ICDM 2024

ELiCiT: Effective and Lightweight Lossy Compression of Tensors

Jihoon Ko

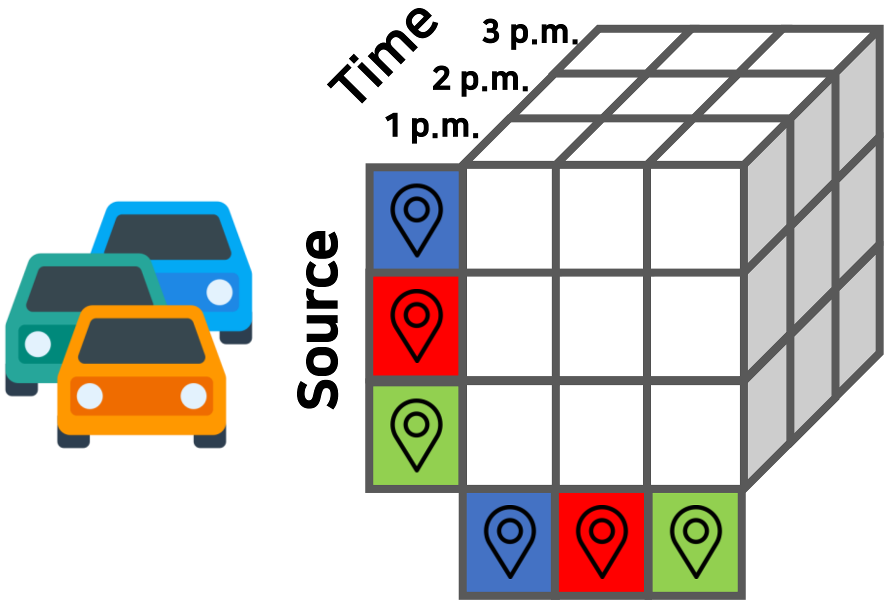
Taehyung Kwon

Jinhong Jung

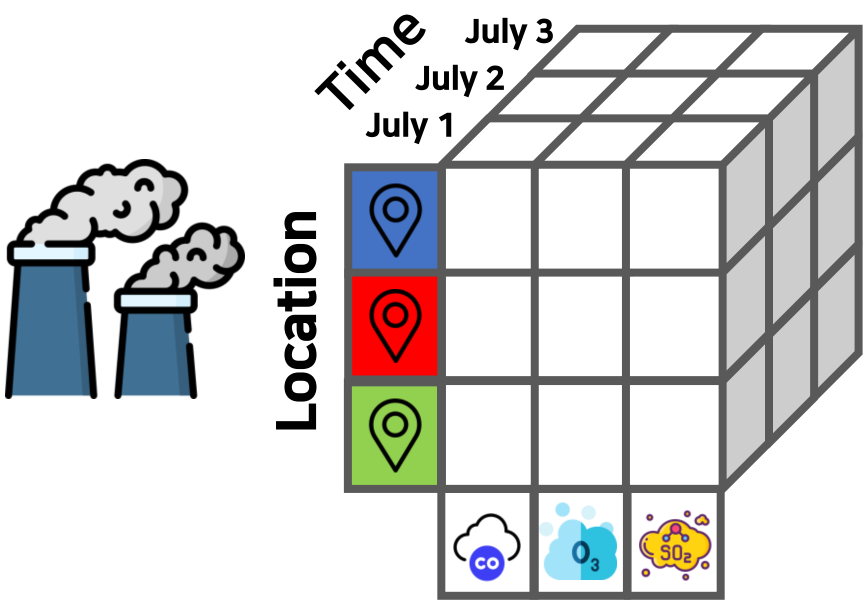
Kijung Shin

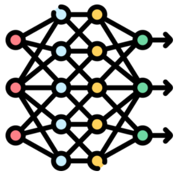
Large-scale Tensors are Everywhere!

Traffic volumes



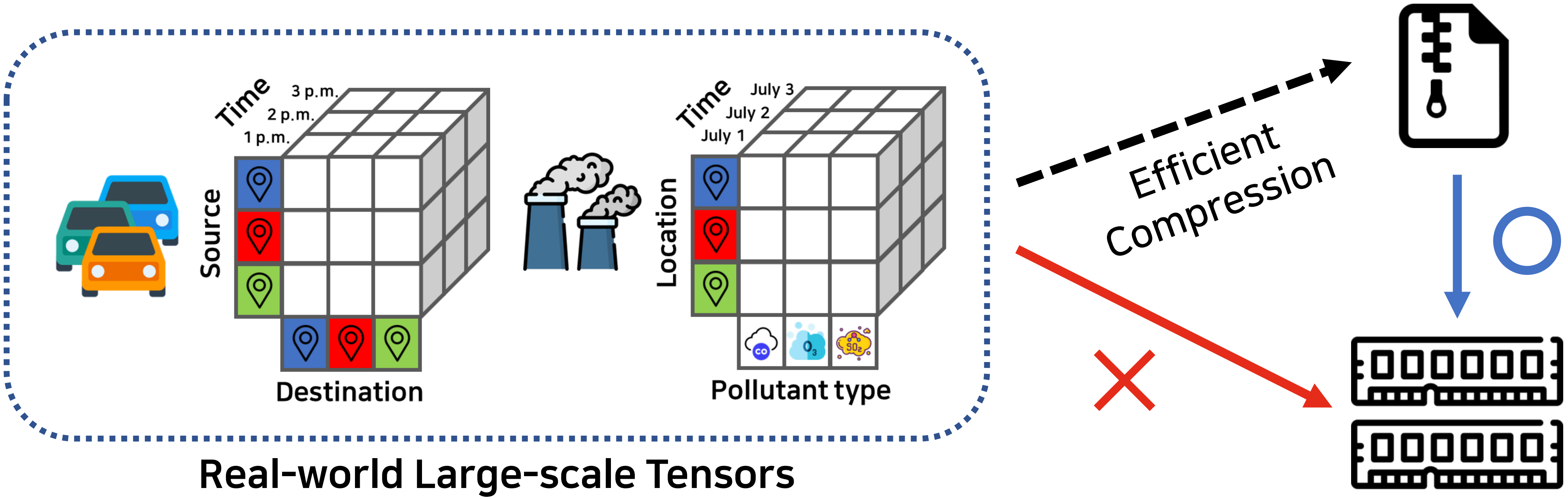
Air pollutant measurements



Images  Videos  Neural-network parameters 

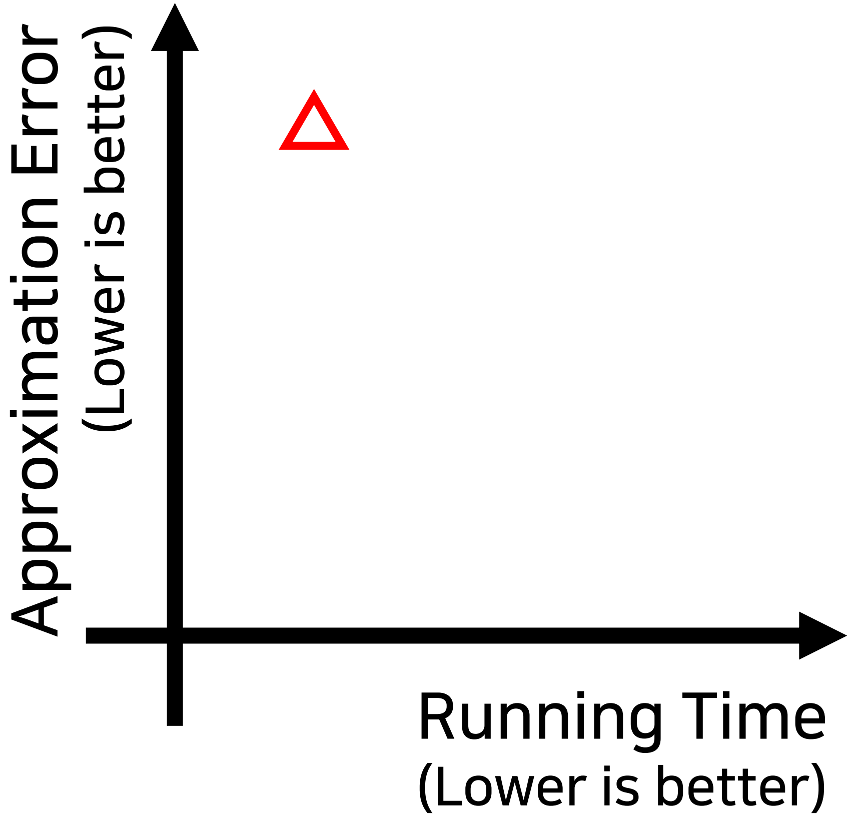
Why Compression is Important?

- Handling large-scale tensors as they are...
  - requires heavy memory or network I/O usage
- Their compact representation reduces storage and communication costs



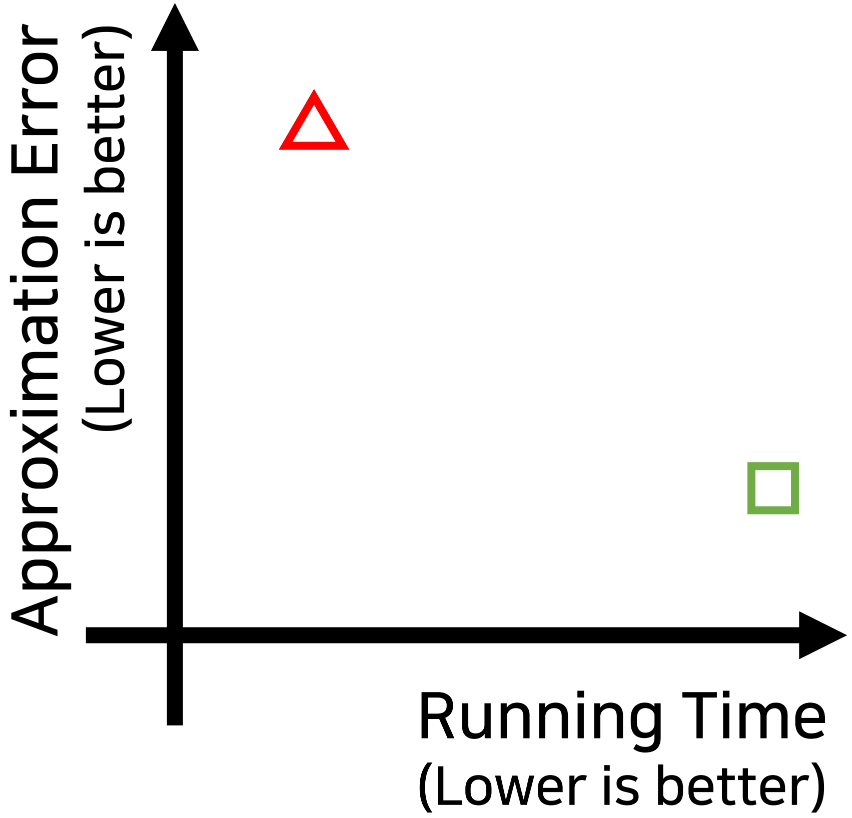
Generic Tensor Compression

- Methods that do not rely on specific data property assumptions are important
  - ex) Video compression methods, which heavily rely on continuity, are not applicable to tensors in other contexts
- Decomposition-based Methods
  - CP [Caroll et al, 1970], Tucker [Tucker et al., 1966]
  - Breaks down tensors into smaller components
  - Has also been used for other applications
    - ex) tensor completion, neural-network compression



