ESPACE: DIMENSIONALITY REDUCTION OF ACTIVATIONS FOR MODEL COMPRESSION

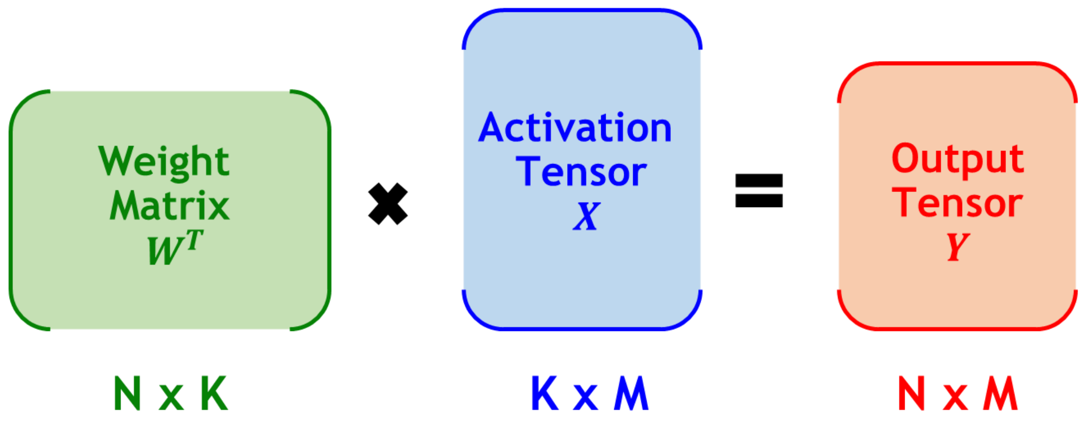
Charbel Sakr and Brucek Khailany

COMPRESSING MODELS USING MATRIX FACTORIZATION

General Matrix Multiplication (GEMM)

$$Y = W^T X$$

⬆️ trainable parameters: ⬆️ expressivity ✅
⬆️ model size: ⬆️ inference cost ❌



K = input dimension, N= output dimension, M = batch ∗ sequence

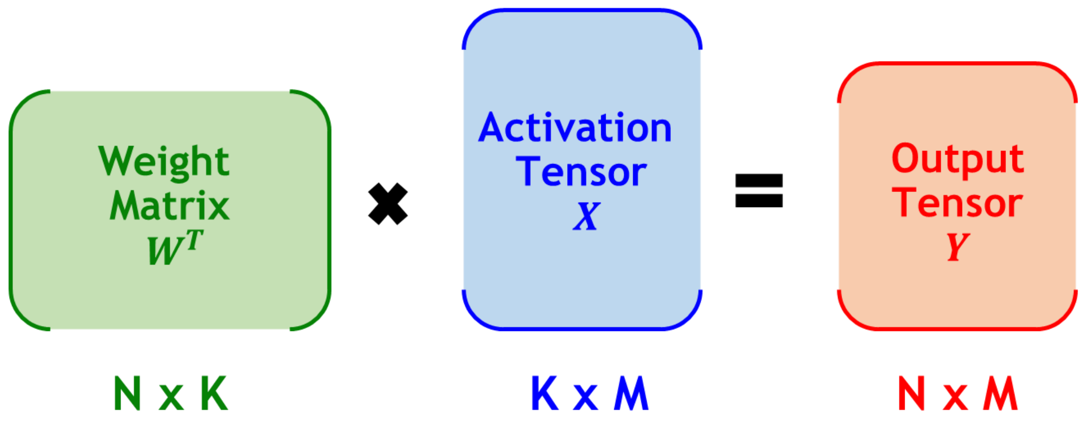
- Matrix multiplications between weights and activations
- Pervasive operation in LLM serving and training
- Weight matrix contains the model parameters
  - it stores its representative power
- Activation tensor is generated on the fly for every input

COMPRESSING MODELS USING MATRIX FACTORIZATION

General Matrix Multiplication (GEMM)

