



Why do we care?

- Tensors are representations for multivariate or high-dimensional data.
- Storage and computational requirements scale exponentially with dimensions as the size of a tensor increases.
- Tensor decompositions provide compressed representation of tensors.

What is a tensor-train?

Tensor-train[1] (TT) is a compact representation format for n dimensional tensors.

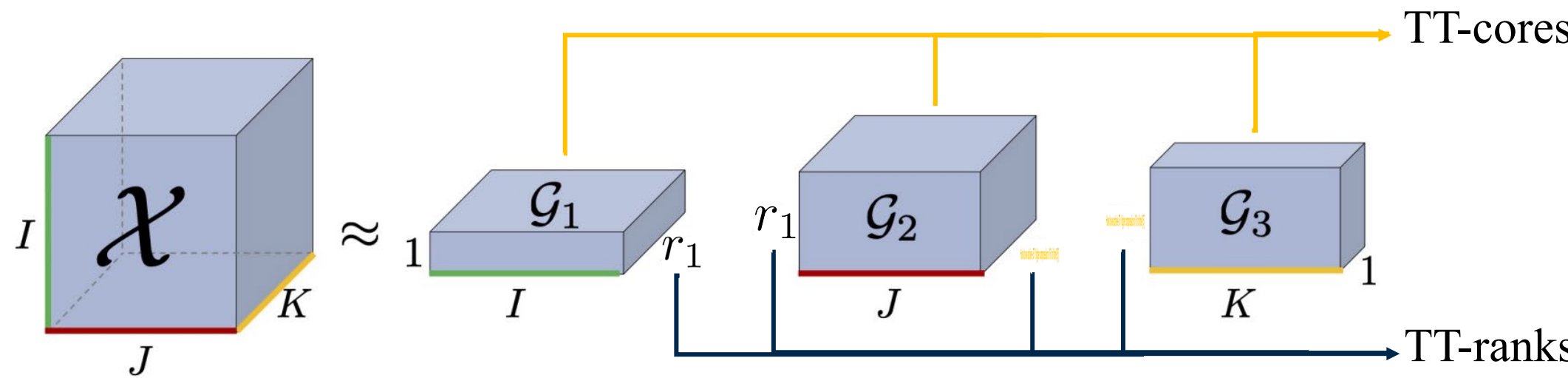


Figure 1: Visual representation of a 3-dimensional tensor in TT-format [2]

Rank of the mode- i unfolding of the d -way tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ provides an upper bound to the i -th TT-rank r_i and is computed as

$$X_{(i)} = \text{reshape}(\mathcal{X}[n_i, n_1 \dots n_{i-1} n_{i+1} \dots n_d])$$

The i -th TT-core \mathcal{G}_i can be interpreted as basis for the i -th dimension if reshaped as

$$U_i = \text{reshape}(\mathcal{G}_i, [r_{i-1} n_i, r_i])$$

Why do we need incremental algorithms?

In many applications, data is streaming. Even when it is available, the size of data becomes a bottleneck due to limited memory.

Existing incremental algorithms such as ITTD[3] and TT-FOA[4] have limitations:

- ✗ Inefficient rank adaptation or fixed rank with no adaptivity
- ✗ No compression guarantees

