing all network GEMMs in (1) or (3), which we measure using a NVIDIA A100 GPU and a simple, un-optimized implementation (see Appendix B.4). We also report prefill inference latency, metricized via the Time to First Token (TTFT), and measured using the Megatron-LM implementation. In our measurements, we use a batch size of 1 and sequence length of 2048 and 4096 for GPT3 and Llama2/Nemotron4 models, respectively. The reported reductions in total GEMM latency and TTFT constitute evidence that compression improves inference throughput. It is beyond the scope of this paper to evaluate the impact of ESPACE on end-to-end token throughput and latency on LLM inference serving systems, since this requires a complex set of optimizations including but not limited to optimization of back-to-back GEMMs into fused kernels, KV caching, continuous batching, as well as thorough performance studies with varying input and output sequence lengths. Thus, we leave an evaluation of token generation throughput and energy savings and improvements to future work.

4.2 Validation perplexity studies

Our experiments start with a calibration phase where we prepare the static projection matrix $\mathbf{P}$ for each layer. The dimension L in $\mathbf{P}$ is chosen as the lowest power of two such that layer compression is at least 50%. The power-of-two restriction ensures best tensor core utilization, and the resulting compression rate depends on the dimensions of the original layer (K and N). Exact details of these values for all layers and models are included in Appendix B.2.

We perform a sensitivity study on the Wikitext-103 validation perplexity when ESPACE is applied out-of-the-box (no retraining) one layer at a time. For each layer, we identify which of our six candidates projection matrices in Section 3 yields lowest validation perplexity. Layers are then sorted according to their impact on perplexity from least to most destructive. We then evaluate the validation perplexity when ESPACE is progressively applied to out-of-the-box to all layers according to this ranking. Fine-grained details of this exploration are included in Appendix C for all models.

This exploration yields an interesting finding: as we progressively apply ESPACE to more layers, the perplexity marginally increases until an inflection point after which accuracy degradation accelerates. This inflection occurs at 20% to 40% compression depending on the model. Figure 3 depicts this phenomenon for GPT3-22B, and the same data for other models can be found in Appendix C.2.

We find that out-of-the-box application of ESPACE works better for larger models; GPT3-22B, the largest model we experimented on, exhibits an inflection in perplexity at 40% compression, which is the highest in our results. This is consistent with many earlier works on general compression of neural networks [44, 45, 46, 47]. Interestingly, a 20% out-of-the-box compressed GPT3-22B is iso-accurate to its uncompressed counterpart (see Figure 3); without retraining, its test perplexity of **6.61** which is within 1% of the 6.55 baseline.

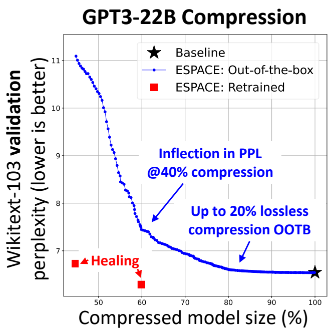

Figure 3: Validation perplexity for GPT3-22B when ESPACE is progressively applied to its GEMM layers. The order of layer selection is based on a layer-wise sensitivity analysis.

After the above validation study is performed, we select two configurations for layers to be compressed using ESPACE: (a) layers corresponding to the inflection point, i.e., 20% to 40% compression, and (b) as many layers needed to achieve a compression of ~50%. For both configurations, we retrain the compressed models and further evaluate their achievable accuracy.

4.3 Compression of GPT3 models

Once compression targets and layer configurations are set, we retrain GPT3 models on the MTNLG dataset. Although we use all of the 330B available tokens, we do observe the training loss quickly

Table 1: GEMM latency, time to first token, Wikitext-103 perplexity (WK-103 PPL), and downstream task accuracy of GPT3, Llama2, and Nemotron4 models compressed with ESPACE[3].

Method (Compression)	# of Weights	Total GEMM Latency	TTFT impl. in Megatron-LM	WK-103 PPL ↓	Downstream Task Accuracy ↑					
					BQ	HS	PQ	RA	WG	Avg.
GPT3-1.3B										
Baseline	1.21×10^9	24.2ms	39.8ms	9.94	**64.3**	43.5	**74.2**	**37.6**	58.1	55.5
ESPACE (20%)	9.71×10^8	20.6ms (-15%)	36.1ms (-9%)	**9.53**	60.6	**45.1**	*73.0*	*36.4*	**62.9**	**55.6**
ESPACE (47%)	6.42×10^8	15.9ms (-34%)	31.7ms (-20%)	11.07	*62.3*	39.9	*71.6*	34.5	*58.7*	*53.4*
GPT3-8B										
Baseline	8.05×10^9	136ms	186ms	7.38	69.0	54.2	78.1	**41.4**	67.8	62.1
ESPACE (21%)	6.33×10^9	110ms (-19%)	155ms (-16%)	**7.00**	**70.3**	**55.3**	**78.9**	*40.7*	**69.3**	**62.9**
ESPACE (50%)	4.08×10^9	76.8ms (-44%)	122ms (-35%)	*7.66*	*66.5*	*52.3*	*77.6*	38.9	*66.9*	*60.4*
GPT3-22B										
Baseline	2.17×10^{10}	354ms	457ms	6.55	76.4	57.2	79.3	**40.7**	**70.5**	**64.8**
ESPACE (40%)	1.30×10^{10}	229ms (-35%)	313ms (-32%)	**6.29**	**76.6**	**57.3**	**79.5**	*40.2*	*70.2*	**64.8**
ESPACE (55%)	9.74×10^9	181ms (-49%)	261ms(-43%)	*6.73*	*72.2*	*55.8*	*79.3*	*40.1*	*69.7*	*63.4*
Llama2-7B										
Baseline	6.48×10^9	210ms	368ms	**5.06**	**79.2**	57.1	78.1	**44.0**	69.5	65.6
Retrained (0%)	6.48×10^9	210ms	368ms	**5.06**	78.2	**57.9**	78.0	43.7	**70.6**	**65.7**
ESPACE (21%)	5.11×10^9	169ms (-19%)	322ms (-12%)	*5.07*	*77.1*	*57.1*	**78.7**	*42.7*	*69.2*	*65.0*
ESPACE (50%)	3.24×10^9	113ms (-46%)	266ms (-28%)	5.67	72.2	52.0	*76.5*	38	63.5	60.4
Llama2-13B										
Baseline	1.27×10^{10}	406ms	643ms	4.61	**82.4**	60.2	**79.5**	**46.8**	71.9	**68.2**
ESPACE (20%)	1.01×10^{10}	336ms (-17%)	562ms (-13%)	**4.59**	*78.3*	**60.5**	**79.5**	43.0	**72.8**	*66.8*
ESPACE (50%)	6.34×10^9	259ms (-36%)	447ms (-31%)	5.13	75.7	56.2	*78.0*	41.5	*69.1*	64.1
Nemotron4-15B										
Baseline	1.25×10^{10}	414ms	741ms	**6.06**	78.3	**62.1**	**81.1**	**47.0**	**75.2**	**68.7**
ESPACE (25%)	9.54×10^9	324ms (-22%)	655ms (-12%)	*6.28*	**78.9**	*59.9*	*80.0*	*46.4*	*72.8*	*67.6*
ESPACE (50%)	6.25×10^9	223ms (-46%)	545ms (-26%)	6.93	*77.9*	57.0	*77.8*	42.4	69.9	65.0

converging. We leave training hyperparameters unchanged except for one: we disable dropout. Our rationale is that activation projection is one form of deterministic and structured dropout such that additional regularization may not be needed. Results[3] on GPT3 models are included in Table 1.

We find that ESPACE can compress GPT3 models by $\sim$50% at the cost of a small accuracy degradation. In the case of GPT3-22B, the perplexity increase is of only 0.18; in general, the gap decreases for larger overall model size. By and large, similar trends are observed for downstream task accuracies and we note that most scores of 50% compressed models fall within 5% of the baseline.