$Y = W^T X$

⬆️ trainable parameters: ⬆️ expressivity ✅
⬆️ model size: ⬆️ inference cost ❌



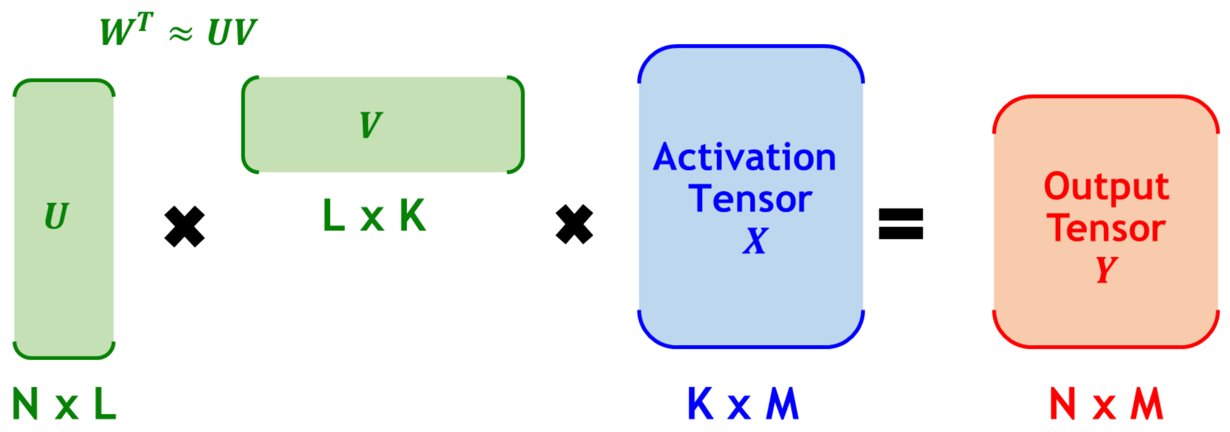
K = input dimension, N= output dimension, M = batch ∗ sequence

- Matrix multiplications between weights and activations
- Pervasive operation in LLM serving and training
- Weight matrix contains the model parameters
  - it stores its representative power
- Activation tensor is generated on the fly for every input

GEMM with Weight Decomposition (e.g., truncated SVD)

$Y \approx UVX$

⬇️ trainable parameters: ⬇️ expressivity ❌
⬇️ model size: ⬇️ inference cost ✅



K = input dimension, N= output dimension, M = batch ∗ sequence
L = intermediate dimension

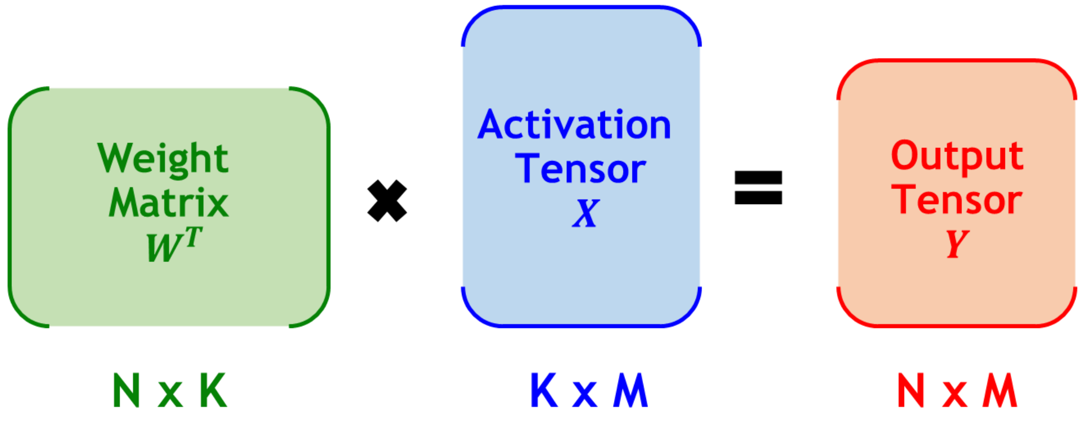
- Weight matrix can be approximated using factorization
  - e.g. a truncated SVD, where we hope to factorize $W^T \approx UV$
- To achieve compression, we need $L \ll K$
  - Since we now have two matrices instead of one
- Such decomposition has two limitations
  - fewer weights → lesser representative power
  - Breakage of weight structure → optimizer cannot be recovered