Generic Tensor Compression

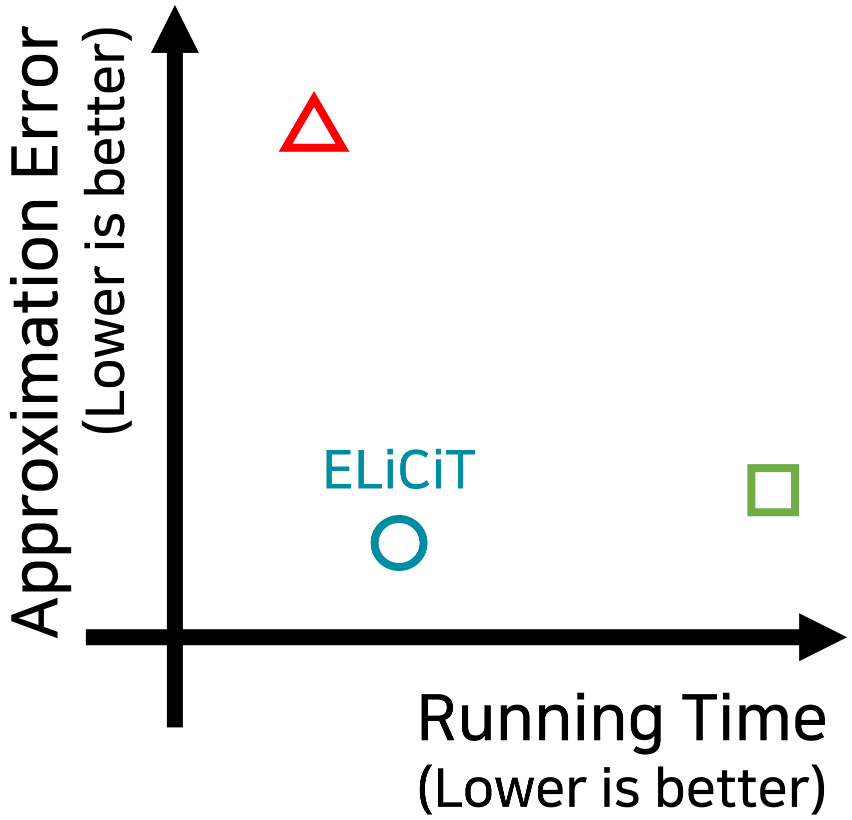
- Methods that do not rely on specific data property assumptions are important
  - ex) Video compression methods, which heavily rely on continuity, are not applicable to tensors in other contexts
- Decomposition-based Methods
  - CP [Caroll et al, 1970], Tucker [Tucker et al., 1966]
  - Breaks down tensors into smaller components
  - Has also been used for other applications
    - ex) tensor completion, neural-network compression
- Deep-learning-based methods
  - Generalizes decomposition-based methods with neural networks
  - Extremely slower than decomposition-based methods due to heavy computational cost



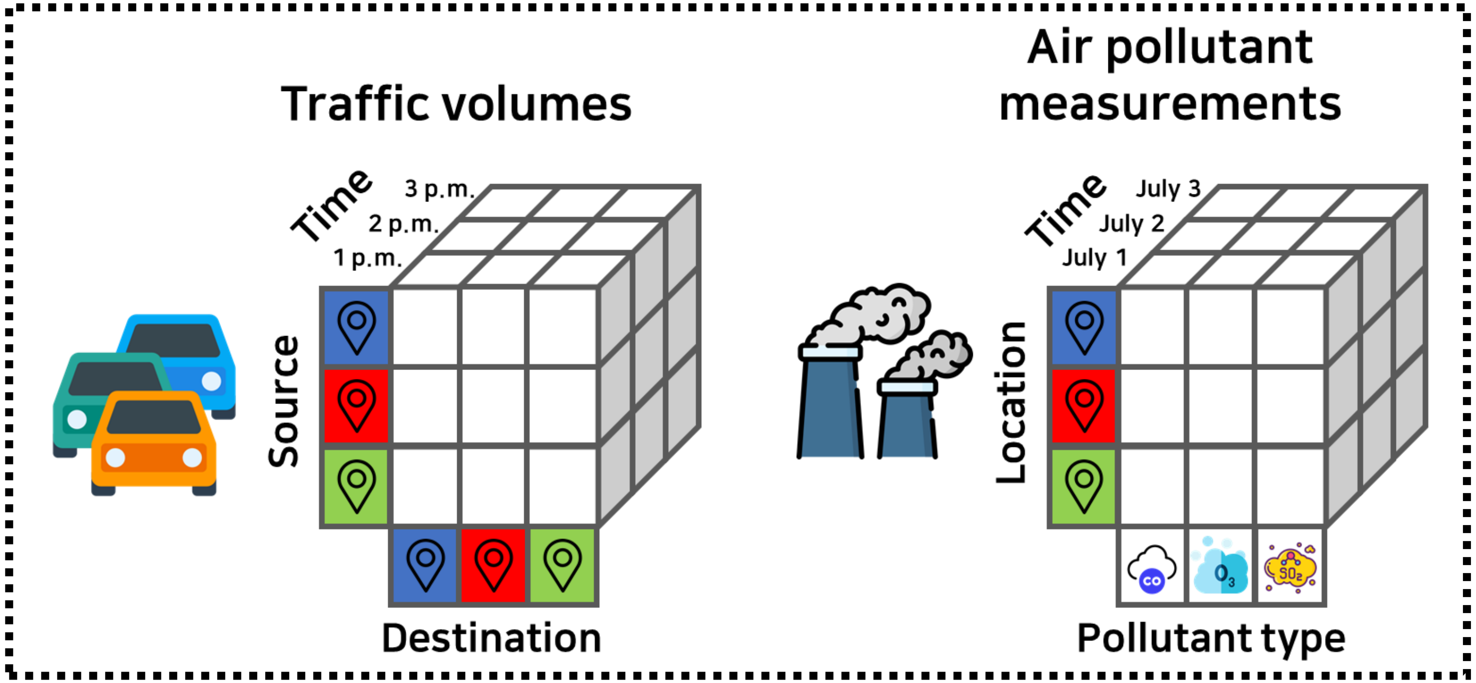
ELiCiT: Our Proposed Method

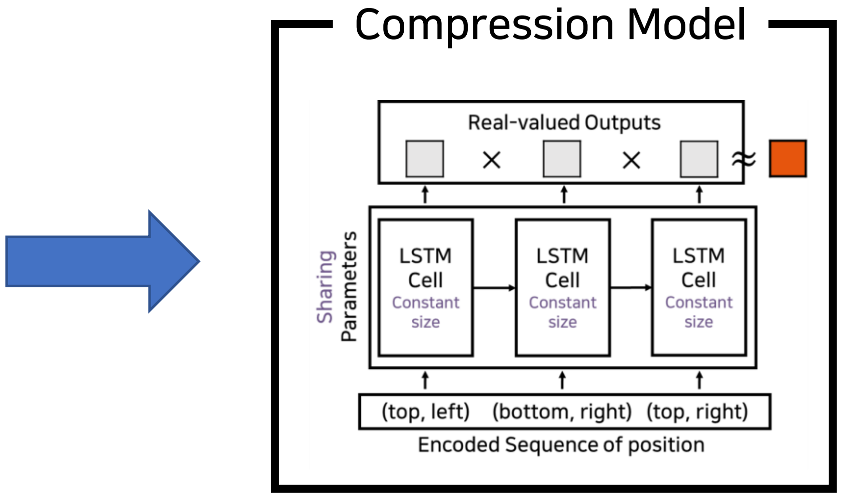
- **Question:** Can we accelerate tensor compression while maintaining or even improving compression performance?

- Our solution: ELiCiT (Effective Lightweight Compression of Tensors)
  - Significantly reduces computational costs
  - (Partially) generalizes decomposition-based methods and deep-learning-based methods
  - Can be extended to various applications



Problem Definition



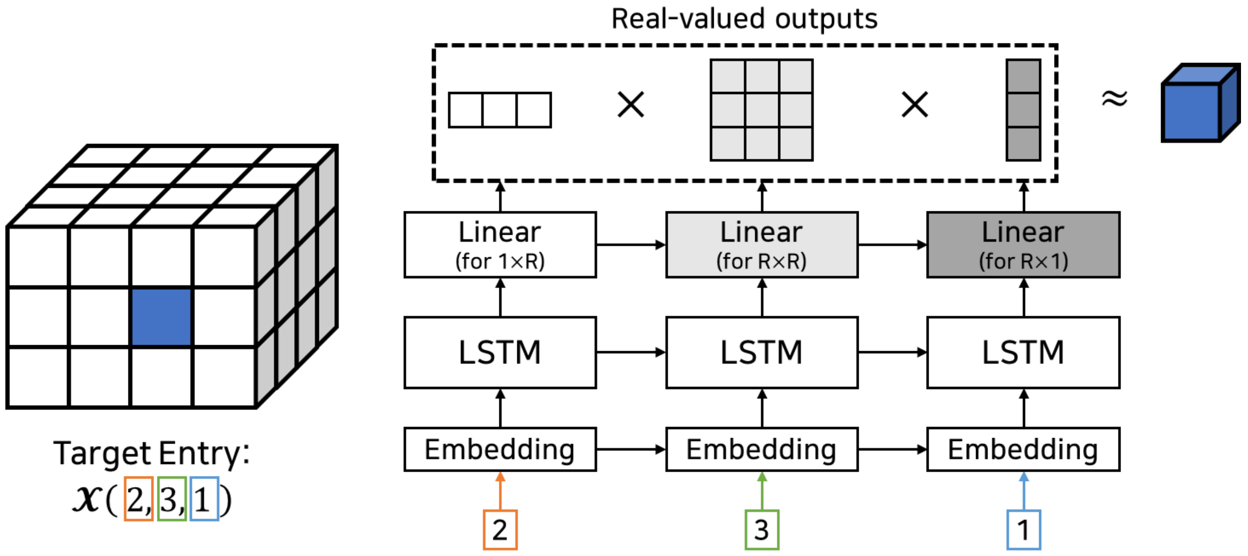


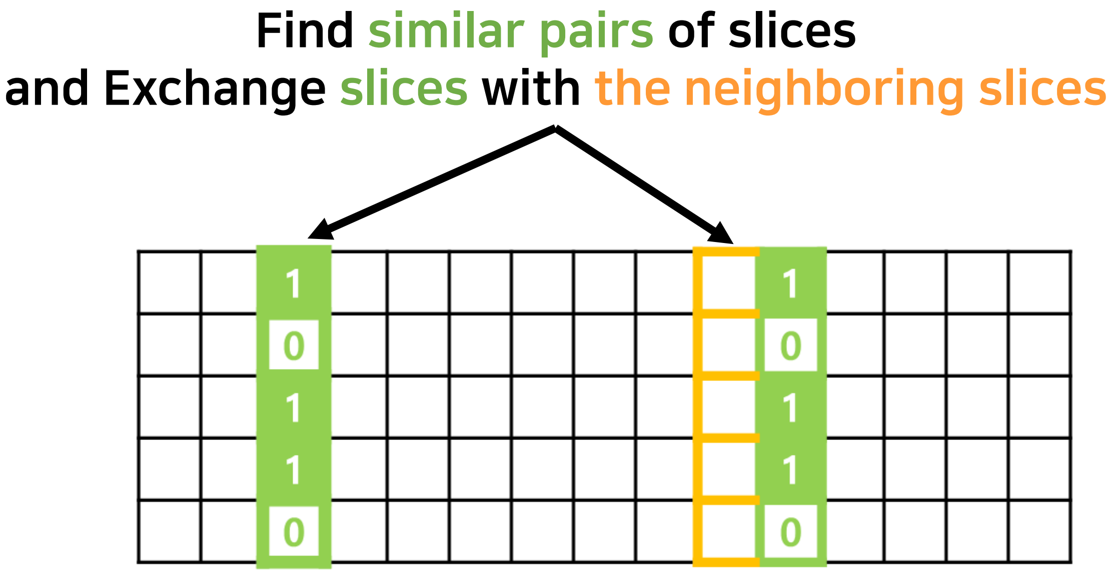
- **Given:** A tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_d}$
- **Find:** A fitting model $\boldsymbol{\Theta}$
- **To minimize: (1)** The total size of parameters of $\boldsymbol{\Theta}$