For lower compression ratios (inflection points at 20% to 40%), ESPACE converges to an accuracy *better than that of the baseline*. The best improvement occurs for GPT3-8B, where ESPACE produces a 6B model with 0.38 lower perplexity than its 8B baseline. The improvements are observed both in terms of perplexity and downstream task accuracy. While GPT3 models may be over-parameterized, we posit that ESPACE acts a regularizer at moderate compression rates. Specifically, we believe that projection onto principal activation components filters out unnecessary information coming from small eigenvalue components.

For all models, we observe an encouraging translation of compression to GEMM latency reduction by up to 49% which leads to noticeable speed-up in TTFT by up to 43%..

4.4 Compression of Llama2 models and comparison to related works

For Llama2, we only retrain using 200B MTNLG tokens because we observed quick convergence for GPT3. Llama2 models were trained on an undisclosed dataset of 2T tokens [34]. Therefore, with 200B tokens, the healing phase of ESPACE constitutes no more than 10% of the original pre-training session. Since Llama2 pre-training details are not openly available, we re-used all hyperparameters from GPT3, which is likely to be sub-optimal. In spite of the two handicaps of dataset disparity and hyperparameter sub-optimality, we obtained promising results as reported in Table 1.

For Llama2-7B, we first retrained the uncompressed baseline. The purpose of this experiment is twofold: (a) ensure that our hyperparameters at least do not corrupt the model, and (b) verify that the

[3]**Boldface** indicates best result per task. *Italics* indicates ESPACE results within 5% of the baseline

healing process is not just an artifact of processing more tokens. Both hypotheses appeared to be valid: the retrained baseline has nearly identical accuracy compared to the original model.

Generally, we find that the trends of ESPACE compression for Llama2 are similar to GPT3, albeit slightly less successful. Though the results are still promising, we attribute the slight shortcomings in accuracy to the handicaps above. We find that 50% ESPACE compression on Llama2 leads to $\sim$0.6 perplexity increase and similar degradation in terms of downstream task accuracy. Notably, compressing the Llama2-13B model to to a 6.3B model yields comparable accuracy to the Llama2-7B baseline which itself is a 6.5B model.

In addition, for 20% compression, we find that ESPACE matches the accuracy of the baseline for Llama2 models. While not as impressive as the improvements observed with GPT3, ESPACE is able to produce 5B and 10B models matching the 7B and 13B baselines, which does push the pareto frontier of accuracy versus model size in the right direction as shown in Figure 1.

The Llama2-7B model has been used in related works on tensor decomposition mentioned in Section 1.1; specifically, ASVD [20], SVD-LoRa [18], and SliceGPT [21]. Both ASVD and sliceGPT have reported perplexity on Wikitext, but SVD-LoRa performed task-specific finetuning on a variety of datasets and averaged perplexities. Therefore, in Figure 4, we compare our results to these works using perplexity increase over baseline, rather than raw perplexity, for maximum inclusivity.

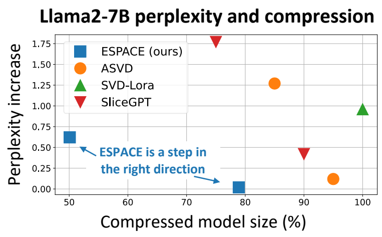

Figure 4: Comparison to related works compressing Llama2-7B using matrix factorization techniques.

SVD-LoRa performed an SVD decomposition on the weights such that the intermediate dimension is half of dot-product which leads to no compression. On the other hand, ASVD and sliceGPT can only achieve modest compression ratios of up to 25% with some loss in accuracy. Recall that these works apply factorization on weights which is the fundamental difference to ESPACE. As seen in Figure 4, ESPACE is a step in the right direction towards improving the state-of-the-art in tensor decomposition of LLMs.

4.5 Compression of Nemotron4-15B

Finally, we used ESPACE to compress Nemotron4-15B into 9.54 and 6.25 billion parameters, as reported in Table 1. Retraining consumed 275B tokens which corresponds to $\sim$ 3% of this model's original training session. Once more, compression with ESPACE leads to minimal degradation in the moderate regime (25%) and yields a small accuracy drop in the aggressive regime (50%).

Consistently with our findings for the above models, ESPACE reduces GEMM execution time by up to 46%. This, in turn, improves the TTFT by up to 26%. An interesting observation is that, for Llama2 and Nemotron4 models, the TTFT improvement is slightly less pronounced than for GPT3 models. This is simply due to the fact that the latter uses a sequence length of 2048, whereas the former two use 4096. A larger sequence length means more time is spent in attention cross-activation products which amortizes the speed-up in the GEMM layers.

5 Conclusion

We have presented ESPACE, a novel compression technique realizing tensor decomposition of LLMs in an activation-centric manner. A set of theoretical results were derived to guide the construction of activation projection which is done statically. Experimentally, we have shown promising results where ESPACE is able to $\sim$50% compress modern LLMs at the cost of a small accuracy degradation. Compared to related works, ESPACE is a first step in pushing the frontier of model size versus accuracy trade-offs. Future work includes combining ESPACE with alternate compression techniques such as quantization and pruning, and evaluating decomposition of activation tensors in attention. As potential extension to our algorithm, the use of matrix sketching and random projections may pave the way for better overall compressibility.

References

[1] F. Xue, Y. Fu, W. Zhou, Z. Zheng, and Y. You, “To repeat or not to repeat: Insights from scaling llm under token-crisis,” *Advances in Neural Information Processing Systems*, vol. 37, 2023.

[2] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong, *et al.*, “Understanding llms: A comprehensive overview from training to inference,” *arXiv preprint arXiv:2401.02038*, 2024.

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

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

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

[6] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, “Gptq: Accurate post-training quantization for generative pre-trained transformers,” *arXiv preprint arXiv:2210.17323*, 2022.

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

[8] I. Mirzadeh, K. Alizadeh, S. Mehta, C. C. Del Mundo, O. Tuzel, G. Samei, M. Rastegari, and M. Farajtabar, “Relu strikes back: Exploiting activation sparsity in large language models,” *arXiv preprint arXiv:2310.04564*, 2023.

[9] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, “Orca: A distributed serving system for {Transformer-Based} generative models,” in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, 2022.

[10] S. Kim, K. Mangalam, S. Moon, J. Malik, M. W. Mahoney, A. Gholami, and K. Keutzer, “Speculative decoding with big little decoder,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[11] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with pagedattention,” in *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

[12] L. Zheng, L. Yin, Z. Xie, J. Huang, C. Sun, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, *et al.*, “Efficiently programming large language models using sglang,” *arXiv preprint arXiv:2312.07104*, 2023.

[13] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, “Flashattention: Fast and memory-efficient exact attention with io-awareness,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 16344–16359, 2022.

[14] A. Edalati, M. Tahaei, A. Rashid, V. P. Nia, J. J. Clark, and M. Rezagholizadeh, “Kronecker decomposition for gpt compression,” *arXiv preprint arXiv:2110.08152*, 2021.

[15] B. Chen, H. Chen, J. He, F. Sun, and S. Jui, “Ternary singular value decomposition as a better parameterized form in linear mapping,” *arXiv preprint arXiv:2308.07641*, 2023.

[16] M. Xu, Y. L. Xu, and D. P. Mandic, “Tensorgpt: Efficient compression of the embedding layer in llms based on the tensor-train decomposition,” *arXiv preprint arXiv:2307.00526*, 2023.

[17] J. Gu, B. Keller, J. Kossaifi, A. Anandkumar, B. Khailany, and D. Z. Pan, “Heat: Hardware-efficient automatic tensor decomposition for transformer compression,” *arXiv preprint arXiv:2211.16749*, 2022.

[18] H. Badri and A. Shaji, “Low-rank pruning of llama2.” `https://mobiusml.github.io/low-rank-llama2/`, 2024.