COMPRESSING MODELS USING MATRIX FACTORIZATION

General Matrix Multiplication (GEMM)

$Y = W^T X$

⬆️ trainable parameters: ⬆️ expressivity ✅
⬆️ model size: ⬆️ inference cost ❌



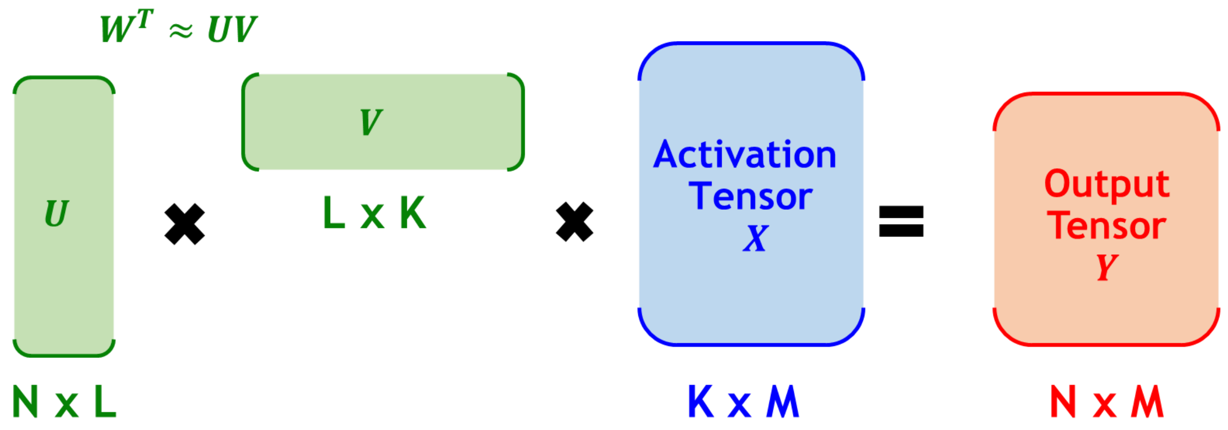
K = input dimension, N= output dimension, M = batch ∗ sequence

- Matrix multiplications between weights and activations
- Pervasive operation in LLM serving and training
- Weight matrix contains the model parameters
  - it stores its representative power
- Activation tensor is generated on the fly for every input
- **Can we decompose activations instead?**

GEMM with Weight Decomposition (e.g., truncated SVD)

