How to Square Tensor Networks and Circuits Without Squaring Them

Lorenzo Loconte
University of Edinburgh, UK

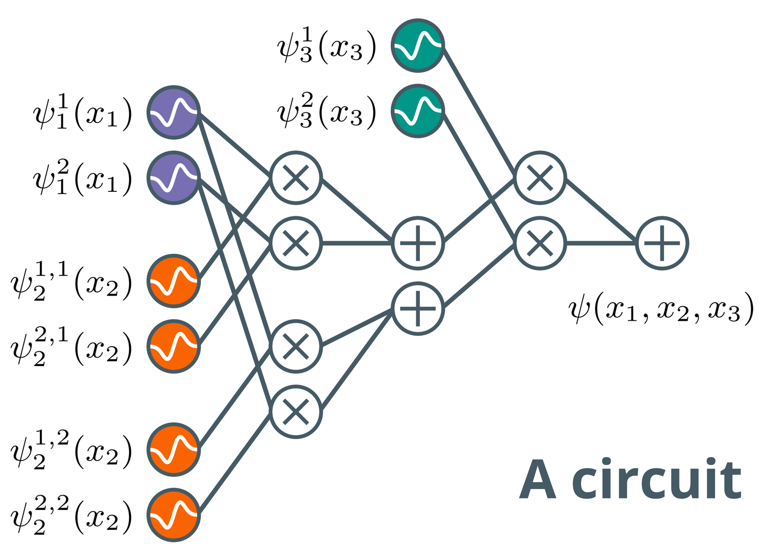
Adrián Javaloy
University of Edinburgh, UK

Antonio Vergari
University of Edinburgh, UK

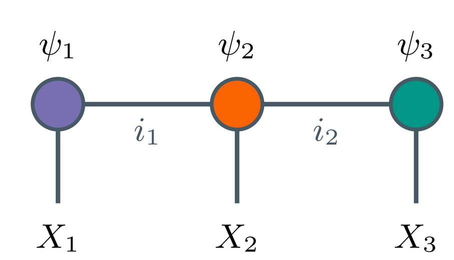
Paper **Code**

TL;DR:

"We reinterpret canonical forms of popular tensor networks as conditions over circuits, unlocking more factorization structures supporting efficient variable marginalization"



A circuit



A tensor train

$$\psi(x_1,x_2,x_3) = \sum_{i_1=1}^{R}\sum_{i_2=1}^{R} \psi_1[x_1,i_1]\ \psi_2[i_1,x_2,i_2]\ \psi[i_2,x_3]$$

Tensor factorizations and tensor networks are circuits [1]

We flexibly build tensor factorizations as neural networks by stacking sum and product computational units [2]

Encode probability distributions via the Born rule

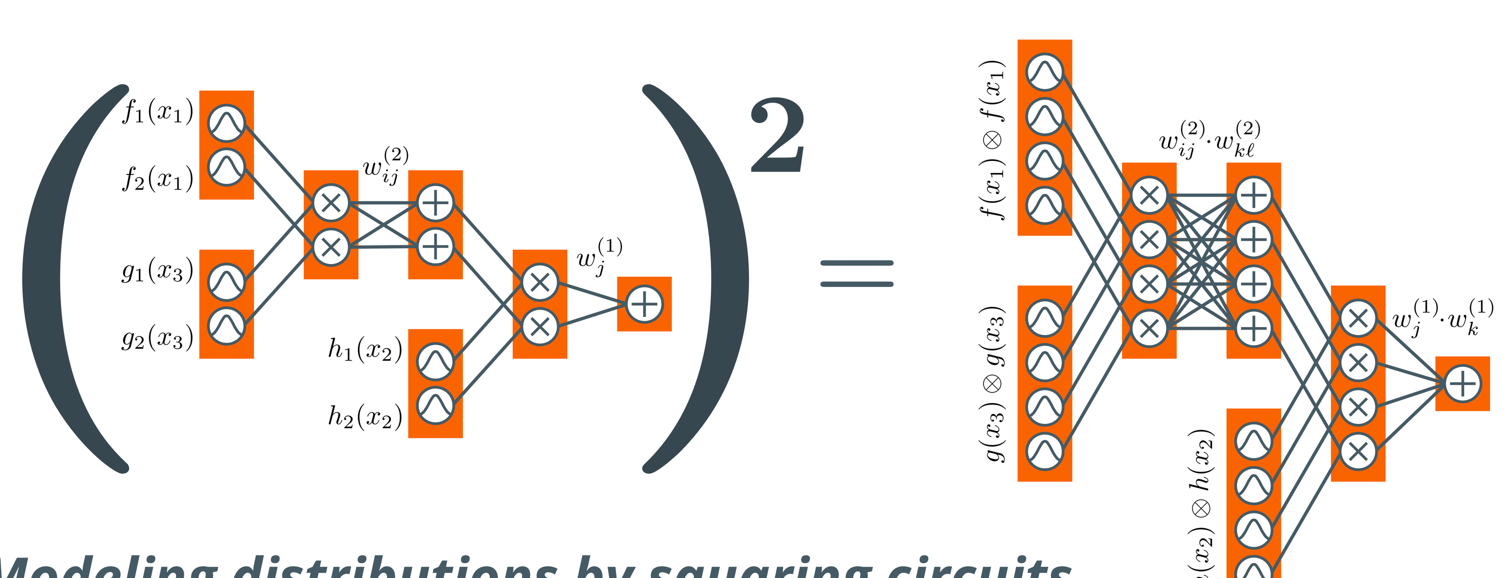
$$p(\mathbf{X}) \propto |\psi(\mathbf{X})|^2 = \psi(\mathbf{X})\psi(\mathbf{X})^*$$