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

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

[21] S. Ashkboos, M. L. Croci, M. G. d. Nascimento, T. Hoefler, and J. Hensman, “Slicegpt: Compress large language models by deleting rows and columns,” *arXiv preprint arXiv:2401.15024*, 2024.

[22] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian, “Galore: Memory-efficient llm training by gradient low-rank projection,” *arXiv preprint arXiv:2403.03507*, 2024.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[24] G. Hinton, N. Srivastava, and K. Swersky, “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent,” *Cited on*, vol. 14, no. 8, p. 2, 2012.

[25] O. Johnson, *Information theory and the central limit theorem.* World Scientific, 2004.

[26] C. Sakr, S. Dai, R. Venkatesan, B. Zimmer, W. Dally, and B. Khailany, “Optimal clipping and magnitude-aware differentiation for improved quantization-aware training,” in *International Conference on Machine Learning*, pp. 19123–19138, PMLR, 2022.

[27] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, “Integer quantization for deep learning inference: Principles and empirical evaluation,” *arXiv preprint arXiv:2004.09602*, 2020.

[28] S. Dai, H. Genc, R. Venkatesan, and B. Khailany, “Efficient transformer inference with statically structured sparse attention,” in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2023.

[29] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[30] C. Sakr, Y. Kim, and N. Shanbhag, “Analytical guarantees on numerical precision of deep neural networks,” in *International Conference on Machine Learning*, pp. 3007–3016, PMLR, 2017.

[31] C. Sakr and N. Shanbhag, “An analytical method to determine minimum per-layer precision of deep neural networks,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1090–1094, IEEE, 2018.

[32] C. Sakr and N. Shanbhag, “Per-tensor fixed-point quantization of the back-propagation algorithm,” in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[33] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, “Megatron-lm: Training multi-billion parameter language models using model parallelism,” 2020.

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

[35] J. Parmar, S. Prabhumoye, J. Jennings, M. Patwary, S. Subramanian, D. Su, C. Zhu, D. Narayanan, A. Jhunjhunwala, A. Dattagupta, *et al.*, “Nemotron-4 15b technical report,” *arXiv preprint arXiv:2402.16819*, 2024.

[36] S. Merity, C. Xiong, J. Bradbury, and R. Socher, “Pointer sentinel mixture models,” 2016.

[37] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, “Boolq: Exploring the surprising difficulty of natural yes/no questions,” in *NAACL*, 2019.

[38] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, “Hellaswag: Can a machine really finish your sentence?,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[39] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, “Piqa: Reasoning about physical commonsense in natural language,” in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[40] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “RACE: Large-scale ReAding comprehension dataset from examinations,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 785–794, Association for Computational Linguistics, Sept. 2017.

[41] S. Keisuke, L. B. Ronan, B. Chandra, and C. Yejin, “Winogrande: An adversarial winograd schema challenge at scale,” 2019.

[42] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," 12 2023.

[43] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, *et al.*, "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model," *arXiv preprint arXiv:2201.11990*, 2022.

[44] E. Park and S. Yoo, "Profit: A novel training method for sub-4-bit mobilenet models," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 430–446, Springer, 2020.

[45] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, "Overcoming oscillations in quantization-aware training," in *International Conference on Machine Learning*, pp. 16318–16330, PMLR, 2022.

[46] A. Kuzmin, M. Nagel, M. Van Baalen, A. Behboodi, and T. Blankevoort, "Pruning vs quantization: Which is better?," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[47] L. Dery, S. Kolawole, J.-F. Kagey, V. Smith, G. Neubig, and A. Talwalkar, "Everybody prune now: Structured pruning of llms with only forward passes," *arXiv preprint arXiv:2402.05406*, 2024.

[48] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[49] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei, "The era of 1-bit llms: All large language models are in 1.58 bits," *arXiv preprint arXiv:2402.17764*, 2024.

[50] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[51] X. Men, M. Xu, Q. Zhang, B. Wang, H. Lin, Y. Lu, X. Han, and W. Chen, "Shortgpt: Layers in large language models are more redundant than you expect," *arXiv preprint arXiv:2403.03853*, 2024.

Supplementary Material

A Proofs of theoretical results

In this first appendix, we provide proofs for the various theoretical results in Section 3.

A.1 Proof of Theorem 1

For a pair of vectors $\mathbf{x} \in \mathbf{X}$ and $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$, and using (5), we have the squared error:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x}\|^2 + \|\tilde{\mathbf{x}}\|^2 - 2\mathbf{x}^T\tilde{\mathbf{x}} = \|\mathbf{x}\|^2 + \|\tilde{\mathbf{x}}\|^2 - 2\sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)\mathbf{p_i}^T\mathbf{x}$$

$$\Rightarrow \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x}\|^2 + \|\tilde{\mathbf{x}}\|^2 - 2\sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)^2$$

Furthermore, note that

$$\|\tilde{\mathbf{x}}\|^2 = \tilde{\mathbf{x}}^T\tilde{\mathbf{x}} = \left(\sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)\mathbf{p_i}\right)^T\left(\sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)\mathbf{p_i}\right)$$

and since $\{\mathbf{p}_i\}_{i=1}^L$ are orthonromal, cross product terms vanish and we have: $\|\tilde{\mathbf{x}}\|^2 = \sum_{i=1}^L\left(\mathbf{p_i}^T\mathbf{x}\right)^2$ which we plug back into the expression for the squared error:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x}\|^2 + \sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)^2 - 2\sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)^2 = \|\mathbf{x}\|^2 - \sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)^2$$

Furthermore, note that:

$$\left(\mathbf{p_i}^T\mathbf{x}\right)^2 = \left(\mathbf{p_i}^T\mathbf{x}\right)\left(\mathbf{p_i}^T\mathbf{x}\right) = \left(\mathbf{p_i}^T\mathbf{x}\right)\left(\mathbf{x}^T\mathbf{p_i}\right) = \mathbf{p_i}^T\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{p_i}$$

where we used the commutativity of dot product and associativity of matrix multiplication. Thus the squared error is given by:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x}\|^2 - \sum_{i=1}^{L}\mathbf{p_i}^T\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{p_i}$$

Finally, we take expectation on both sides and obtain a formula for the MSE:

$$\mathbb{E}\left[\|\mathbf{x} - \tilde{\mathbf{x}}\|^2\right] = \mathbb{E}\left[\|\mathbf{x}\|^2\right] - \mathbb{E}\left[\sum_{i=1}^{L}\mathbf{p_i}^T\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{p_i}\right] = \mathbb{E}\left[\|\mathbf{x}\|^2\right] - \sum_{i=1}^{L}\mathbf{p_i}^T\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\mathbf{p_i}$$

where we used linearity of expectation and the fact that $\{\mathbf{p}_i\}_{i=1}^L$ are not random. In this formula for the MSE, $\mathbb{E}\left[\|\mathbf{x}\|^2\right]$ does not depend on $\{\mathbf{p}_i\}_{i=1}^L$, and therefore, minimizing the MSE is equivalent to **maximizing** the following expression involving the auto-correlation matrix:

$$\sum_{i=1}^{L}\mathbf{p_i}^T\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\mathbf{p_i} = \sum_{i=1}^{L}\mathbf{p_i}^T\mathbf{C}_{\mathbf{X}}\mathbf{p_i}$$

where each term in the summation is a quadratic form on the positive semi-definite auto-correlation matrix $\mathbf{C}_{\mathbf{X}}$. Since $\{\mathbf{p}_i\}_{i=1}^L$ are orthonormal, this is an equivalent form of the Rayleigh quotient [48] and the solution is to assign $\{\mathbf{p}_i\}_{i=1}^L$ as the L principal eigenvectors of $\mathbf{C}_{\mathbf{X}}$. This concludes the proof of Theorem 1.