$Y \approx UVX$

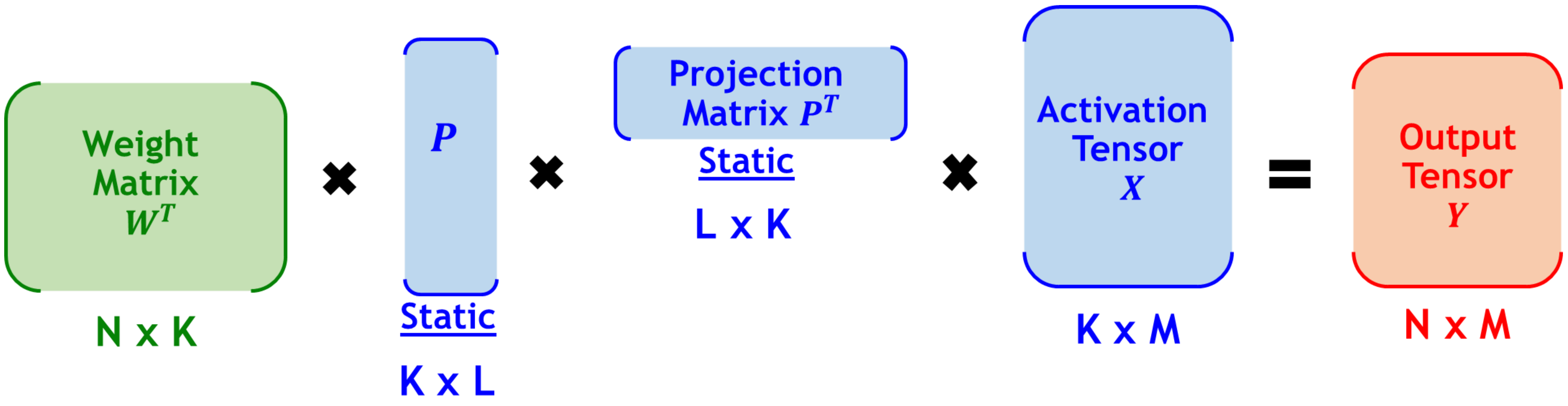
⬇️ trainable parameters: ⬇️ expressivity ❌
⬇️ model size: ⬇️ inference cost ✅



K = input dimension, N= output dimension, M = batch ∗ sequence
L = intermediate dimension

- Weight matrix can be approximated using factorization
  - e.g. a truncated SVD, where we hope to factorize $W^T \approx UV$
- To achieve compression, we need $L \ll K$
  - Since we now have two matrices instead of one
- Such decomposition has two limitations
  - fewer weights → lesser representative power
  - Breakage of weight structure → optimizer cannot be recovered

ESPACE: OUR PROPOSAL TO DECOMPOSE ACTIVATIONS



- Rather than computing: $Y = W^T X$
- let's insert a small matrix P and its transpose P^T
  - this *static and orthonormal* matrix will be related to the activation tensor X
  - It is likely that activation tensors have more redundancies as well (long sequences, repeated tokens, etc..)
- we will compute: $\hat{Y} = W^T P P^T X$

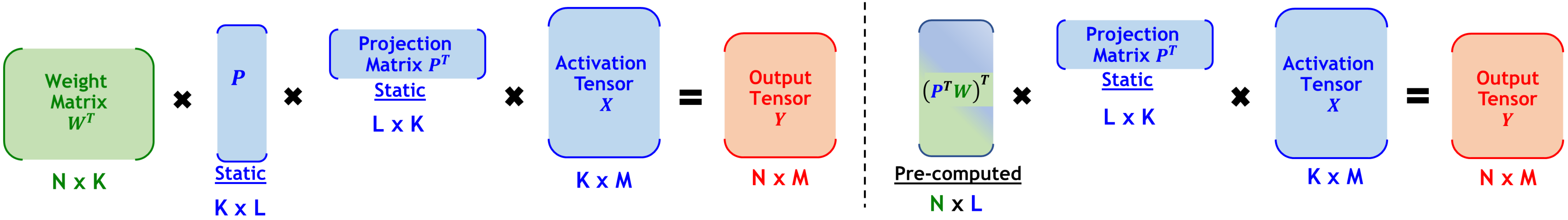
ESPACE: EIGEN STATIC PRINCIPAL ACTIVATION COMPONENT ESTIMATION

LLM compression via activation dimensionality reduction

$$Y \approx W^T(PP^TX) = (P^TW)^T(P^TX)$$

During Training: ⬆️ trainable parameters: ⬆️ expressivity ✅

During Inference: ⬇️ model size: ⬇️ inference cost ✅



K = input dimension, N= output dimension, M = batch ∗ sequence, L = intermediate dimension