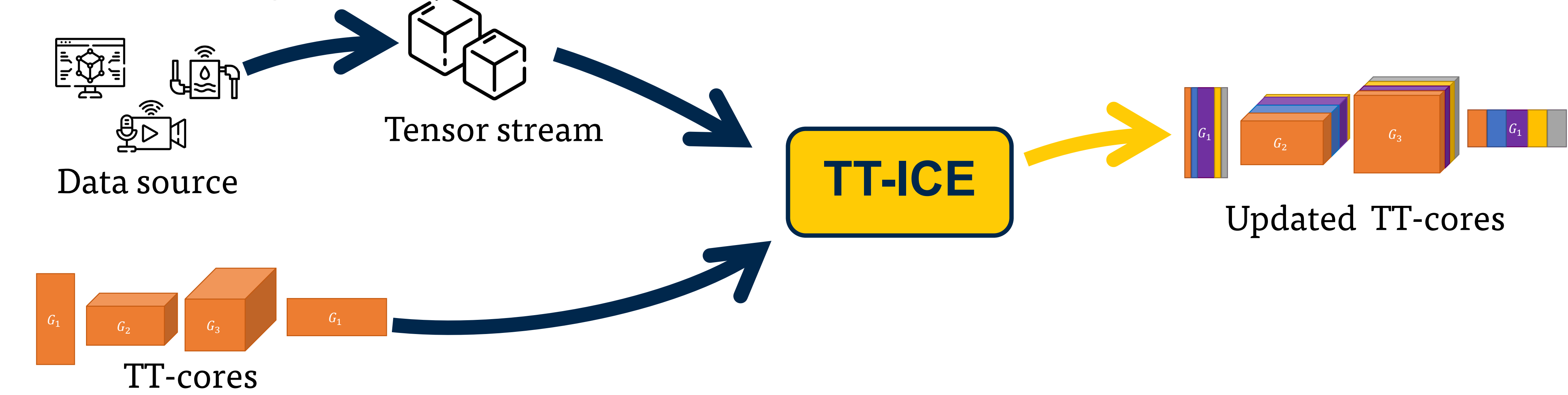
What do we offer?

- ✓ Controlled rank growth
- ✓ Obtaining and maintaining target accuracy
- ✓ Low memory footprint
- ✓ Faster execution

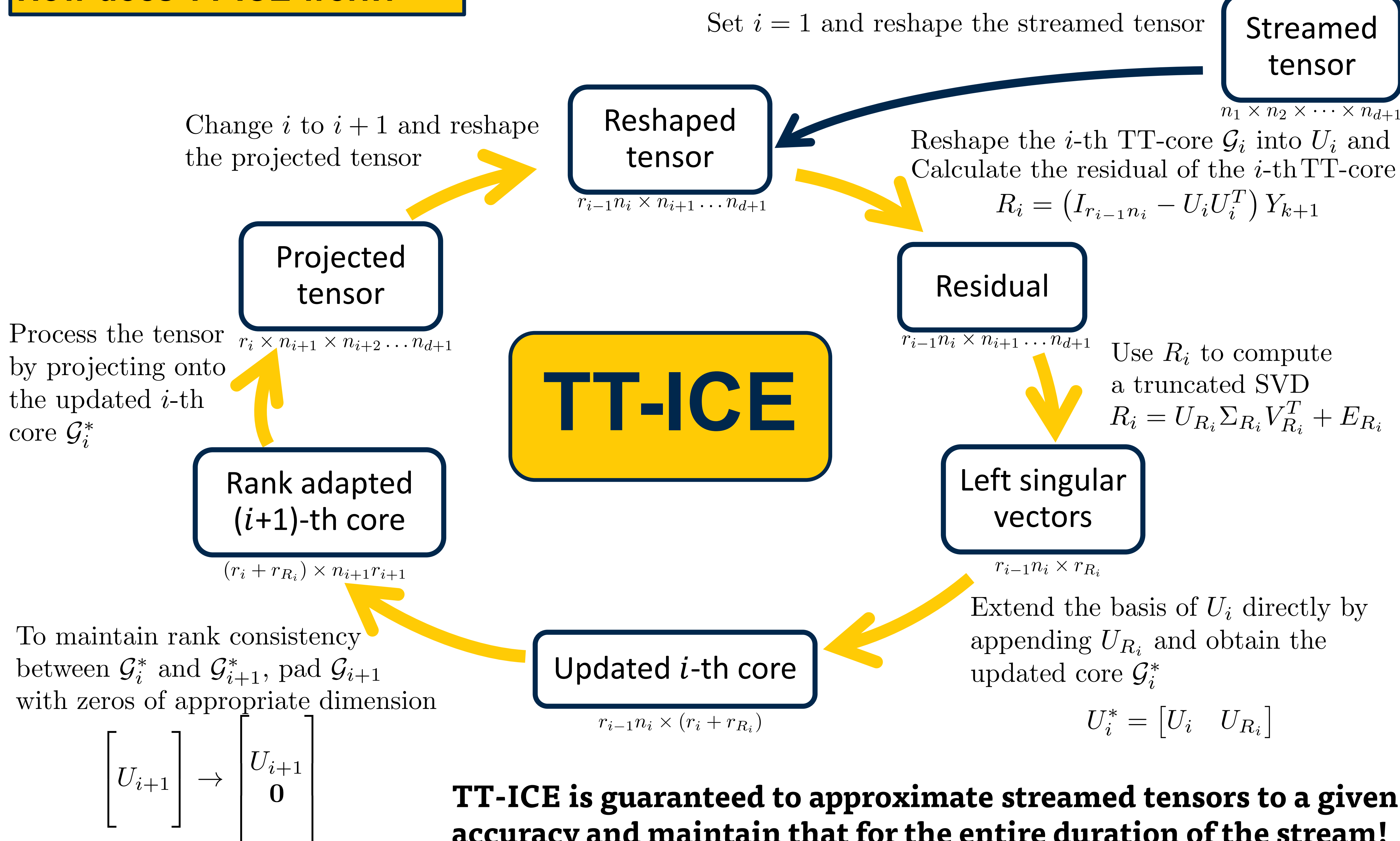
How do we achieve 57x higher compression in 5% of time?[5]