A.2 Proof of Corollary 2

The result can be readily obtained as a consequence of the following:

$$\mathbb{E}\left[\frac{\|\tilde{\mathbf{x}}-\mathbf{x}\|^2}{\|\mathbf{x}\|^2}\right] = \mathbb{E}\left[\left\|\frac{\tilde{\mathbf{x}}}{\|\mathbf{x}\|} - \frac{\mathbf{x}}{\|\mathbf{x}\|}\right\|^2\right] = \mathbb{E}_{\mathbf{X}}\left[\left\|\frac{\sum_{i=1}^{L}\langle\mathbf{p}_i,\mathbf{x}\rangle\mathbf{p}_i}{\|\mathbf{x}\|} - \frac{\mathbf{x}}{\|\mathbf{x}\|}\right\|^2\right]$$

$$\Rightarrow \mathbb{E}\left[\frac{\|\tilde{\mathbf{x}}-\mathbf{x}\|^2}{\|\mathbf{x}\|^2}\right] = \mathbb{E}\left[\left\|\sum_{i=1}^{L}\langle\mathbf{p}_i,\frac{\mathbf{x}}{\|\mathbf{x}\|}\rangle\mathbf{p}_i - \frac{\mathbf{x}}{\|\mathbf{x}\|}\right\|^2\right]$$

Therefore, the setup is identical to that of Theorem 1 and we may apply the same solution as Appendix A.1 above. The only difference is that activation vectors are L_2-normalized which is why $\hat{\mathbf{C}}_{\mathbf{X}}$ (which is also positive semi-definite) is used in lieu of $\mathbf{C}_{\mathbf{X}}$ in Corollary 2.

A.3 Proof of Proposition 3

A prelimnary result needed is to show that for any activation vector, we have $\|\tilde{\mathbf{x}}\|^2 < \|\mathbf{x}\|^2$. We first note that while $\tilde{\mathbf{x}} = \sum_{i=1}^{L}\langle\mathbf{p}_i,\mathbf{x}\rangle\mathbf{p}_i$ per (5), we also have that $\mathbf{x} = \sum_{i=1}^{K}\langle\mathbf{p}_i,\mathbf{x}\rangle\mathbf{p}_i$, where $\{\mathbf{p}_i\}_{i=1}^{K}$ extend the set of orthonormal vectors $\{\mathbf{p}_i\}_{i=1}^{L}$ to be complete, i.e., equivalent to no truncation of columns of the full rank matrix $\mathbf{V}$ when constructing $\mathbf{P}$, regardless of the metric being optimized.

Using orthonormality of projection vectors, similar to the proof of Theorem 1 in Appendix A.1 above, we obtain $\|\tilde{\mathbf{x}}\|^2 = \sum_{i=1}^{L}\left(\mathbf{p_i}^T\mathbf{x}\right)^2$ and $\|\mathbf{x}\|^2 = \sum_{i=1}^{K}\left(\mathbf{p_i}^T\mathbf{x}\right)^2$. Therefore:

$$\|\mathbf{x}\|^2 - \|\tilde{\mathbf{x}}\|^2 = \sum_{i=L+1}^{K}\left(\mathbf{p_i}^T\mathbf{x}\right)^2 \geq 0 \Rightarrow \|\tilde{\mathbf{x}}\|^2 < \|\mathbf{x}\|^2$$

where we used the fact that a sum of non-negative quantities is non-negative.

Then for a scalar $y \in \mathbf{Y}$ and its counterpart $\tilde{y} \in \tilde{\mathbf{Y}}$, we have:

$$\begin{aligned}(y-\tilde{y})^2 &= \left(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\tilde{\mathbf{x}}\right)^2 = \left(\mathbf{w}^T\mathbf{x}\right)^2 + \left(\mathbf{w}^T\tilde{\mathbf{x}}\right)^2 - 2\left(\mathbf{w}^T\mathbf{x}\mathbf{w}^T\tilde{\mathbf{x}}\right)\\ &\leq \|\mathbf{w}\|^2\cdot\|\mathbf{x}\|^2 + \|\mathbf{w}\|^2\cdot\|\tilde{\mathbf{x}}\|^2 - 2\left(\mathbf{w}^T\mathbf{x}\mathbf{w}^T\tilde{\mathbf{x}}\right)\\ &\leq \|\mathbf{w}\|^2\cdot\|\mathbf{x}\|^2 + \|\mathbf{w}\|^2\cdot\|\mathbf{x}\|^2 - 2\left(\mathbf{w}^T\mathbf{x}\mathbf{w}^T\tilde{\mathbf{x}}\right)\\ &= 2\|\mathbf{w}\|^2\cdot\|\mathbf{x}\|^2 - 2\left(\mathbf{w}^T\mathbf{x}\mathbf{w}^T\tilde{\mathbf{x}}\right)\end{aligned}$$

where the first upper bound uses the Cauchy-Schwarz inequality while the second uses $\|\tilde{\mathbf{x}}\|^2 < \|\mathbf{x}\|^2$ which we proved above. Taking expectations on both sides of the inequality yields the upper bound on GO-MSE in (8).

Next, when a first order Taylor approximation on the loss function is assumed, we have the following relation between the unperturbed loss value $\mathcal{L}$ and its counterpart $\tilde{\mathcal{L}}$ when an activation vector $\mathbf{x}$ is projected to $\tilde{\mathbf{x}}$ per (5):

$$\tilde{\mathcal{L}} = \mathcal{L} + \nabla_{\mathbf{x}}^T\left(\tilde{\mathbf{x}}-\mathbf{x}\right) \Rightarrow \tilde{\mathcal{L}} - \mathcal{L} = \nabla_{\mathbf{x}}^T\tilde{\mathbf{x}} - \nabla_{\mathbf{x}}^T\mathbf{x}$$

$$\Rightarrow \left(\tilde{\mathcal{L}}-\mathcal{L}\right)^2 = \left(\nabla_{\mathbf{x}}^T\tilde{\mathbf{x}} - \nabla_{\mathbf{x}}^T\mathbf{x}\right)^2 = \left(\nabla_{\mathbf{x}}^T\tilde{\mathbf{x}}\right)^2 + \left(\nabla_{\mathbf{x}}^T\mathbf{x}\right)^2 - 2\left(\nabla_{\mathbf{x}}^T\mathbf{x}\nabla_{\mathbf{x}}^T\tilde{\mathbf{x}}\right)$$

once more, we use the Cauchy-Schwarz inequality and the fact that $\|\tilde{\mathbf{x}}\|^2 < \|\mathbf{x}\|^2$ to establish:

$$\left(\nabla_{\mathbf{x}}^T\mathbf{x}\right)^2 \leq \|\nabla_{\mathbf{x}}\|^2\cdot\|\mathbf{x}\|^2 \quad \& \quad \left(\nabla_{\mathbf{x}}^T\tilde{\mathbf{x}}\right)^2 \leq \|\nabla_{\mathbf{x}}\|^2\cdot\|\tilde{\mathbf{x}}\|^2 \leq \nabla_{\mathbf{x}}\|^2\cdot\|\mathbf{x}\|^2$$

which we plug into the difference in network losses above to obtain:

$$\left(\tilde{\mathcal{L}}-\mathcal{L}\right)^2 \leq 2\|\nabla_{\mathbf{x}}\|^2\cdot\|\mathbf{x}\|^2 - 2\left(\nabla_{\mathbf{x}}^T\mathbf{x}\nabla_{\mathbf{x}}^T\tilde{\mathbf{x}}\right)$$

Taking expectations on both sides yields the upper bound on NL-MSE in (9). This completes the proof of Proposition 3.