  (2) The approximation error $\|\mathcal{X} - \tilde{\mathcal{X}}_{\boldsymbol{\Theta}}\|_F^2$

Existing Deep-learning-based Techniques

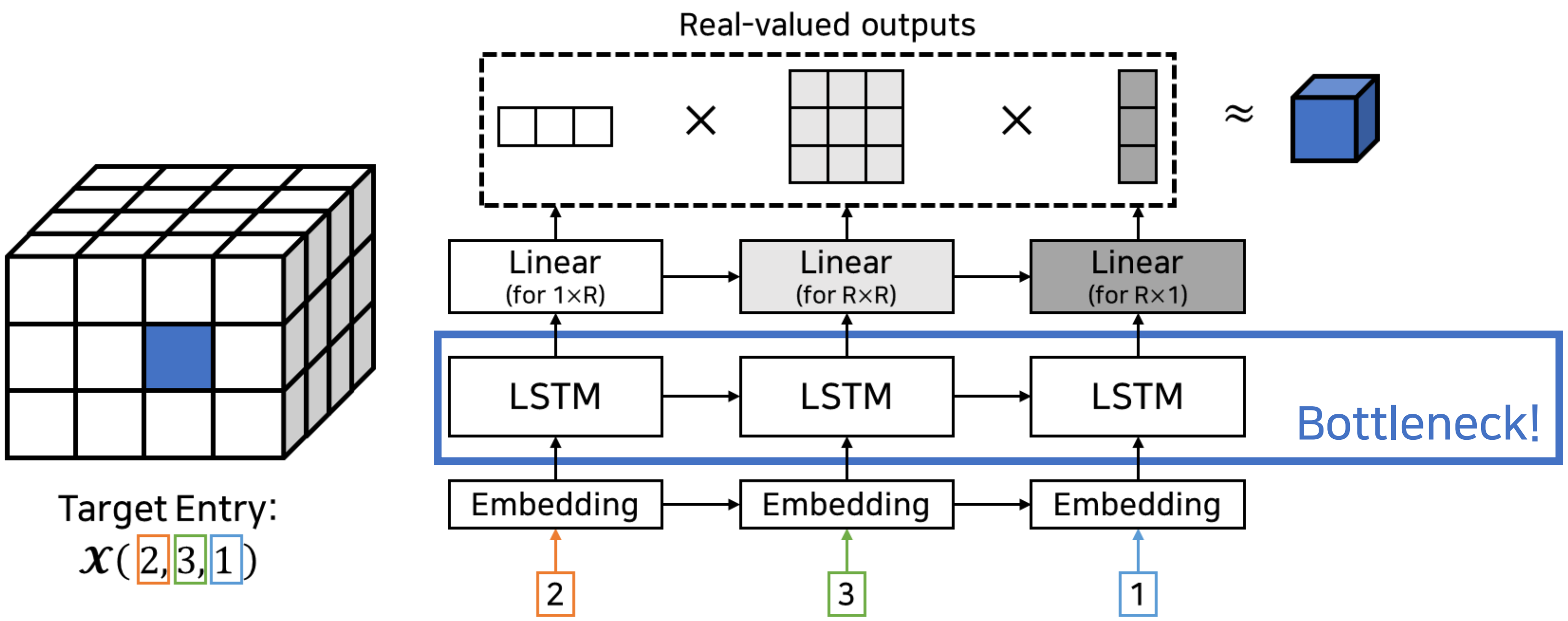
- NeuKron and TensorCodec [Kwon et al., 2023]
  - Generalizes the Kronecker model and tensor-train decomposition using LSTM
  - Proposes order optimization method for tensors to exploit meaningful patterns
  - Outperforms decomposition-based method





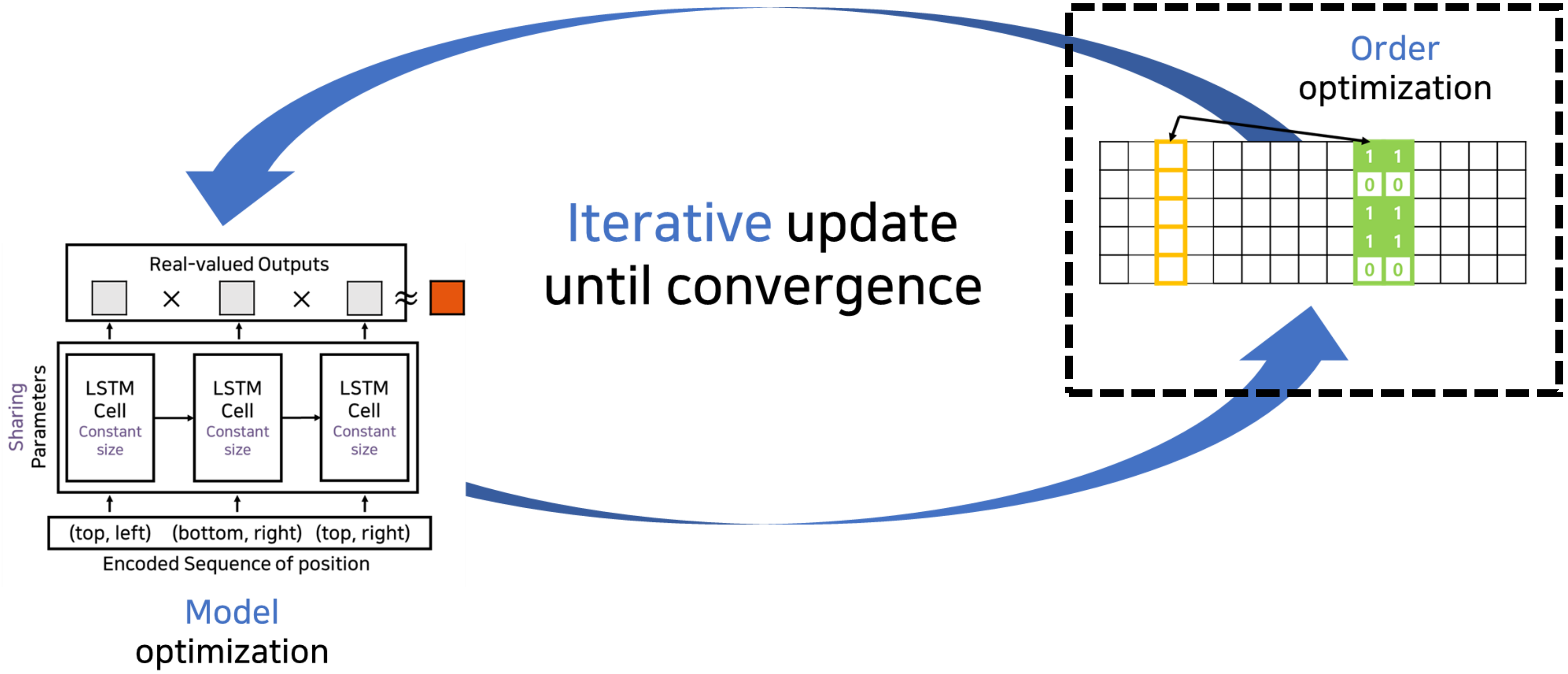
Limitations of Deep-learning-based Methods

- While these methods exhibit exceptional compression performance...
- Limitation 1: High Computational Cost of Neural networks
  - Introduces a significant computational burden during training
  - The sequential design imposes limitations on parallelism



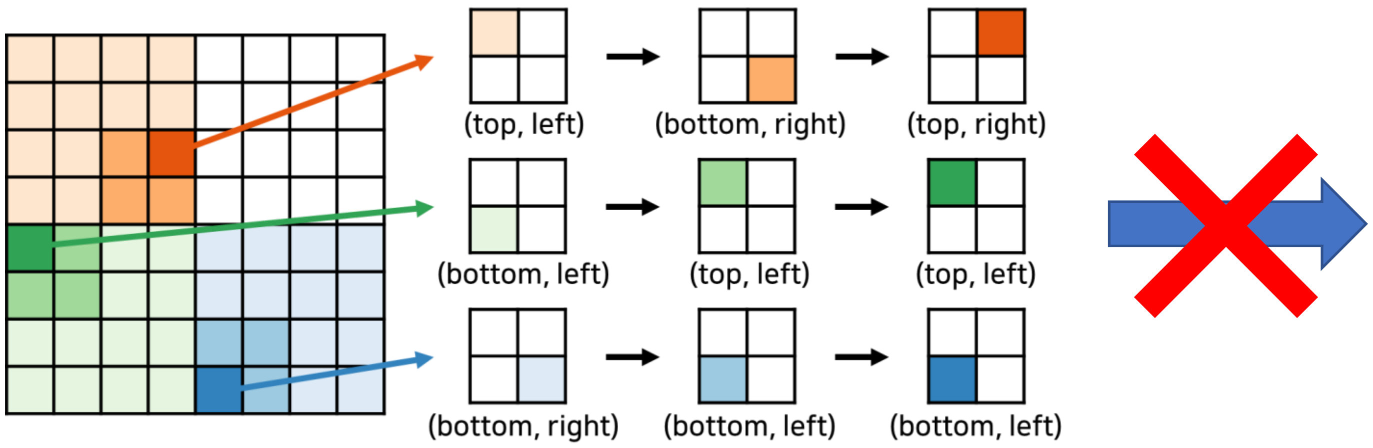
Limitations of Deep-learning-based Methods