Complexity of variable marginalization is $\mathcal{O}(R^2)$ ...

... but it is simplified by using ***canonical forms*** [3, 4]

e.g., $\psi_1^\dagger\psi_1 = \mathbf{I}_R \qquad \sum_{x_2}\psi_2[:,x_2,:]^\dagger\psi_2[:,x_2,:] = \mathbf{I}_R$

$\implies$ computing $p(x_3)$ requires time $\mathcal{O}(R)$



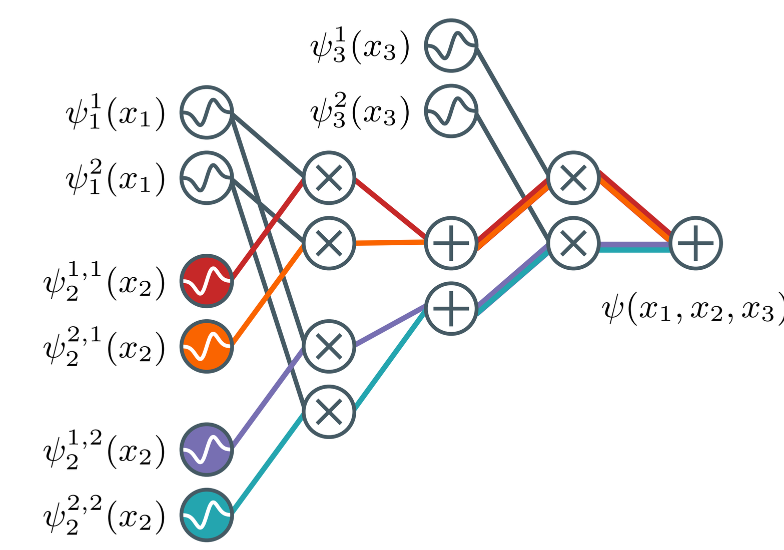
Modeling distributions by squaring circuits

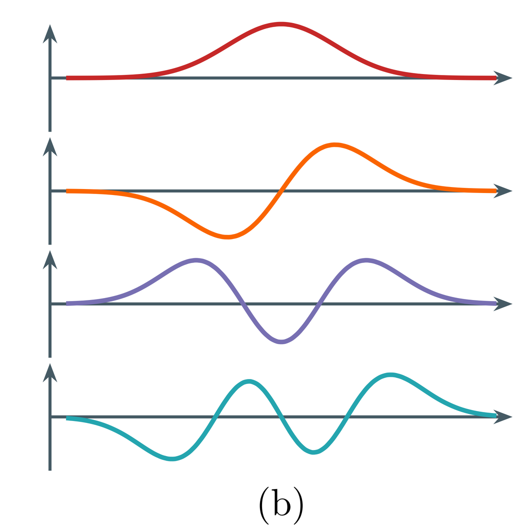
$p(\mathbf{x}) = Z^{-1}|c(\mathbf{x})|^2 = c(\mathbf{x})c(\mathbf{x})^*$ (Born rule)

Provable expressiveness improvements... [5, 6]

...but variables marginalization requires time $\mathcal{O}(|c|^2)$

How to speed-up marginalization in squared circuits?