A.4 Proof of Theorem 4

In order to minimize the upper bound on the GO-MSE in (8), note that it suffices to maximize the quantity $2\mathbb{E}\left[\langle \mathbf{w}, \mathbf{x}\rangle \cdot \langle \mathbf{w}, \tilde{\mathbf{x}}\rangle\right]$ since $2\mathbb{E}\left[\|\mathbf{w}\|^2 \cdot \|\mathbf{x}\|^2\right]$ does not depend on $\{\mathbf{p}_i\}_{i=1}^L$. We have the following:

$$\begin{aligned} 2\langle \mathbf{w}, \mathbf{x}\rangle \cdot \langle \mathbf{w}, \tilde{\mathbf{x}}\rangle &= 2\mathbf{w}^T\mathbf{x}\mathbf{w}^T\left(\sum_{i=1}^{L}\left(\mathbf{p}_i^T\mathbf{x}\right)\mathbf{p}_i\right) = 2\sum_{i=1}^{L}\mathbf{w}^T\mathbf{x}\mathbf{w}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{x} \\ &= \sum_{i=1}^{L}\left(\mathbf{w}^T\mathbf{x}\mathbf{w}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{x} + \mathbf{w}^T\mathbf{x}\mathbf{w}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{x}\right) \end{aligned}$$

But since the dot product is commutative, i.e., $\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$ for any two vectors $\mathbf{a}$ and $\mathbf{b}$, we may rearrange each of the two identical terms inside the summation as follows:

$$\mathbf{w}^T\mathbf{x}\mathbf{w}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{x} = \mathbf{p}_i^T\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T\mathbf{p}_i \quad \& \quad \mathbf{w}^T\mathbf{x}\mathbf{w}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{x} = \mathbf{p}_i^T\mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{p}_i$$

Therefore, we obtain:

$$\begin{aligned} 2\langle \mathbf{w}, \mathbf{x}\rangle \cdot \langle \mathbf{w}, \tilde{\mathbf{x}}\rangle &= \sum_{i=1}^{L}\left(\mathbf{p}_i^T\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T\mathbf{p}_i + \mathbf{p}_i^T\mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{p}_i\right) \\ &= \sum_{i=1}^{L}\mathbf{p}_i^T\left(\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right)\mathbf{p}_i \end{aligned}$$

Taking expectations, we find that the quantity that needs to be maximized in order to minimize the bound on GO-MSE in (8) is:

$$\sum_{i=1}^{L}\mathbf{p}_i^T\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right]\mathbf{p}_i$$

Similar to the proof of Theorem 1 in Appendix A.1, this is yet again a sum of a quadratic form over the orthonormal set of vectors $\{\mathbf{p}_i\}_{i=1}^L$ and the solution is therefore to assign these vectors as the L principal vectors of $\mathbf{C} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right]$ as per (10).

Note that the derivation above decomposed the dot products to obtain a quadratic form on the matrix $\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T$ because its symmetry is required for real eigenvalue decomposition. Also note that if positive definiteness is not achieved, we sort absolute values of eigenvalues. Finally, observe that this solution requires no overhead on the calibration process. Indeed, assuming weights and activations are independent, we note that $\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{x}\mathbf{x}^T\right] = \mathbf{C_X}\mathbf{C_W} + \mathbf{C_W}\mathbf{C_X}$ where $\mathbf{C_W} = \mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right]$ is the weight auto-correlation matrix, which can simply be calibrated as $\mathbf{C_W} = \mathbf{W}\mathbf{W}^T/N$. Thus we only require left and right scaling of the calibrated auto-correlation matrix.

Similarly, to minimize the upper bound on NL-MSE in (9), it suffices to maximize $2\mathbb{E}\left[\langle \nabla_\mathbf{x}, \mathbf{x}\rangle \cdot \langle \nabla_\mathbf{x}, \tilde{\mathbf{x}}\rangle\right]$. Using the exact same derivation as the above, replacing $\mathbf{w}$ by $\nabla_\mathbf{x}$, we obtain that the quantity to be maximized is

$$\sum_{i=1}^{L}\mathbf{p}_i^T\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\nabla_\mathbf{x}\nabla_\mathbf{x}^T + \nabla_\mathbf{x}\nabla_\mathbf{x}^T\mathbf{x}\mathbf{x}^T\right]\mathbf{p}_i$$

which is done by assigning $\{\mathbf{p}_i\}_{i=1}^L$ as the L principal vectors of $\mathbf{C} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\nabla_\mathbf{x}\nabla_\mathbf{x}^T + \nabla_\mathbf{x}\nabla_\mathbf{x}^T\mathbf{x}\mathbf{x}^T\right]$ as per (10). Calibrating this matrix does require an extra step, where we perform a backward pass to estimate activation gradients. For ease of implementation, in our results, we make an approximation on the per-sequence independence of activation vectors and their gradients. This greatly reduces the memory requirements of the calibration process. And for each sample sequence in the calibration set, we compute $\mathbf{C}^{(i)} = \left(\mathbf{X}^{(i)}\mathbf{X}^{(i)T}\mathbf{G}^{(i)}\mathbf{G}^{(i)T} + \mathbf{G}^{(i)}\mathbf{G}^{(i)T}\mathbf{X}^{(i)}\mathbf{X}^{(i)T}\right)/M^2$ where $\mathbf{G}^{(i)}$ is the gradient tensor (whose vectors are instantiations of $\nabla_\mathbf{x}$). As always, we end the calibration phase by averaging across samples: $\mathbf{C} = \sum_{i=1}^{B}\mathbf{C}^{(i)}/B$. This completes the proof of Theorem 4.

B Implementation details

In this appendix, we discuss all details behind our implementation in Section 4. These details include per-model specific application of ESPACE as well as retraining recipes. We strive to provide excessive details such that independent reproducibility of our results is seamless. We also encourage readers to reach out to us (after blind reviewing) for any questions on implementations.

B.1 Software implementation

As was mentioned in Section 4, our implementation is built on top of Megatron-LM [33] which itself is based on the Pytorch framework. We use Pytorch for all extra introductions needed by ESPACE except for eigenvalue decomposition. Our experiments were carried out in a cluster of A100 GPUs and use BF16 precision.

Specifically, during the calibration process, we use Pytorch to track the required auto-correlation matrices; this simply done by averaging repeated instantiations of $\mathbf{X}\mathbf{X}^T$ as described in Section 3.

Once the calibration of auto-correlation matrices is over, we use DLPack to transfer them from the Pytorch framework to RAPIDS framework. We then use the CUPY library in RAPIDS to perform fast (a few milliseconds per auto-correlation matrix) eigenvalue decomposition on GPUs. After truncating eigenvectors, we send back the projection matrix to Pytorch using DLPack.

Once the projection matrix $\mathbf{P}$ is calibrated and inference/training is to be done using ESPACE, we simply insert a projection operation within the Megatron implementation to perform the operations in (3) as appropriate. The projection matrices are inserted as Pytorch buffers, rather than parameters, since they do not get updated during training.

B.2 ESPACE configurations

In Section 4, we mentioned that ESPACE was applied at each layer such that the number of components L satisfies two constraints: (a) be a power of two for best tensor core utilization, and (b) yield a compression of at least 50% at that layer. The exact values of L for each model and layer type are included in Table 2. Note that the only exception corresponds to QKV layers in Llama2-13B and Nemotron4-15B, where we use a value of $L = 2048$ which corresponds to a compression of $\sim$ 45% instead of $>$50% at least. This is only because this amount of compression is already significant that we didn't feel the need to push for $L = 1024$, which would have lead to a compression of $> 70\%$.

B.3 Retraining hyperparameters

By and large, we use the exact same recipe that was used to pretrain the open source GPT3 models [33]. As mentioned in Section 4, the only modification to hyperparameters is disabling dropout and weight decay, and identical hyperparameters are used for both sets of experiments on GPT3 and Llama2 families. The only arbitrary choices we had to make was on the selection of learning rate schedule and global batch size. We use a cosine decay for all runs, and remaining choicesa are as follows:

- For GPT3-1.3B, the initial learning rate is set to 1.0×10^{-4}, the final learning rate is set to 1.0×10^{-5}, and the global batch size is set to 512.
- For GPT3-8B, the initial learning rate is set to 5.0×10^{-5}, the final learning rate is set to 5.0×10^{-6}, and the global batch size is set to 512.
- For GPT3-22B, the initial learning rate is set to 5.0×10^{-5}, the final learning rate is set to 5.0×10^{-6}, and the global batch size is set to 1024.
- For Llama2-7B and Llama2-13B, training is done in two stages (each of 100B tokens). In the first stage, the initial learning rate is set to 5.0×10^{-4}, and the final learning rate is set to 5.0×10^{-5}. In the second stage, the initial learning rate is set to 5.0×10^{-5}, and the final learning rate is set to 5.0×10^{-6}. For both stages, the global batch size is set to 256.
- For Nemotron4-15B, the initial learning rate is set to 1.0×10^{-5}, the final learning rate is set to 0, and the global batch size is set to 512.

