

Table A5: Tensorization shape search space for the query, value, key, and projection weight matrices with three factorization methods.

Model	Candidate orders	Top-k Candidate shapes	Candidate ranks	Compression Ratio Limits	#Candidates
TTM	6, 8, 10	3	Multiple of 32	0.35~0.5	3235
Tucker	4, 6, 8	3	Multiple of 8	0.35~0.5	4754
CP	2, 3	3	Multiple of 32	0.35~0.5	44

## A1 MEMORY ACCESS TYPES FOR FUSED EINSUM

To implement fused einsum, we separate all nodes in the tensor contraction path  $p$  into 4 categories according to their memory access types, as shown in Fig. 5.

- Type 1: Load dynamic (D) input from DRAM and static (S) weights from the global buffer (GB), and write the intermediate tensor back to GB for data reuse.
- Type 2: Only read/write from/to GB without costly DRAM transaction.
- Type 3: Load both operands from GB and write the final results back to DRAM.
- Type 4: Load one dynamic input from DRAM and write the final results to DRAM.

Different node types are simulated with corresponding memory access constraints in Timeloop, so we can implement a fused einsum without unnecessary DRAM access.

## A2 HARDWARE COST SIMULATION SETTINGS

To construct the hardware cost table  $\mathcal{T}$ , we construct the tensorization shape candidate spaces in Table A5. For each shape, we collect all candidate ranks that are multiples of 32 or 8 and satisfy the compression ratio constraints. We use `Timeloop` with TSMC 5 nm energy model to simulate the fused einsum operation corresponding to each shape-rank pair. Our `Simba-L` architecture has a 2.91 MB global buffer and 32 PEs. Each PE has 32 32-KB weight buffers, one 64-KB input buffer, 32 384-B accumulation buffers, and 1024 8-bit MAC units. The hardware mapping objective of `Timeloop-mapper` is to minimize the energy-delay product. Based on the simulated hardware cost table  $\mathcal{T}$ , we use the `pareto` library to automatically select the Pareto optimal tensorization shape for the subsequent rank search flow.

## A3 RANK SUPERNET TRAINING SETTINGS

With the searched optimal tensorization shape  $s^*$ , we construct a Rank SuperNet with maximum ranks following a  $\sim 60\%$  target compression ratio. We use the original fine-tuned BERT-base as the teacher model and launch the 10-epoch logit distillation flow to train the Rank SuperNet. We use Adam optimizer with a learning rate of  $3e-5$  and a linear decay schedule. In the limited difference technique, we restrict the maximum allowed rank change across iterations to 3. We use a sandwich rule with one largest SubNet, one smallest SubNet, and two randomly sampled SubNets.

## A4 PER-TENSOR RANK SEARCH SETTINGS

With the trained Rank SuperNet, we uniformly sample 2560 SubNets with the largest and smallest SubNets and evaluate their validation F1 scores on a 5% validation set. We use 95% SubNet evaluation data to train a random forest ensemble model as the accuracy predictor. The model is an AdaBoostRegressor with 100 ExtraTreeRegressor, each tree regressor containing 60 decision trees with a maximum depth of 10.

In the evolutionary search stage, we use 200 populations with 40 parents, 80 mutations with 50% mutation probability, and 80 crossovers. After 100 steps, we obtain the optimal per-tensor rank settings.