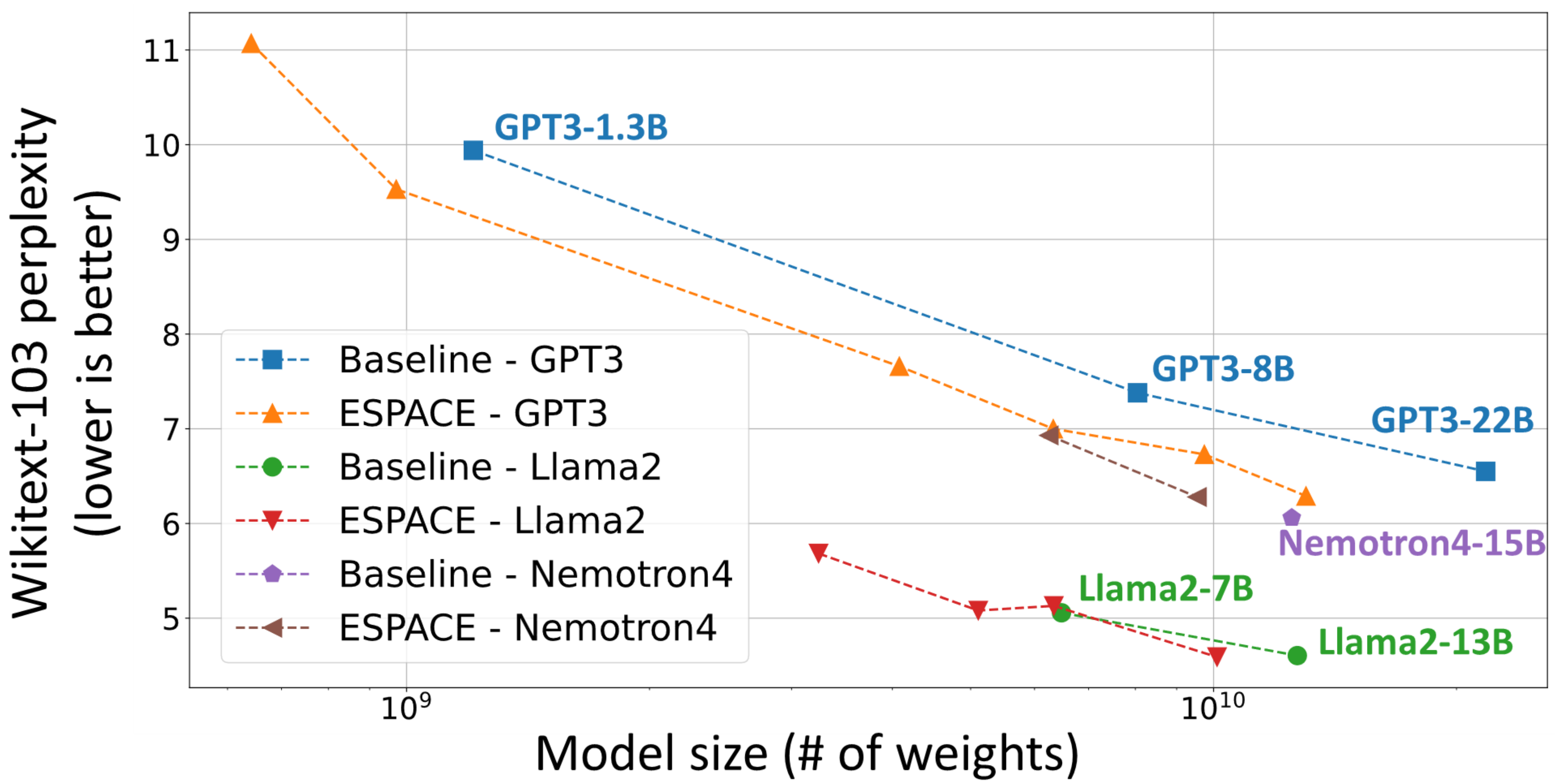
- We made an approximation $\hat{X} = PP^TX \approx X$ → will use continuous training to help model adapt to this approximation
- Activation projection does not interfere with weight learnability → the whole weight matrix W^T is preserved
  - we didn't reduce the number of learnable parameters and we can load the optimizer state
- At inference, this leads to compression when $L \ll K$
  - even during training, we end up computing less and observed a small end-to-end speed-up