We did not perform hyperparameter tuning, the above was purely arbitrary, but based on the following intuition:

Table 2: Number of components L retained by ESPACE for each layer. For models implementing Tensor Parallelism (TP), we apply ESPACE *per rank*.

Attn QKV layers	Attn Proj layers	FC1 (H-to-4H) layers	FC2 (4H-to-H) Layers
GPT3-1.3B (TP=1)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 2048 \quad N = 6144$	$K = 2048 \quad N = 2048$	$K = 2048 \quad N = 8192$	$K = 8192 \quad N = 2048$
ESPACE: $L = 512$	**ESPACE**: $L = 512$	**ESPACE**: $L = 512$	**ESPACE**: $L = 512$
Compression: 66%	**Compression**: 50%	**Compression**: 69%	**Compression**: 69%
GPT3-8B (TP=4)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 4096 \quad N = 12288$	$K = 4096 \quad N = 4096$	$K = 4096 \quad N = 16384$	$K = 16384 \quad N = 4096$
GEMM per rank:	**GEMM per rank**:	**GEMM per rank**:	**GEMM per rank**:
$K = 4096 \quad N = 3072$	$K = 1024 \quad N = 4096$	$K = 4096 \quad N = 4096$	$K = 4096 \quad N = 4096$
ESPACE: $L = 1024$	**ESPACE**: $L = 256$	**ESPACE**: $L = 1024$	**ESPACE**: $L = 1024$
Compression: 66%	**Compression**: 69%	**Compression**: 69%	**Compression**: 50%
GPT3-22B (TP=8)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 6144 \quad N = 18432$	$K = 6144 \quad N = 6144$	$K = 6144 \quad N = 24576$	$K = 24576 \quad N = 6144$
GEMM per rank:	**GEMM per rank**:	**GEMM per rank**:	**GEMM per rank**:
$K = 6144 \quad N = 2304$	$K = 768 \quad N = 6144$	$K = 6144 \quad N = 3072$	$K = 3072 \quad N = 6144$
ESPACE: $L = 2048$	**ESPACE**: $L = 256$	**ESPACE**: $L = 2048$	**ESPACE**: $L = 1024$
Compression: 56%	**Compression**: 63%	**Compression**: 58%	**Compression**: 50%
Llama2-7B (TP=4)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 4096 \quad N = 12288$	$K = 4096 \quad N = 4096$	$K = 4096 \quad N = 22016$	$K = 11008 \quad N = 4096$
GEMM per rank:	**GEMM per rank**:	**GEMM per rank**:	**GEMM per rank**:
$K = 4096 \quad N = 3072$	$K = 1024 \quad N = 4096$	$K = 4096 \quad N = 5504$	$K = 2752 \quad N = 4096$
ESPACE: $L = 1024$	**ESPACE**: $L = 256$	**ESPACE**: $L = 1024$	**ESPACE**: $L = 512$
Compression: 66%	**Compression**: 69%	**Compression**: 69%	**Compression**: 69%
Llama2-13B (TP=8)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 5120 \quad N = 15360$	$K = 5120 \quad N = 5120$	$K = 5120 \quad N = 27648$	$K = 13824 \quad N = 5120$
GEMM per rank:	**GEMM per rank**:	**GEMM per rank**:	**GEMM per rank**:
$K = 5120 \quad N = 1920$	$K = 640 \quad N = 5120$	$K = 5120 \quad N = 3456$	$K = 1728 \quad N = 5120$
ESPACE: $L = 2048$	**ESPACE**: $L = 256$	**ESPACE**: $L = 2048$	**ESPACE**: $L = 1024$
Compression: 47%	**Compression**: 55%	**Compression**: 53%	**Compression**: 60%
Nemotron4 (TP=8)			
GEMM Dimension:	**GEMM Dimension**:	**GEMM Dimension**:	**GEMM Dimension**:
$K = 6144 \quad N = 8192$	$K = 6144 \quad N = 6144$	$K = 6144 \quad N = 24576$	$K = 24576 \quad N = 6144$
GEMM per rank:	**GEMM per rank**:	**GEMM per rank**:	**GEMM per rank**:
$K = 6144 \quad N = 1024$	$K = 768 \quad N = 6144$	$K = 6144 \quad N = 3072$	$K = 3072 \quad N = 6144$
ESPACE: $L = 2048$	**ESPACE**: $L = 256$	**ESPACE**: $L = 2048$	**ESPACE**: $L = 1024$
Compression: 42%	**Compression**: 67%	**Compression**: 58%	**Compression**: 50%

- For GPT3 models, we use a smaller learning rate for larger models, and start with a learning rate $10\times$ smaller than it's pre-training value. We use identical batch sizes as pre-training.
- For Llama2 models, as pre-training hyperparameters are undisclosed, we use our best guess of what *could* work well. The two stage training approach is inspired by a recent work on 1.58-bit LLMs [49], while the choice of a batch size of 256 is inspired by ChipNemo [50].

B.4 GEMM latency measurements

Here we describe the methodology employed to measure GEMM latency as reported in Table 1. We assume a batch size of 1, such that the M dimension of tensor $\mathbf{X}$ equals the sequence length (2048 and 4096 for GPT3 and Llama2 models, respectively). We also assume a single-GPU implementation throughout, such that any tensor parallelism is first folded into single GEMM per-layer. Similar to our accuracy experiments, we use BF16 precision for all latency measurements.

For each GEMM layer implementing either (1) or (3), we measure its latency individually. In Table 1, we report aggregated measurements depending on the model configuration. Specifically, we measure