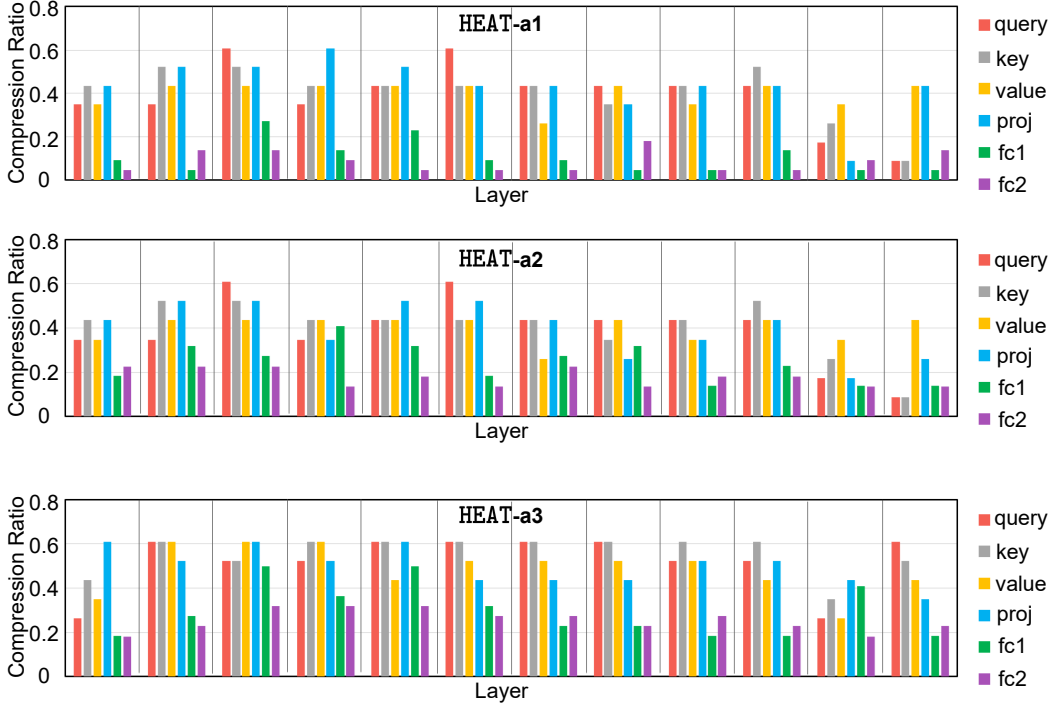


Figure A13: Compression ratio breakdown on each matrix in our HEAT-variants with TTM decomposition.

## A5 KNOWLEDGE DISTILLATION SETTINGS

We use a two-stage knowledge distillation flow to train the model with the searched ( $s^*$ ,  $r^*$ ) settings. We use the original BERT-base as the teacher model. We first launch an 8-epoch layer-wise distillation flow with attention and hidden state mapping with a constant learning rate of  $3e-5$  for TTM and Tucker,  $1e-5$  for CP. Then, we launch an 8-epoch logit distillation flow with an initial learning rate of  $1e-5$  on SQuAD-v1.1 and  $6e-6$  on SST-2 and a linear decay rate with 10% warm-up.

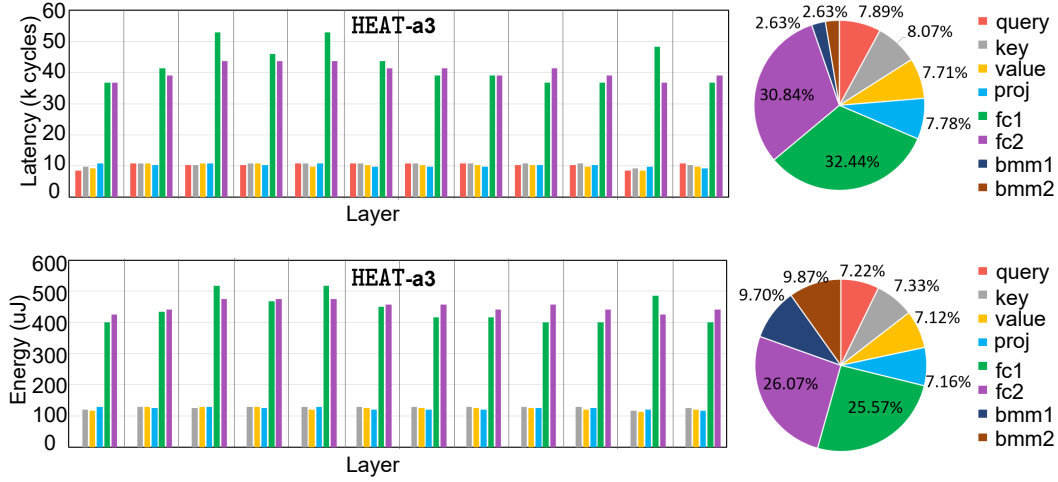
## A6 BREAKDOWN OF SEARCHED HEAT VARIANTS

**Compression Ratio.** We plot the compression ratio breakdown of our searched HEAT-variants in Fig. A13. We can observe that deeper layers tend to have higher redundancy and thus have fewer parameters. Feedforward networks tend to have a lower compression ratio (fewer parameters) than query/value/key matrices.

**Latency and Energy.** In Fig. A14, the fully-connected (FC) layers in FFNs have lower compression ratios but nearly  $4\times$  higher latency than other linear layers. The batched matrix multiplication (BMM) in attention operations, i.e.,  $\mathbf{QK}^T$  and  $\mathbf{AV}^T$ , only take around 5.3% total latency and 19.5% total energy in the entire network, which validates that the most costly operations in Transformer are indeed linear layers.

## A7 DECOMPOSITION ON EMBEDDING LAYERS

Some prior work applies low-rank decomposition on the embedding layer in Transformer models and claims it can save parameters. However, when off-chip DRAM capacity is not a limiting factor, this embedding layer decomposition comes with a non-trivial accuracy drop and no hardware efficiency benefits. Indexing the original look-up table only contains DRAM read without extra computations. In contrast, indexing factorized tensors is very costly and requires tensor dot-product among all

Figure A14: Latency (*Top*) and energy (*Bottom*) breakdown of HEAT-a3.

decomposed core tensors. The computation overhead far outweighs the saved storage capacity. Therefore, we do not decompose the embedding layers.