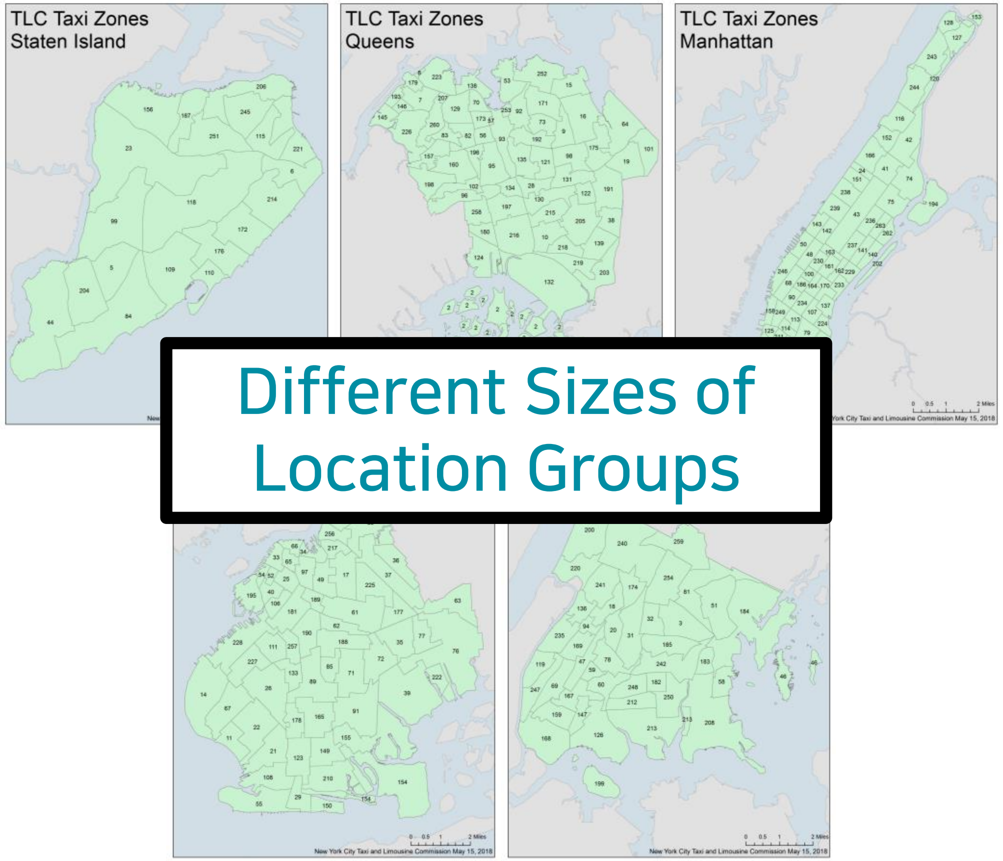
- Limitation 2: High Computational Cost of Order Optimization
  - The orders cannot be optimized concurrently with other parameters
  - Consumes a substantial portion of the total compression time



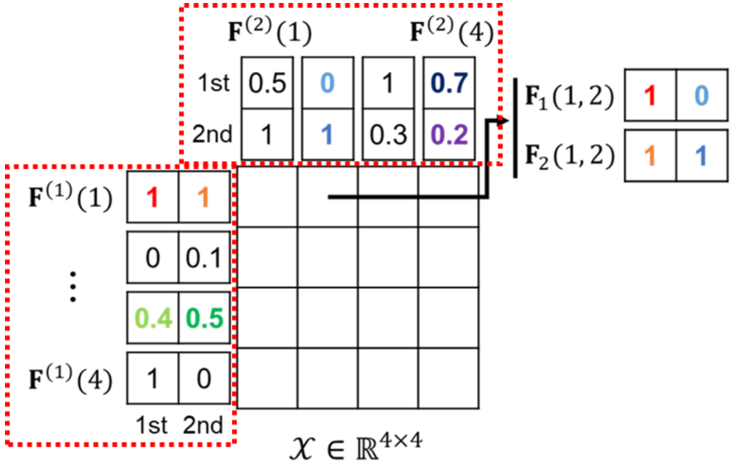
Limitations of Deep-learning-based Methods

- Limitation 3: Limited Expressiveness of Order-based Models
  - Recursively dividing the mode indices into equal-sized groups can be limited



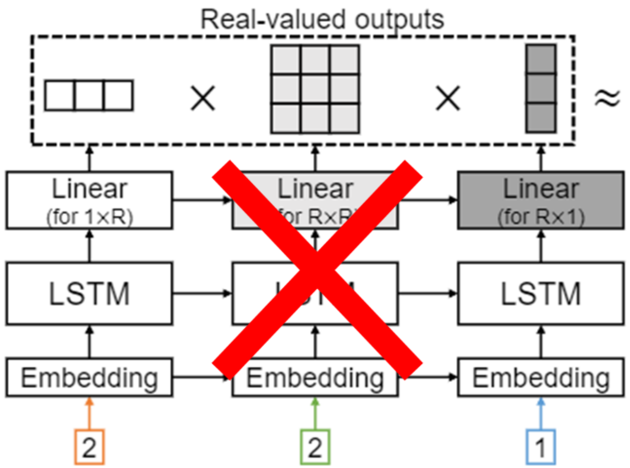


Key Idea of ELiCiT



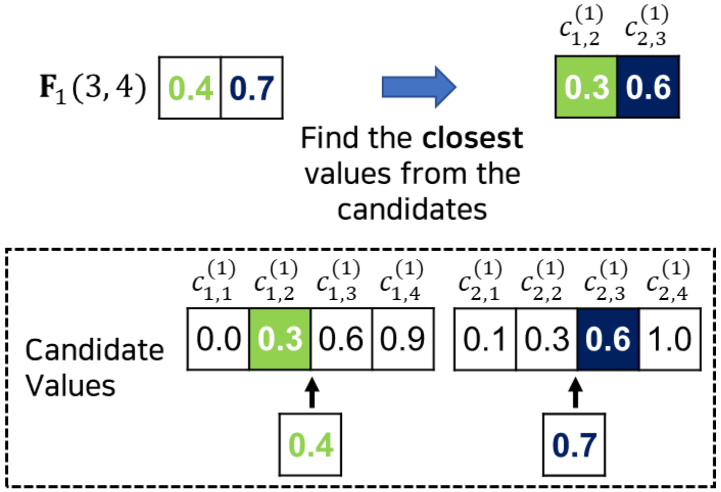
Feature-based Index Model

- More expressive index model
- Eliminates the inefficient order optimization process



Lightweight Approximation Process

- Avoids the use of computationally expensive deep neural networks (spec. LSTM)

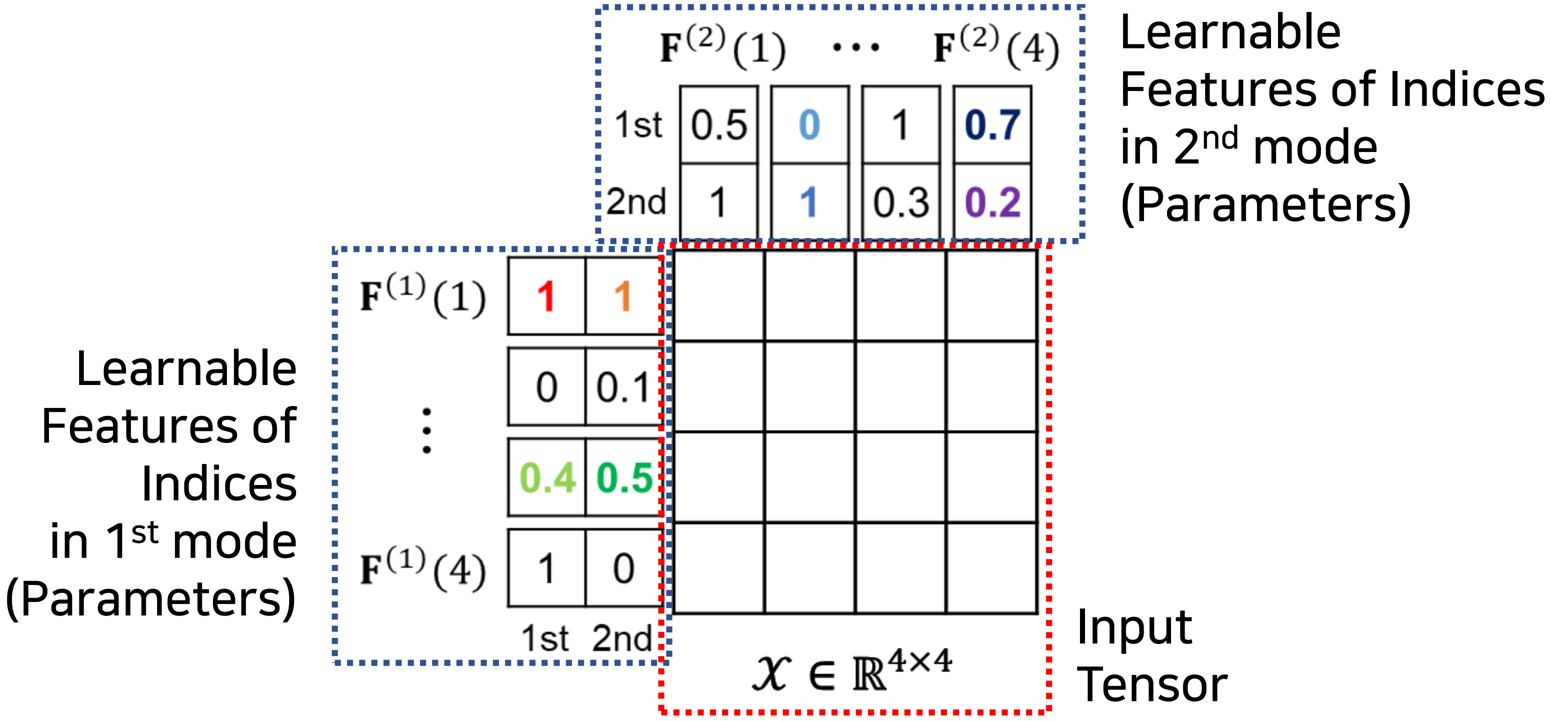


Clustering-based Quantization

- Reduces the size of features for more efficient compression

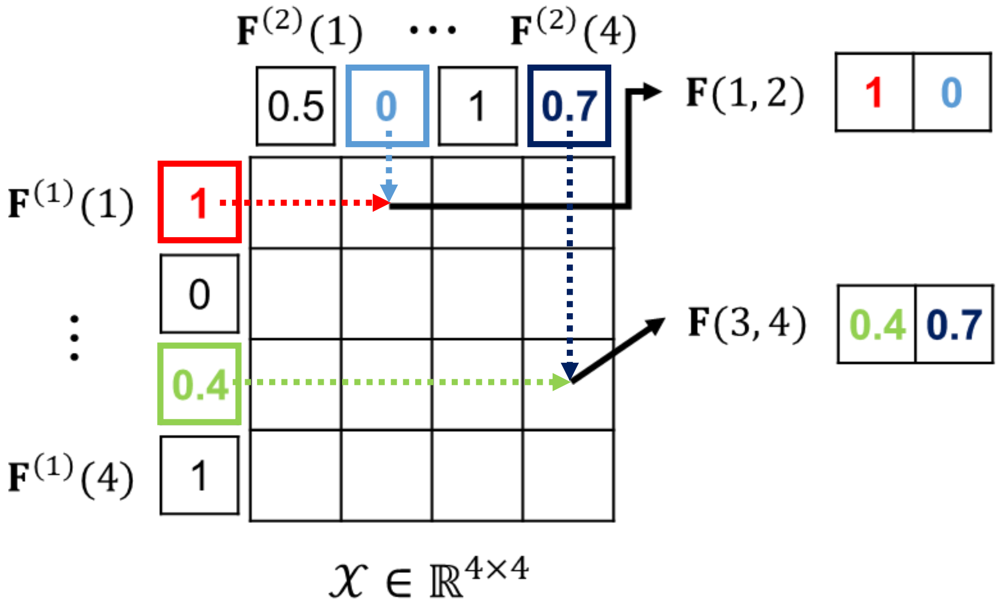
Key Idea (1) – Feature-based Index Model

- Feature-based Index Model: ELiCiT uses r-dimensional continuous learnable features to model each index as parameters