Main idea: Each core represents a basis for the corresponding dimension. We expand or adapt this basis when it is insufficient for representing a new data point. We call this algorithm TT-Incremental Core Expansion (TT-ICE)[5].

Problem setting:



How does TT-ICE work?



How do we make it faster?

We have developed three heuristics to further speed up the process. This heuristically enhanced version of TT-ICE is called TT-ICE*[5].

Skip updating all cores

When the existing basis adequately represents a new tensor, we don't update them.

Select a subset from batch

If multiple tensors arrive in a batch, we only update with those that are not well represented.

Skip updating filled cores

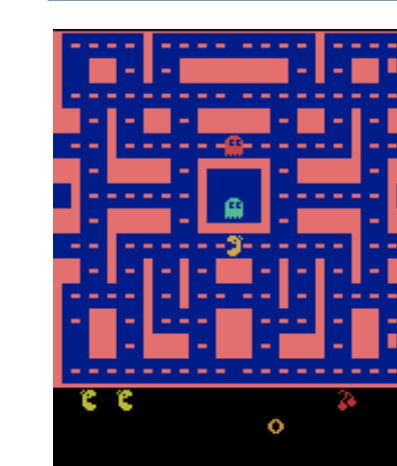
If a core is full rank, skip updating that core and proceed with the next one.

How does it perform?

We benchmark TT-ICE and TT-ICE* against ITTD [4], the only available algorithm with rank adaptation, on two problems:

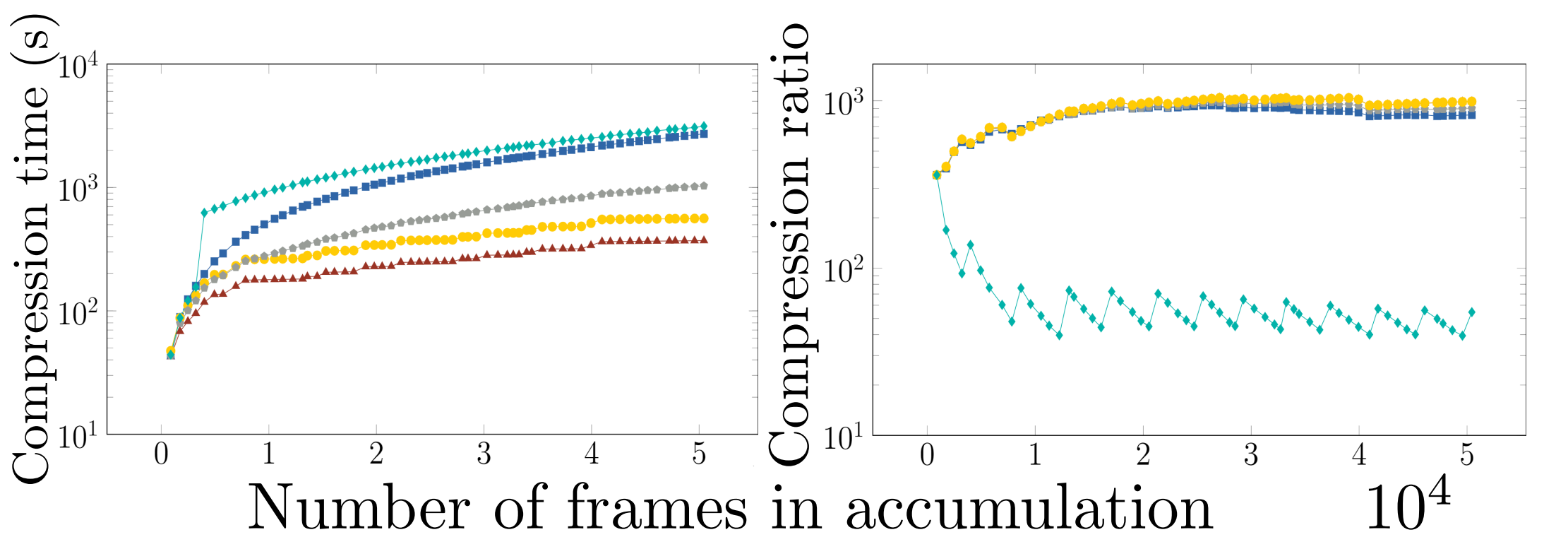
1. ATARI-game video sequences
2. Time-dependent PDE solutions of self-oscillating gels

ATARI frames



60 game sequences with varying number of frames. Each sequence is a 5-way tensor with shape $30 \times 28 \times 40 \times 3 \times N(i)$

Figure 2: Example MsPacman frame[3]



Figures 3 and 4: Number of frames vs execution time (left) and compression ratio (right) with $\epsilon_{des} = 0.1$ [5]

We achieve 88% reduction in time with TT-ICE*!

Self-oscillating gel simulations

Parametric PDE simulated for 6400 unique parameter combinations. Each simulation is a 5-way tensor with shape $7 \times 13 \times 37 \times 3 \times 10$.