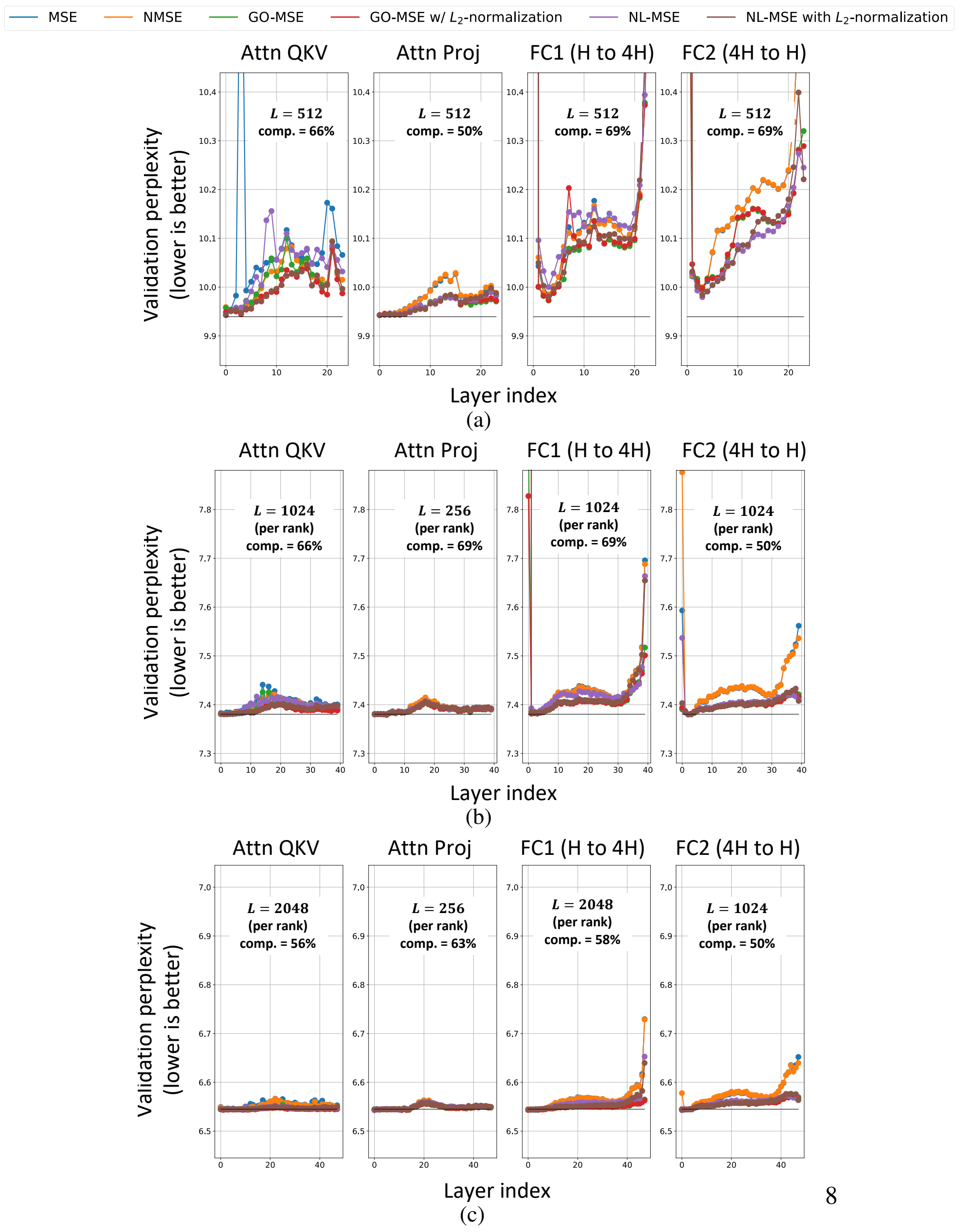

Figure 5: Sensitivity studies on the choice of projection construction for (a) GPT3-1.3B, (b) GPT3-8B, (c) GPT3-22B. For each layer, we apply ESPACE out-of-the-box using the six various candidates for the projection matrix $\mathbf{P}$ constructed in Section 3. The black line corresponds to the baseline perplexity.

latency of computing QKV, Proj, FC1, and FC2 GEMMs with dimensions listed in Table 2, and then add all results together for each transformer block of the corresponding model.

We measure individual GEMM latencies in Pytorch. Specifically, for each configuration, we sample 1000 set of matrices of appropriate dimension and compute the appropriate GEMM. We synchronize before and after the computation occurs, and record times after synchronization. The elapsed times are averaged and then aggregated. Since the measurements were taken with native PyTorch code, we note that the implementation is un-optimized. Further improvements could be possible in future work from removing PyTorch overheads, implementing custom fused kernels, or other optimizations.

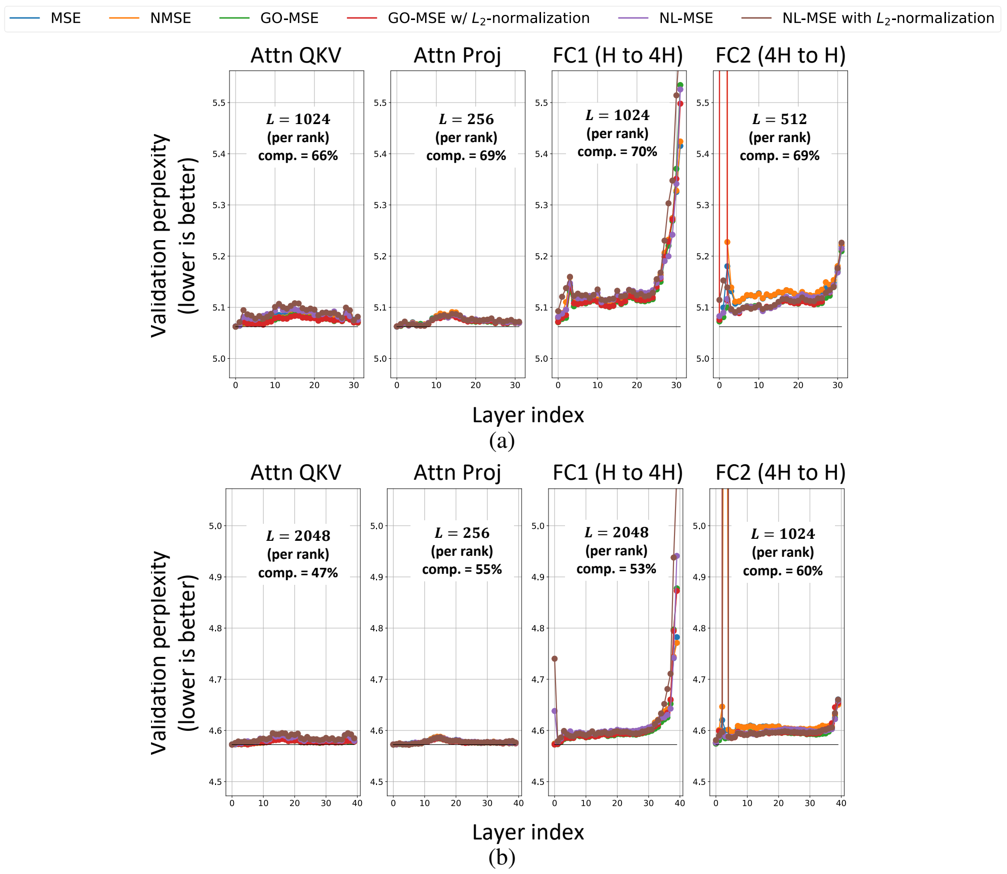

Figure 6: Sensitivity studies on the choice of projection construction for (a) Llama2-7B, (b) Llama2-13B. For each layer, we apply ESPACE out-of-the-box using the six various candidates for the projection matrix $\mathbf{P}$ constructed in Section 3. The black line corresponds to the baseline perplexity.

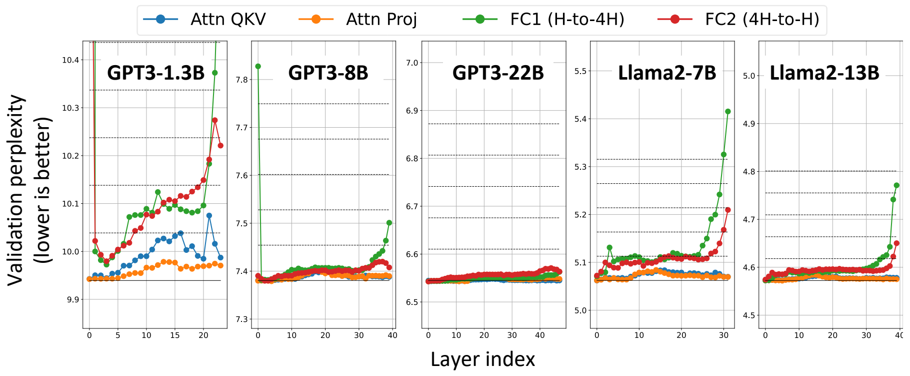

Figure 7: Validation perplexity when ESPACE is applied out-of-the-box one layer at a time using the best calibrated projection matrix $\mathbf{P}$ as identified by the sensitivity study in Figures 5 and Figures 6. The black line corresponds to the baseline perplexity and the dashed lines correspond to 1% increments over it.

C Additional experimental results

In this appendix, we include additional experimental results that were not included in the main paper. These results are not essential to the description of our work nor its conclusion, and the main paper integrally contains all essential information related to our contribution. The additional results listed in this appendix are for the benefit of readers interested in going further and learning about fine-grained details behind the main results of Section 4.