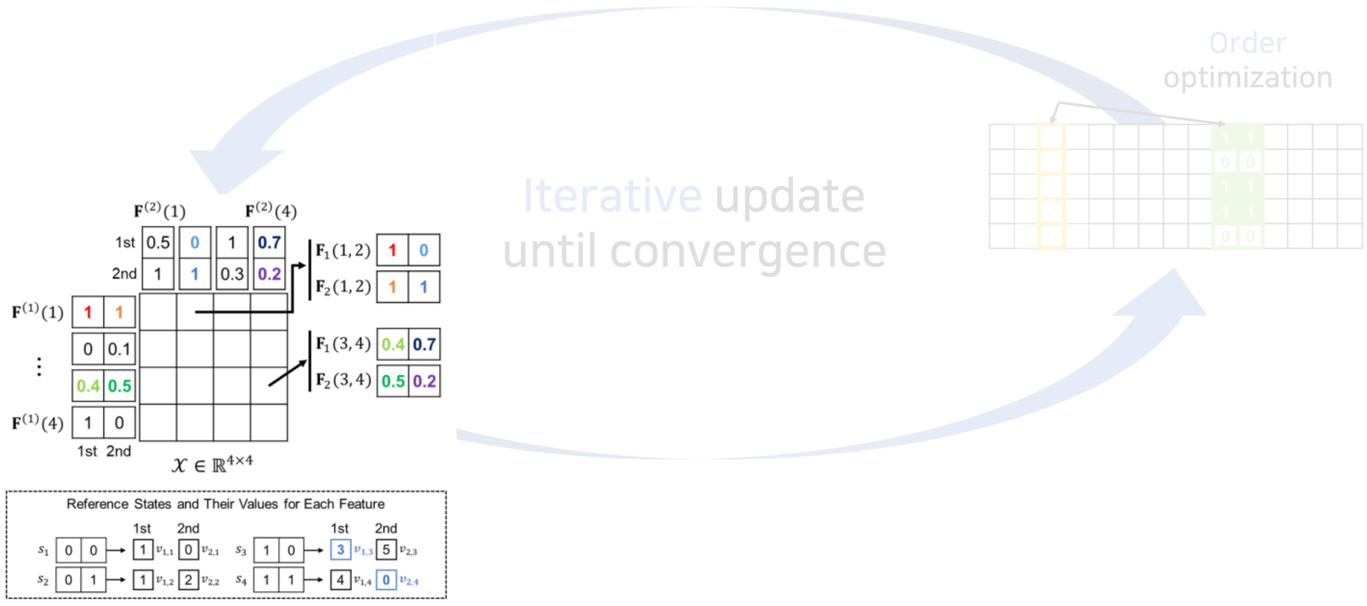
Key Idea (1) – Feature-based Index Model

- Feature-based Index Model: ELiCiT uses r-dimensional continuous learnable features to model each index as parameters
- The features of entries are determined based on the features of indices



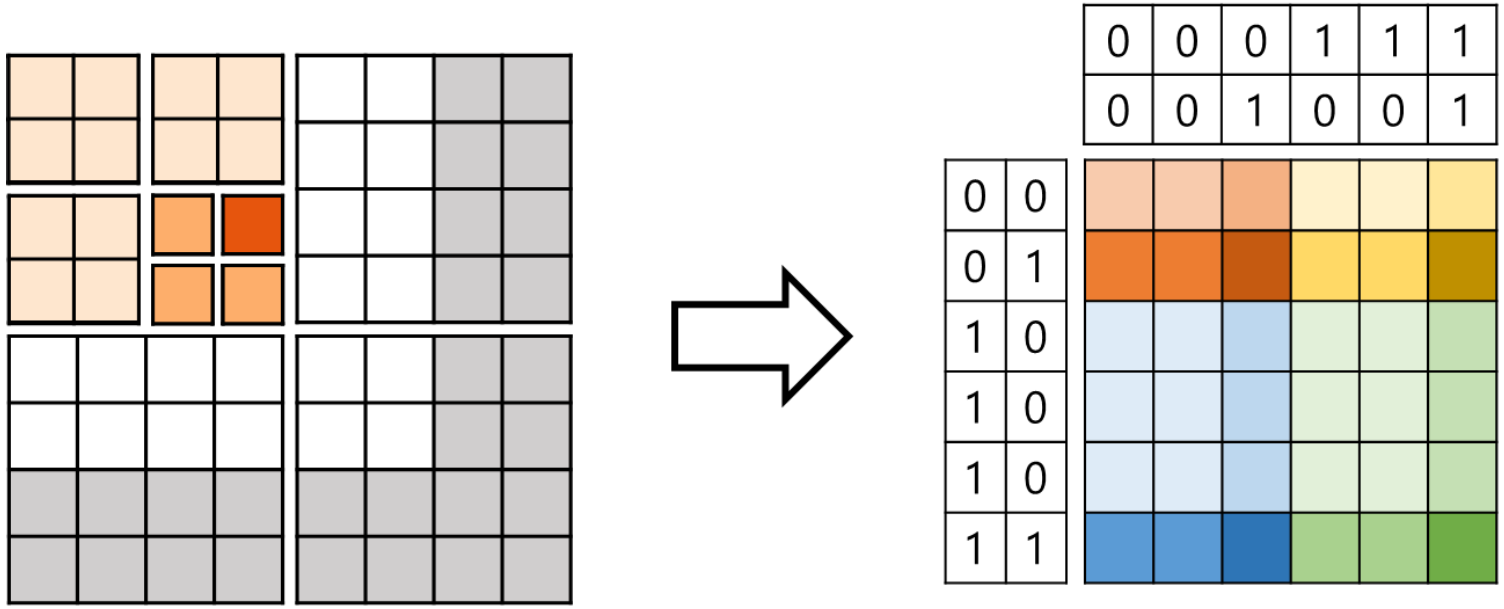
Key Idea (1) – Feature-based Index Model

- Limitation: High Computational Cost of Order Optimization
- Advantage: End-to-end Training
  - Eliminates the order optimization process and supports gradient descent-based update
  - Significantly reduces the required training time



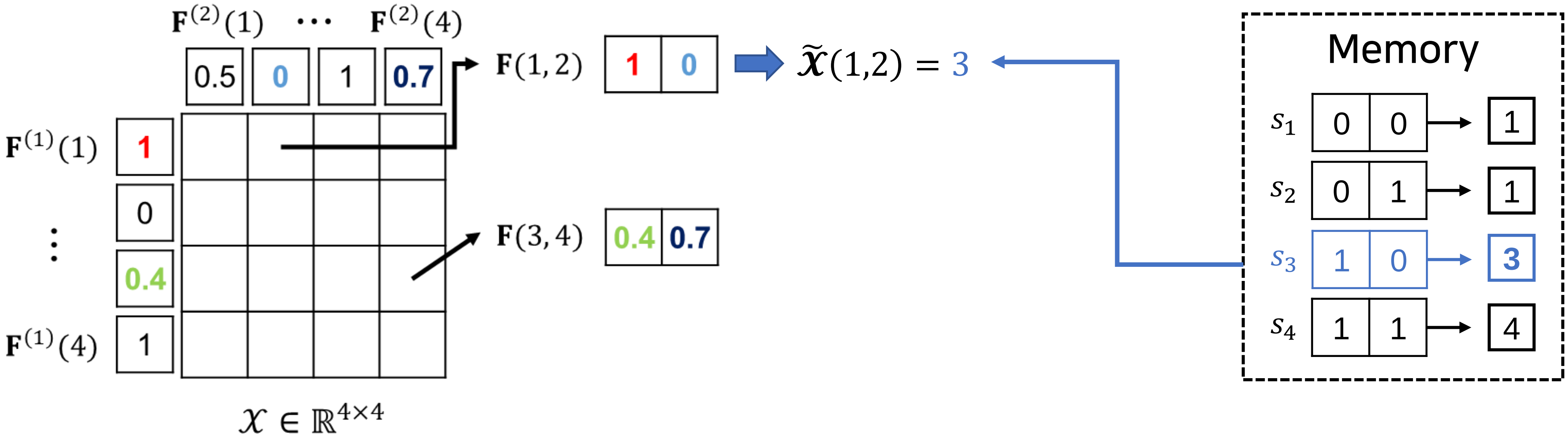
Key Idea (1) – Feature-based Index Model

- Limitation: Limited Expressiveness of Order-Dependent Models
- Solution: Feature-based Index Model
  - Generalizes the order-based models
  - Naturally interprets groups of varying numbers and sizes



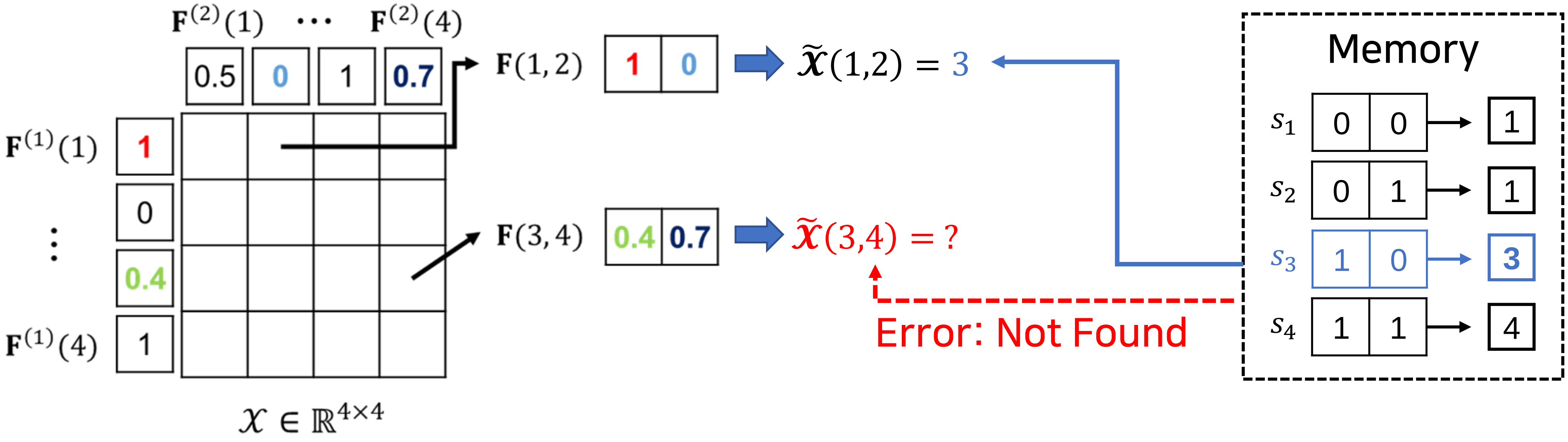
Key Idea (2) – Approximation Process

- **Simple case (1):** Each feature of index is one-dimensional binary feature
- 2^d binary reference states (d: order of tensor)
  - Each reference state s_i has its corresponding value v_i
- Finds the reference state identical to the feature vector
- Retrieves the corresponding value