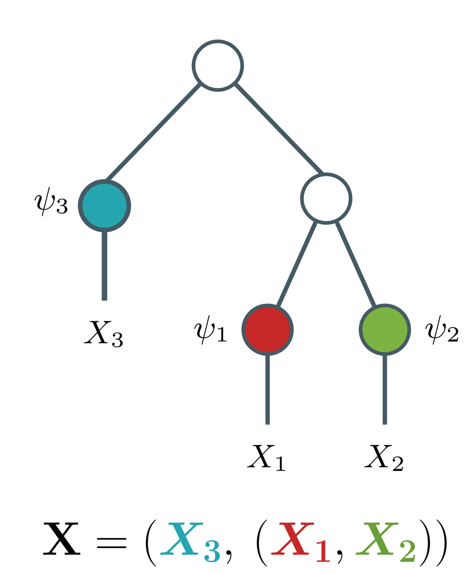


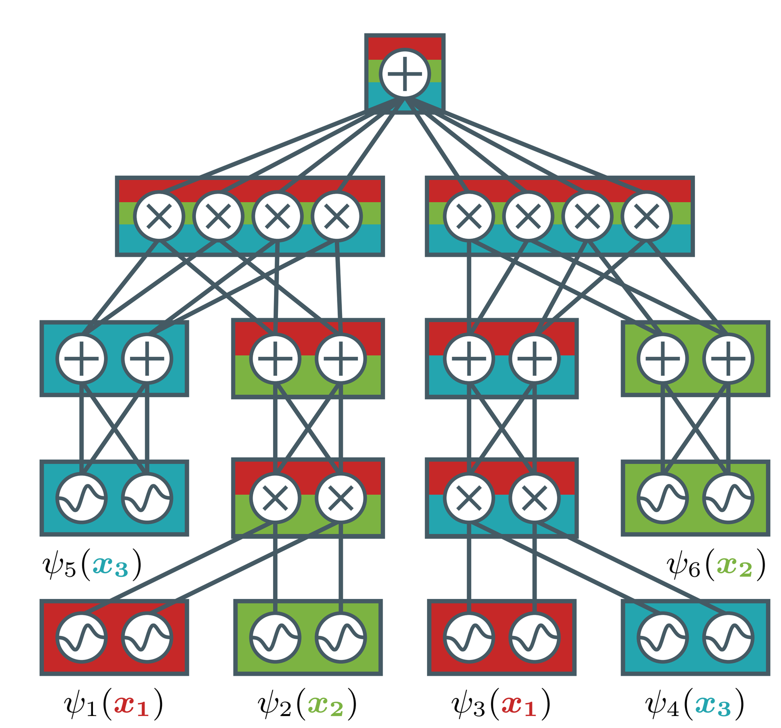
Orthonormality ensures linear-time marginalization

P1. input functions over X form an orthonormal basis

P2. each sum unit partitions its set of input basis

Computing $p(\mathbf{Z})$, $\mathbf{Z} \subseteq \mathbf{X}$, can be done in time $\mathcal{O}(|c|)$





Enable polytime marginalization in factorization structures encoding multiple variables partitionings

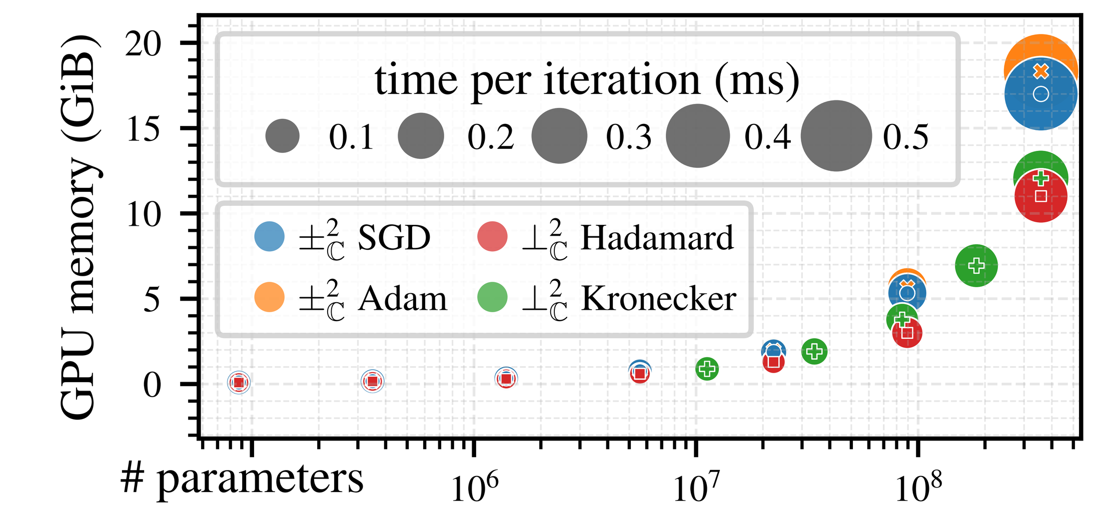
Unlock a *strictly* larger set of factorization structures [7]