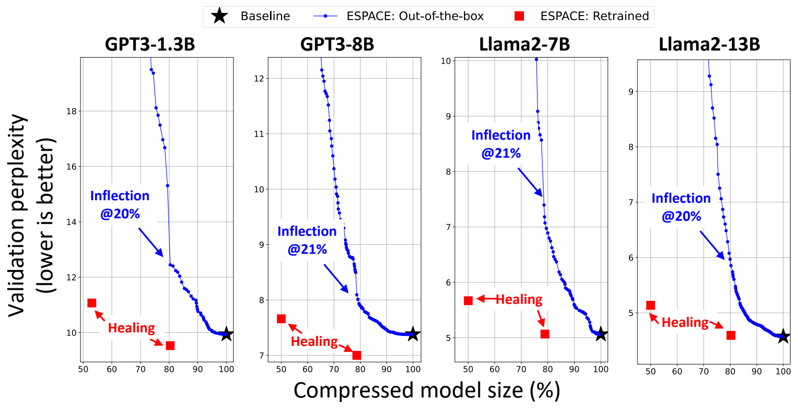

Figure 8: Progressive out-of-the-box application of ESPACE on GPT3{1.3B, 8B} and Llama2-{7B, 13B}. The plot for GPT3-22B was provided in the main text in Figure 3. The progressive application of ESPACE is based on the ranking of layers from least to most destructive based on validation perplexity sensistivity in Figure 7.

C.1 Sensitivity studies on the construction of projection matrix

In Section 3, we presented theoretical results leading to six choices for the construction of projection matrix $\mathbf{P}$. These constructions were done in a manner to optimize one of the following fidelity metrics: MSE, NMSE, GO-MSE, GO-MSE with L_2-normalization, NL-MSE, and NL-MSE with L_2-normalization. Here we show the impact of various choices of $\mathbf{P}$ constructions on the validation perplexity, at a per-layer granularity. Specifically, we apply the ESPACE projection out-of-the-box one layer at a time, for each of the six candidates, and evaluate the resulting validation perplexity. In this way, we are able to determine which of the six candidate choices of $\mathbf{P}$ works best at each layer.

The results of this sensitivity analysis are included in Figures 5 and 6 for GPT3 and Llama2 models, respectively. It is shown that the best choice of projection matrix $\mathbf{P}$ depends on layer instance, and there is no clear pattern to find out which solution works best *a priori*. This result justifies the need to optimize several proxy metric for accuracy, not just the MSE of activation approximation. Particularly, most solutions do appear to be related to GO-MSE and NL-MSE, as well as thei L_2-normalized variants. Therefore, these results provide supporting evidence on the importance of the results in Proposition 3 and Theorem A.4. This also validates the choice of using bounds on GO-MSE and NL-MSE for optimization since closed form solution for the unbounded metrics are elusive.

C.2 Progressive application of ESPACE to the layers of a network

Once the best projection matrix $\mathbf{P}$ is identified for each layer, we plot the corresponding validation perplexity for out-of-the-box application of ESPACE at the corresponding layer using the corresponding choice of $\mathbf{P}$. These results are shown in Figure 7, where several observations are made. First, larger models have more resilience to out-of-the-box application of ESPACE; this observation was made in Section 4. Second, it appears that FC1 layers are the most sensitive ones, followed by FC2, and QKV/Proj layers are generally robust to the application of ESPACE. Finally, we observe that in some instances, some layers close to the input and output (i.e., on either ends of the model) appear to be most sensitive to the application of ESPACE. This behavior was observed in other works on compression, such as shortGPT [51].

As mentioned in Section 4, layers are then sorted according to their impact on perplexity from least to most destructive. ESPACE is then progressively applied to out-of-the-box to all layers according to this ranking. In Figure 3 in the main text, we had shown the results corresponding to this applciation for GPT3-22B. In Figure 8, we show similar results for the other four networks we experimented on, i.e., GPT3-{1.3B, 8B} and Llama2-{7B, 13B}. Similar to the findings on GPT3-22B, we do observe an inflection point after which accuracy degradation accelerates. This inflection occurs around 20% for GPT3-{1.3B, 8B} and Llama2-{7B, 13B}. With retraining, the healing process recovers accuracy for all models, as detailed in Section 4.

We note the following:

- For GPT3-1.3B, we exclude some layers from the application of ESPACE. These are the layers for which out-of-the-box application of ESPACE leads to a validation perplexity increase of more than 2% compared to the baseline. These layers can be found in Figure 7 and correspond to several FC1 and FC2 layers close to either ends of the model. For this reason, GPT3-1.3B is compressed to 47% instead of 50% in Section 4 and Table 1.
- For GPT3-22B, we apply ESPACE to all layers since validation perplexity increase is very small in Figure 7. For this reason, the overall compression for GPT3-22B is slightly over 50%; it is 55% in Section 4 and Table 1.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: *We accurately report the paper's contribution in the abstract and introduction; this is why we chose to have a bulleted list in a standalone subsection in Section 1.2. For clarity, we listed all contributions in the paper, in the order in which they appear: our proposal, followed by theoretical results, and summaries of experimental results.*

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: *In our discussion on related works in Section 1.1, we have listed possible combinations of applying ESPACE alongside other methods (e.g., quantization or parameter efficient tuning), and have mentioned that this was not the scope of our paper. Furthermore, in the experimental Section 4, we have discussed the limitations of only metricizing compression via model size and weight times activation GEMM latency reductions. We argued that a more nuanced study is needed on the inference cost implications as Transformers comprise other GEMMs, and the regime (context pre-fill versus auto-regressive phase) needs to be taken into account. We have mentioned that a detailed study on inference cost with ESPACE is part of future work.*

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: *We provide detailed proofs in Appendix A for all results in the theoretical Section 3. Furthermore, following each Theorem, Proposition, and Corollary in Section 3, we also provide a teaser to the full proof in the main text, for the benefit of interested readers, where we try to provide the main intuition before referring to full proofs in Appendix A.*

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: *All details, including fine-grained configurations, software toolkit employed, and training recipes including hyperaparameters are included in Appendix B. These are also briefly mentioned in the main paper in Section 4, with references to Appendix B for the readers interested in more details needed for full reproducibility.*

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: *As was mentioned in the previous question, we have provided all implementation details required to reproduce our results, including hyperparameters and a description of software implementation in Appendix B. We note that our implementation is based on the open-source Megatron-LM [33] and that additions made to this software toolkit needed to reproduce our results are included in Appendix B, where we also include an invitation to the reader to reach out to us (after blind reviewing is over) with any questions regarding reproducibility. As such, we believe the description of the work in the paper is sufficient for reproducibility; yet, we are happy to consider open sourcing our code in the future.*

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so 'No' is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: *Yes, and this was already mentioned in our responses to the two questions above. We provide all fine-grained details of our implementation in Appendix B, which is referred to in the main text.*

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: *While we do not report error bars, we do evaluate our method on a variety of downstream tasks, for the specific purpose of making conclusions with statistical significance; rather than relying on one number here and there. We also note that this approach of evaluating on several tasks is also widely adopted in the community for works on compression of LLMs.*

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: *These are reported in Section 4 and Appendix B.*

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: *Yes, we are only working on a method to compress LLMs using tensor decomposition of activations.*

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: *As per the guideline below, our work falls under the umbrella of optimization to the implementation of LLMs. Similar to the example provided in the guideline, we believe that there is no need to point out societal implications of making LLMs run faster.*

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: *There are no such risks associated with our work, and as such the paper does not describe safeguards.*

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: *We have done our best to provide credit to any prior work upon which we have built ours. This is mostly the case for Megatron-LM [33], which we used for our experiments, and cited extensively throughout Section 4 and Appendix B.*

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: *The paper does not release new assets.*

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: *The paper does not involve crowdsourcing nor research with human subjects.*

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: *The paper does not involve crowdsourcing nor research with human subjects.*

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.