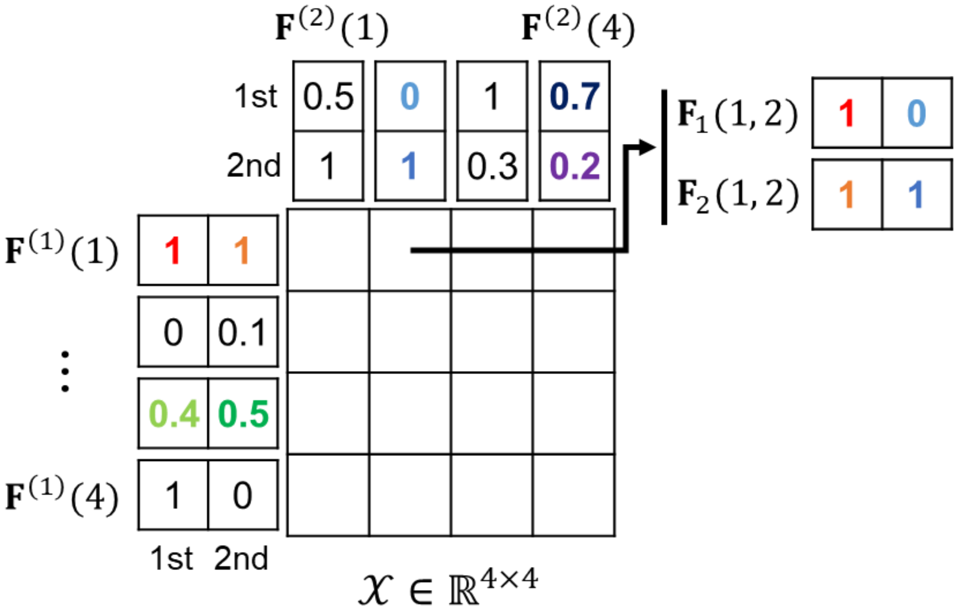
Key Idea (2) – Approximation Process

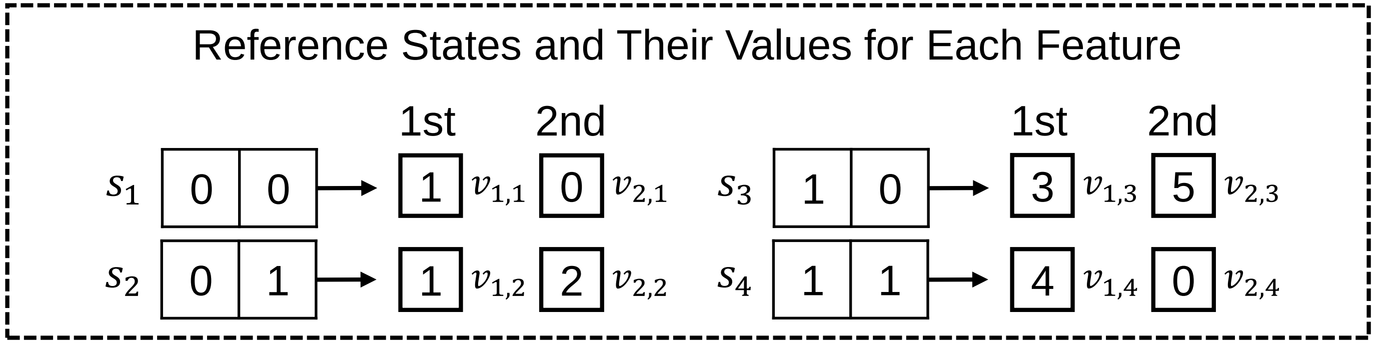
- **Simple case (2):** Each feature of index is one-dimensional continuous feature
- 2^d reference states
  - Each reference state s_i has its corresponding value v_i
- Finds the reference state identical to the feature vector <- impossible!
- Use weighted sum instead of matching the feature vectors



Key Idea (2) – Approximation Process

- Each feature of index is multi-dimensional continuous feature
- For each k-th feature, each reference state s_i has its corresponding value $v_{k,i}$
- Uses differentiable reduce function g to determine the final output





$$\tilde{\mathcal{X}}(1,2) = g(\boxed{3}\, v_{1,3},\ \boxed{0}\, v_{2,4})$$

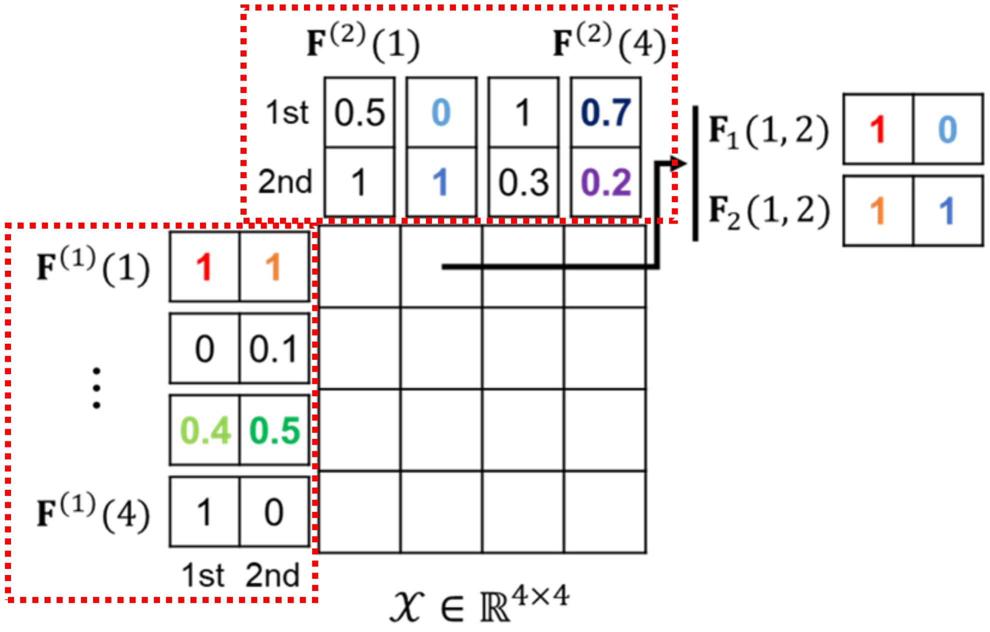
$$g(x_1, x_2, \cdots, x_r) = \left(\sum_{k=1}^{\lfloor r/2 \rfloor} x_k\right) \cdot \tanh\left(\sum_{k=\lfloor r/2 \rfloor+1}^{r} x_k\right).$$

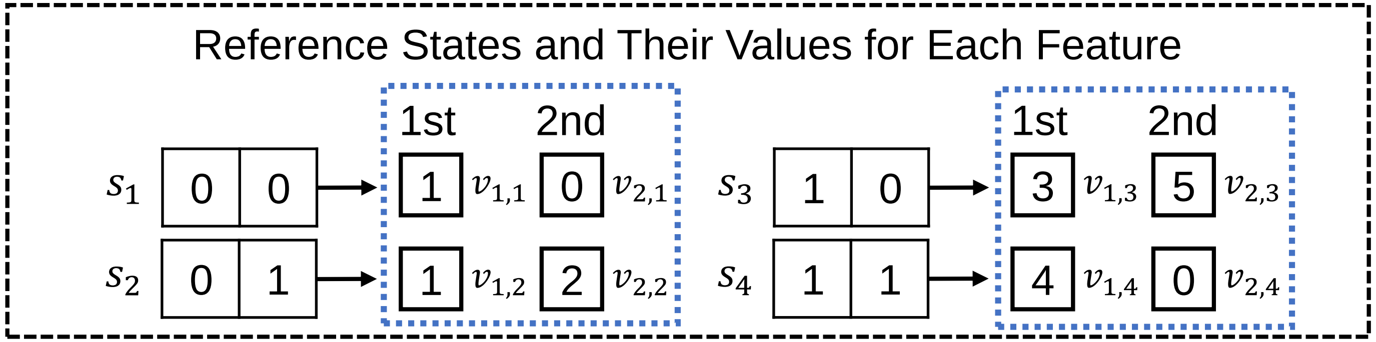
Key Idea (3) – Clustering-based Quantization

- Compressed output size of ELiCiT

(size of features) >> (size of the corresponding values of the reference states)

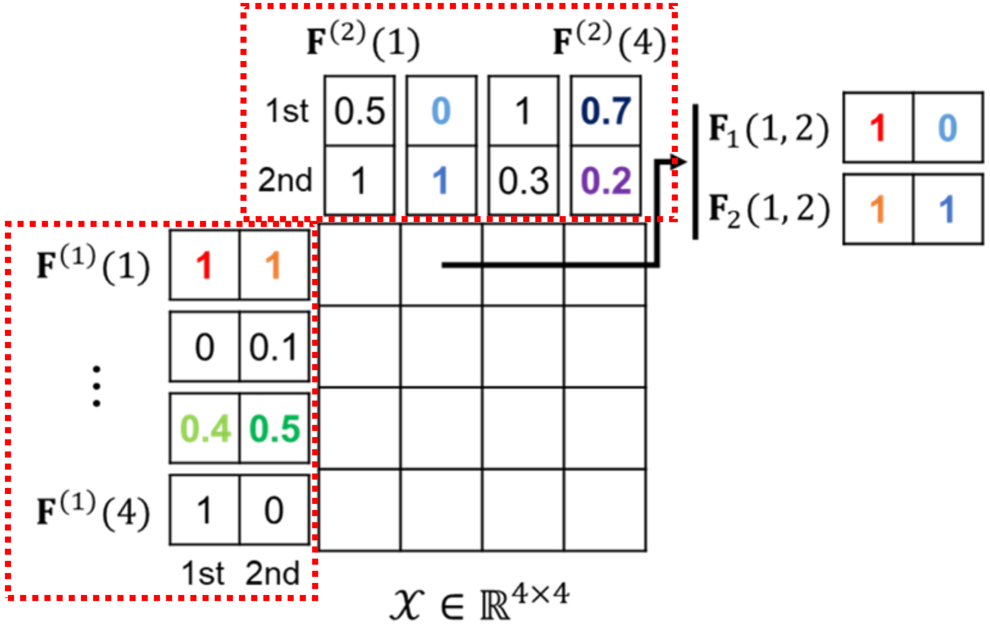
$\propto$ #mode indices $\propto 2^d$ (d: order of tensor)

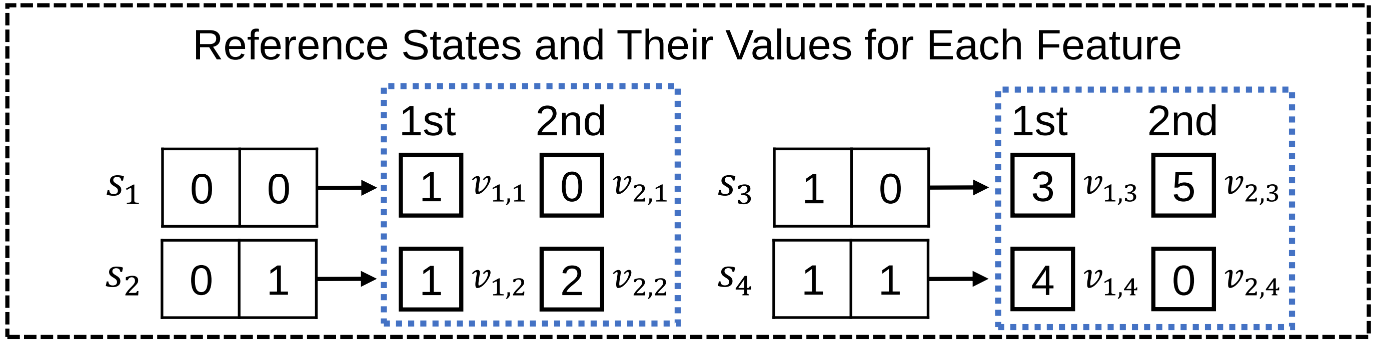


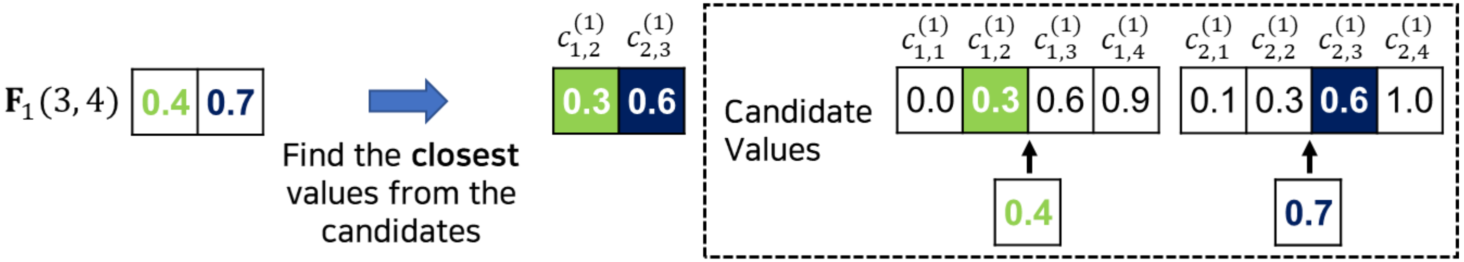


Key Idea (3) – Clustering-based Quantization

- Compressed output size of ELiCiT
  - = (size of features) + (size of the corresponding values of the reference states)
- **Solution - qELiCiT:** Utilizes clustering-based quantization to reduce the size of features -> q + o(1) bit for each feature, where q << 64