U1. input layers over X encode orthonormal basis

U2. each sum layer parameter $\mathbf{W}$ satisfies $\mathbf{W}\mathbf{W}^\dagger = \mathbf{I}$

U3. each sum layer partitions its set of input layers

Ensures the partition function is $Z = \int |c(\mathbf{x})|^2\, \mathrm{d}\mathbf{x} = 1$

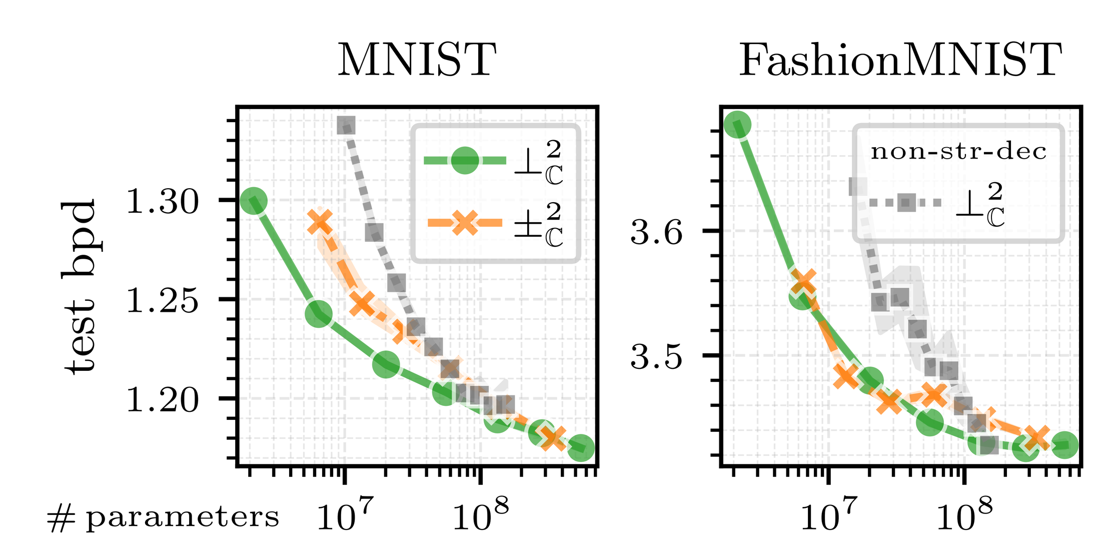


Speed-up training on large circuits

Learning by maximum-likelihood estimation...

$\min\ \log Z - \sum_{\mathbf{x}\in\mathcal{D}} 2\log|c(\mathbf{x})|$

...by optimizing over the Stiefel manifold and by using a preliminary version of **POGO** [8]



Our conditions retain distribution estimation performances with up to 1B parameters

when compared to unconstrained squared circuits and at parity of model size

References

[1] L. Loconte et al. "What is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit it)?" In: *Transactions on Machine Learning Research* (2025). Featured Certification.

[2] Y. Choi, A. Vergari, and G. Van den Broeck. *Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling*. Tech. rep. University of California, Los Angeles (UCLA), 2020.

[3] R. Orús. "A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States". In: *Annals of Physics* 349 (2013), pp. 117–158.

[4] R. Bonnevie and M. N. Schmidt. "Matrix Product States for Inference in Discrete Probabilistic Models". In: *Journal of Machine Learning Research* 22 (2021), 187:1–187:48.

[5] L. Loconte et al. "Subtractive Mixture Models via Squaring: Representation and Learning". In: *The Twelfth International Conference on Learning Representations (ICLR)*. 2024.

[6] L. Loconte, S. Mengel, and A. Vergari. "Sum of Squares Circuits". In: *The 39th Annual AAAI Conference on Artificial Intelligence (AAAI)*. 2025.

[7] A. Vergari et al. "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference". In: *Advances in Neural Information Processing Systems 34 (NeurIPS)*. Curran Associates, Inc., 2021, pp. 13189–13201.

[8] A. Javaloy and A. Vergari. *An Embarrassingly Simple Way to Optimize Orthogonal Matrices at Scale*. 2026. arXiv: 2602.14656 [cs.LG].