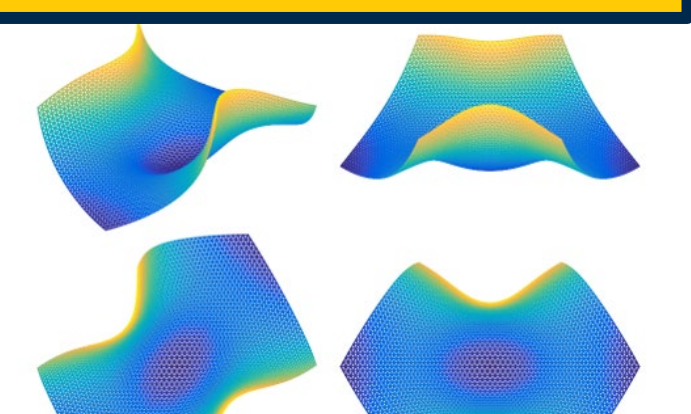
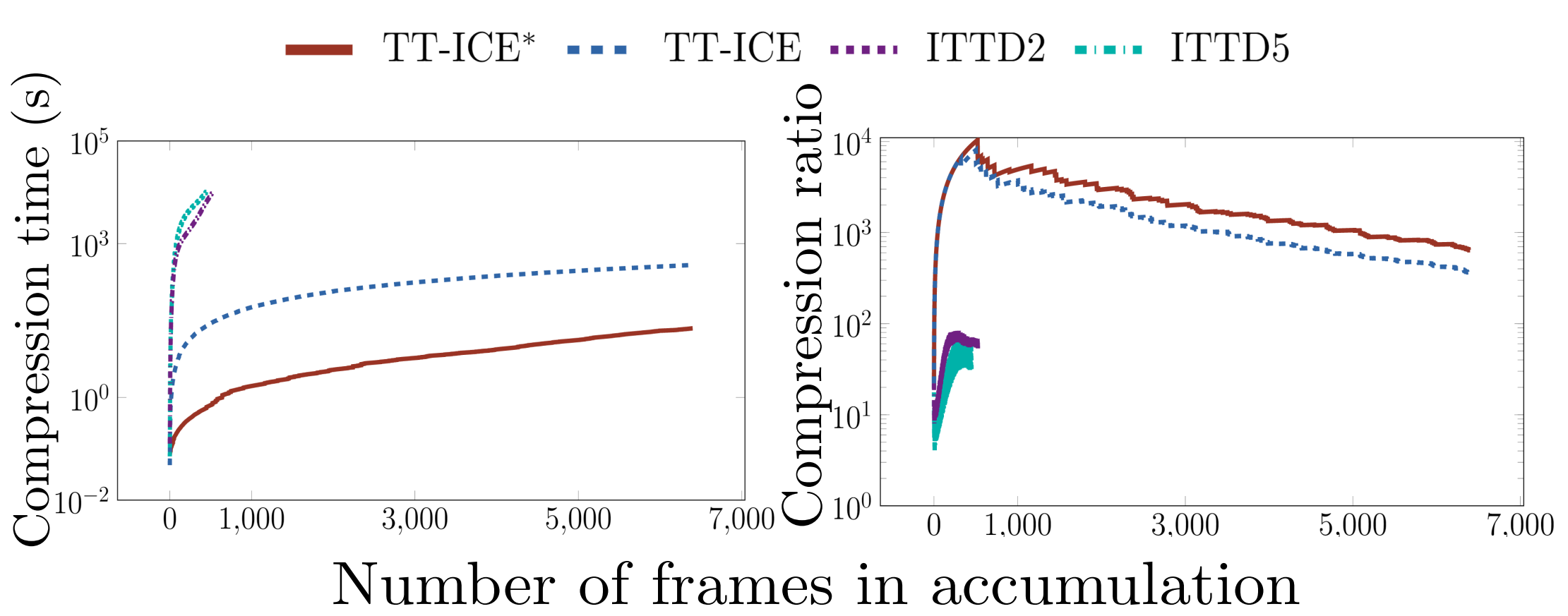


Figure 5: Example snapshots from PDE simulation [3]



Figures 6 and 7: Number of frames vs execution time (left) and compression ratio (right) with $\epsilon_{des} = 0.1$ [5]

Low memory footprint of TT-ICE enables compression where literature fails!

References & Acknowledgements

[1] Oseledets, Ivan V. "Tensor-train decomposition." *SIAM Journal on Scientific Computing* 33.5 (2011): 2295-2317.

[2] Panagakis, Yannis, et al. "Tensor methods in computer vision and deep learning." *Proceedings of the IEEE* 109.5 (2021): 863-890.

[3] Liu, Huazhong, et al. "An incremental tensor-train decomposition for cyber-physical-social big data." *IEEE Transactions on Big Data* 7.2 (2018): 341-354.

[4] Le Trung, Thanh, et al. "Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors." (2021).

[5] Aksoy, Doruk, et al. "An Incremental Tensor Train Decomposition Algorithm." *arXiv preprint arXiv:2211.12487* (2022).