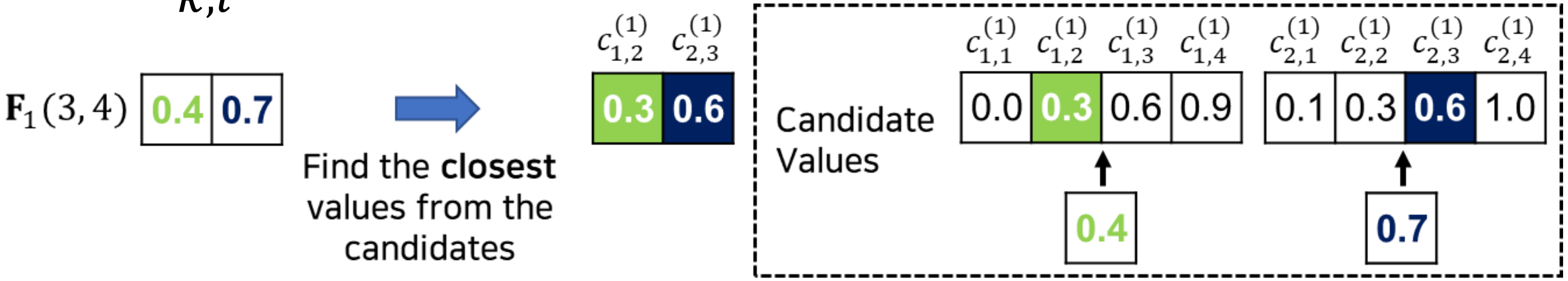






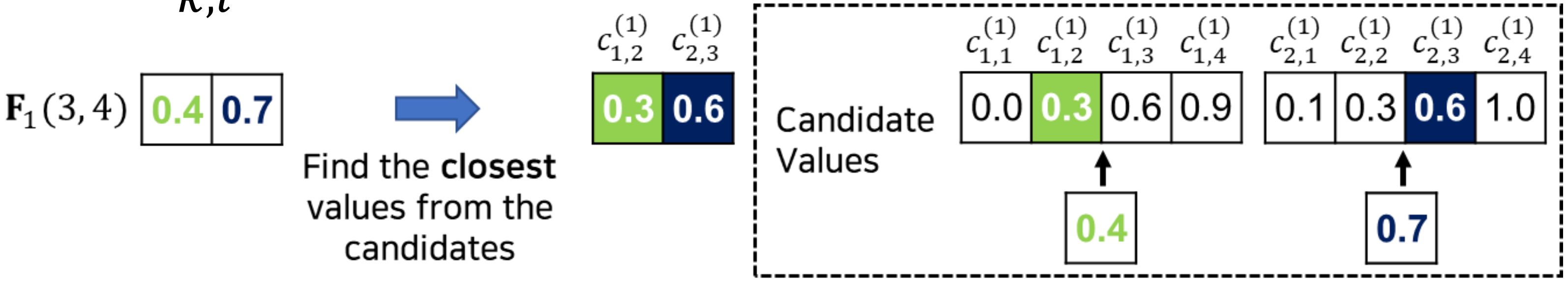
Details Comparison with K-means Clustering

- Prepares 2^q candidates $c_{k,l}^{(j)}$ for each mode index j and feature index k and finds the closest $c_{k,l}^{(j)}$



Details Comparison with K-means Clustering

- Prepares 2^q candidates $c_{k,l}^{(j)}$ for each mode index j and feature index k and finds the closest $c_{k,l}^{(j)}$



- Commonalities: Similar to 1D K-means clustering
  - There are multiple candidate values, and the closest one is selected
  - Candidates and features are corresponded to centroids and samples
- Differences
  - The features also should be updated
  - Our quantization method is trained in an end-to-end manner with an objective designed to optimize compression performance

Details Extension to Other Real-world Applications

- **Matrix Completion**
  - Goal: To predict the missing entries in the input matrix accurately
  - Proposed Method: qELiCiT++
    - Modifies SVD++ [Koren, 2008] by replacing low-rank decomposition with qELiCiT

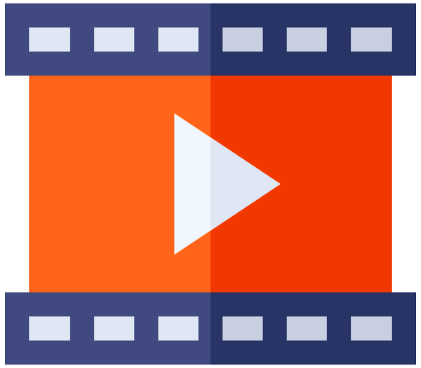
- **Neural-network Compression**
  - Goal: To minimize the number of parameters of a neural-network model while minimizing the degradation in the performance of the compressed model
  - Proposed Method: TFW-qELiCiT
    - Modifies TFWSVD [Hua et al., 2022] by replacing SVD with qELiCiT

Experimental Setting

- Datasets: 8 real-world tensors (up to 200M entries)

Air quality measurement

Traffic volume



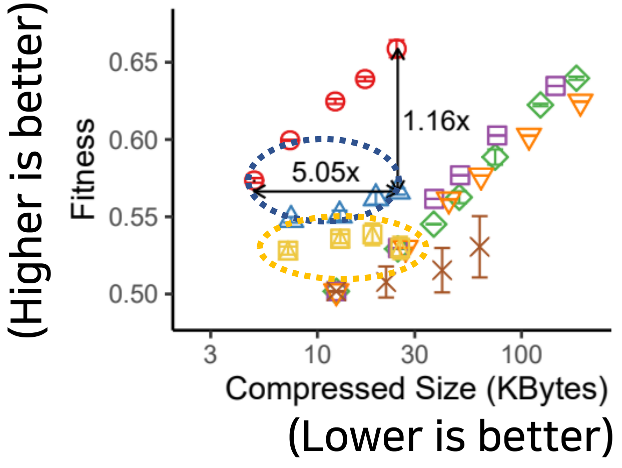
Video feature

Stock datum

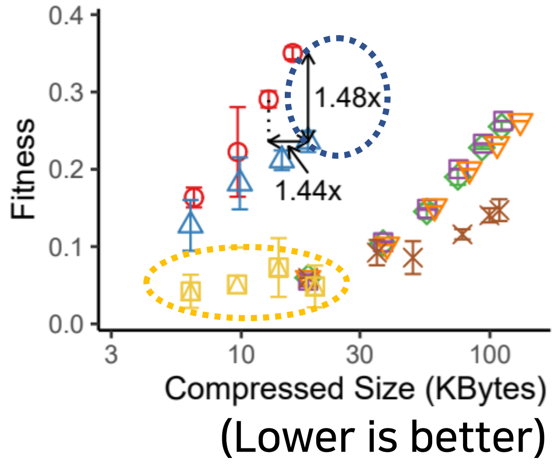
- Baseline Methods:
    - Decomposition-based : CPD, TKD, TTD, TRD
    - Deep-learning-based: NeuKron, TensorCodec

ELiCiT is Compact and Accurate

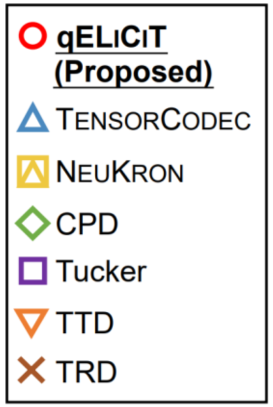
- qELiCiT provides compact and accurate compression of tensors
  - Up to 5.05× smaller compression size and 48% higher accuracy