ESPACE: EIGEN STATIC PRINCIPAL ACTIVATION COMPONENT ESTIMATION

Results and research roadmap

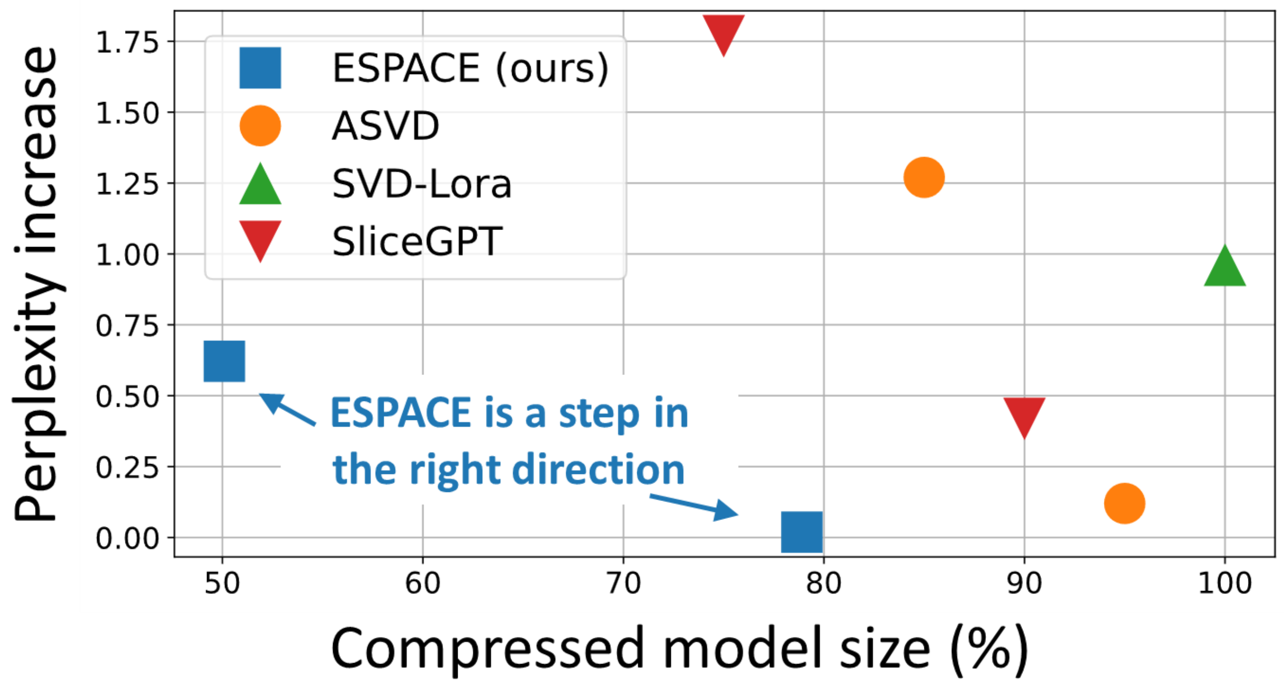
ESPACE strictly improves the pareto frontier of model size vs accuracy.
It also significantly improves on the SOTA of tensor decomposition method

LLM size vs accuracy



Comparison to Prior Work
Tensor Decompositions of Weights

Llama2-7B perplexity and compression



- Benefits during inference:
  - 20%-to-50% compression in # parameters
  - less weight*activation math: 35%-to-45% reduction in GEMM latency
- Future work and research roadmap
  - Combining ESPACE with other compression techniques, accelerate pre-training, optimize HW for back-to-back GEMMs

NVIDIA