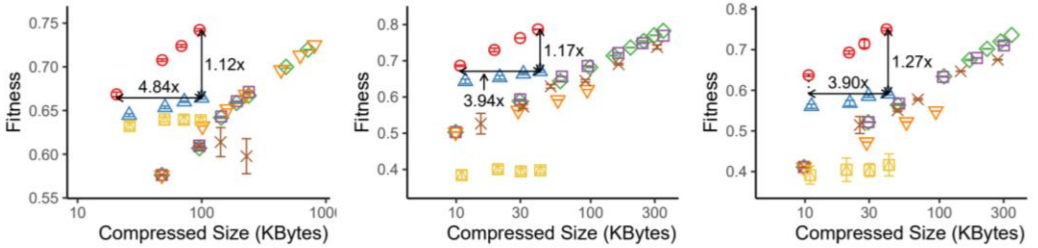


Dataset: PEMS

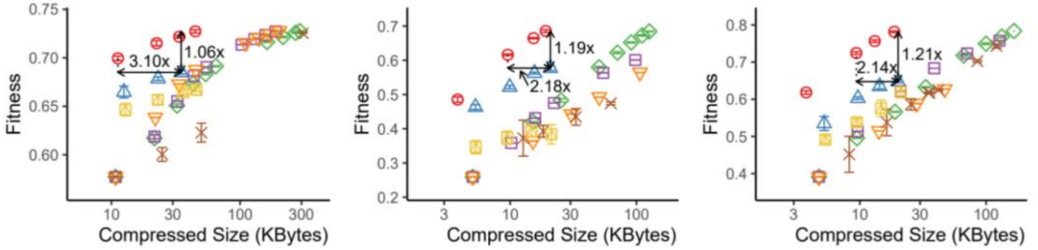


Dataset: Stock





Airquality Action Activity



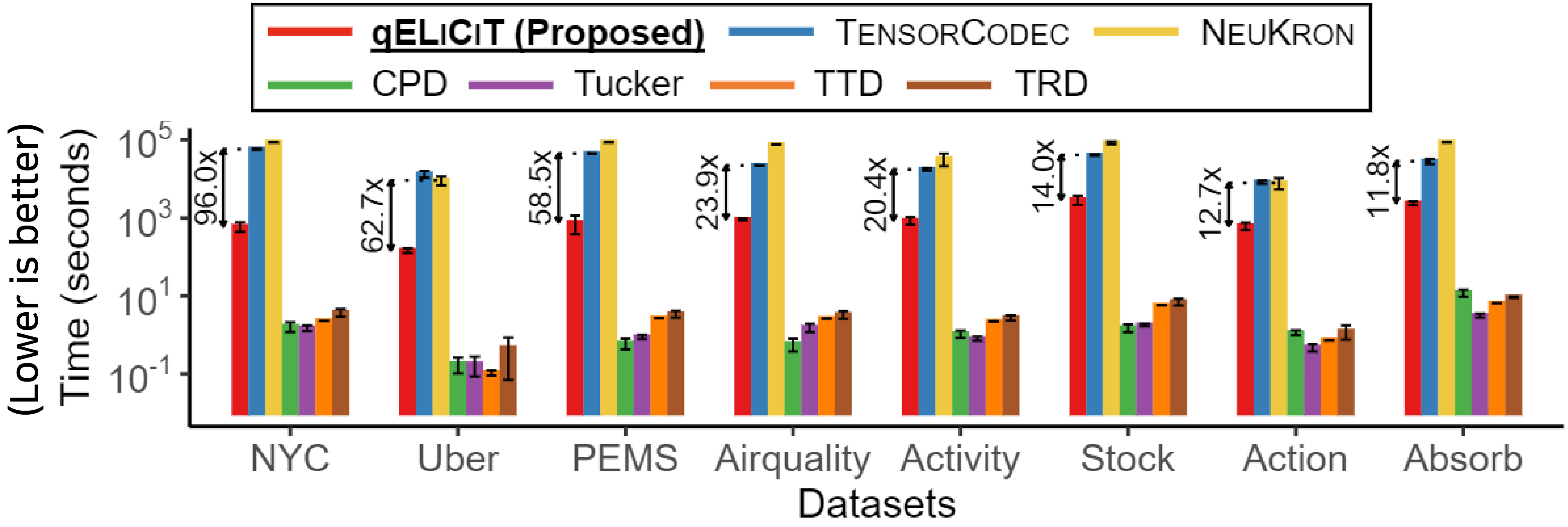
Uber Absorb NYC

Notation

Fitness: $1 - \|\mathcal{X} - \tilde{\mathcal{X}}_{\Theta}\|_F / \|\mathcal{X}\|_F$ (Higher is better)

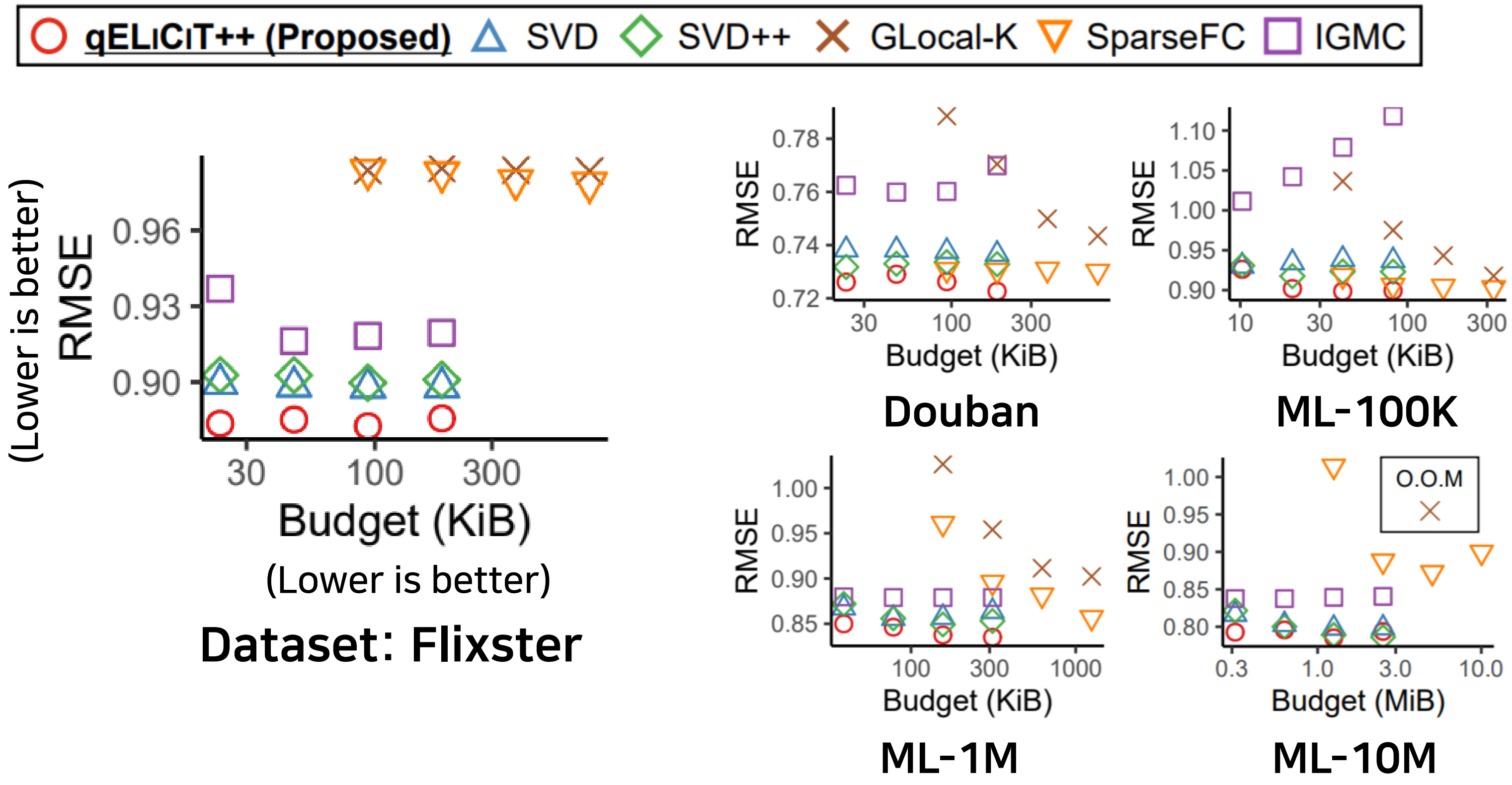
ELiCiT is Fast

- qELiCiT is significantly faster than the existing deep-learning-based methods
  - The compression speed of qELiCiT is up to 96× faster than that of TensorCodec with a similar compression size



ELiCiT is Applicable (1)

- Matrix Completion
  - In all settings, except for the largest budget on the ML-10M dataset, qELiCiT++ outperforms its competitors



ELiCiT is Applicable (2)

- Neural-network Compression
  - Achieved a compression size of 116MiB, which is 54.7% smaller than 256MiB of TFWSVD, while showing competitive accuracy

GLUE Tasks

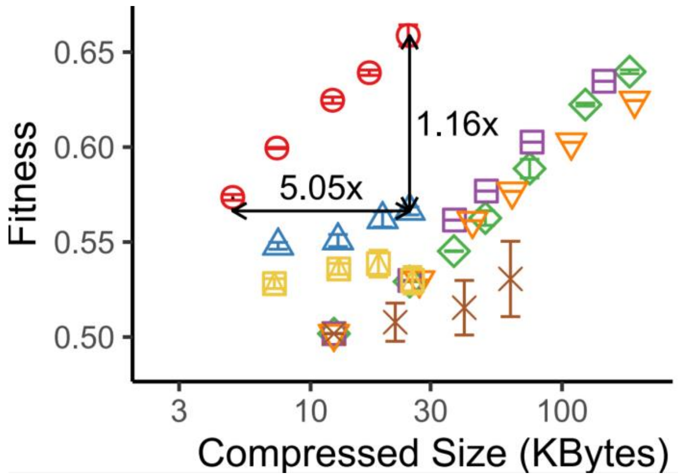
(Higher is better)

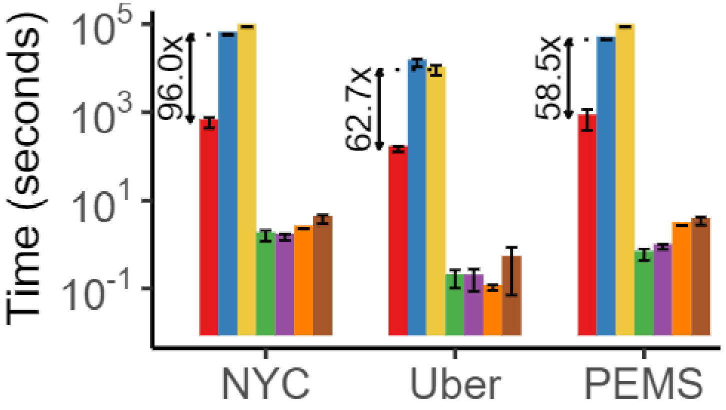
Model (Size)	CoLA	MNLI	MRPC	QNLI	QQP	SST-2	STSB	Avg.
$\text{BERT}_{\text{base}}$ (418MiB)	59.1 ±1.9	84.7 ±0.2	90.5 ±0.7	91.7 ±0.1	88.1 ±0.3	92.8 ±0.4	89.4 ±0.3	85.2
SVD (256MiB)	44.4 ±1.7	82.9 ±0.3	86.8 ±0.6	89.7 ±0.2	87.5 ±0.3	91.3 ±0.6	86.9 ±0.5	81.3
FWSVD (256MiB)	50.2 ±1.1	83.3 ±0.4	88.1 ±0.9	90.3 ±0.2	**87.6** ±0.3	91.0 ±0.5	88.1 ±0.3	82.7
TFWSVD (134MiB)	4.7 ±7.4	79.0 ±0.4	83.7 ±0.6	86.1 ±0.5	85.7 ±0.3	87.4 ±0.9	84.7 ±0.5	73.0
TFWSVD (175MiB)	42.2 ±2.4	81.5 ±0.2	86.9 ±0.9	88.8 ±0.2	86.9 ±0.2	89.4 ±0.5	87.0 ±0.4	80.4
TFWSVD (256MiB)	53.8 ±1.6	**83.5** ±0.2	89.9 ±0.9	90.4 ±0.2	87.4 ±0.3	90.7 ±0.4	88.6 ±0.5	83.5
TFW-qELiCiT (**116MiB**)	55.3 ±1.6	83.3 ±0.3	89.8 ±0.5	90.4 ±0.2	87.4 ±0.3	91.1 ±0.6	88.6 ±0.4	83.7
TFW-ELiCiT (256MiB)	**57.4** ±0.9	**83.5** ±0.5	**90.0** ±0.6	**90.6** ±0.3	87.5 ±0.3	**91.4** ±0.9	**88.7** ±0.4	**84.1**

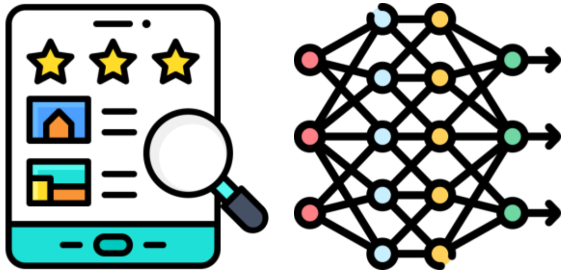
Conclusion

We propose **ELiCiT**, an efficient lossy tensor compression method

✅ **Compact and Accurate** ✅ **Fast** ✅ **Applicable**







Code and datasets are available at https://github.com/jihoonko/icdm24-elicit

ICDM 2024

ELiCiT: Effective and Lightweight Lossy Compression of Tensors


Jihoon Ko


Taehyung Kwon


Jinhong Jung


Kijung Shin