FunFact: Build Your Own Tensor Decomposition Model in a Breeze

Daan Camps[1] **and Yu-Hang Tang**[1¶]

1 Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Submitted: 22 February 2022
Published: unpublished

Summary

`FunFact` is a Python package that aims to simplify the design of matrix and tensor factorization algorithms. It features a powerful programming interface that augments the NumPy API with Einstein notations for writing concise tensor expressions. Given an arbitrary forward calculation scheme, the package will solve the corresponding inverse problem using stochastic gradient descent, automatic differentiation, and multi-replica vectorization. Its application areas include tensor decomposition, quantum circuit synthesis, and neural network compression. It is GPU- and parallelization-ready thanks to modern numerical linear algebra backends such as `JAX` (Bradbury et al., 2018) and `PyTorch` (Paszke et al., 2019).

Statement of Need

Tensor factorizations have found numerous applications in various domains (Liu, 2021), (Kolda & Bader, 2009). Among the most prominent are tensor networks in quantum physics (Orús, 2019), tensor decompositions in machine learning (Kossaifi et al., 2019) and signal processing (Sidiropoulos et al., 2017), (Hunyadi et al., 2014), and quantum computation (Markov & Shi, 2008).

Thus far, most tensor factorization models are solved by special-purpose algorithms designed to factor the target data into a model with the prescribed structure. Furthermore, the models that are being used are often limited to linear contractions between the factor tensors, such as standard inner and outer products, elementwise multiplications, and matrix Kronecker products. Extending such a special-purpose solver to more generalized models can be daunting, especially if nonlinear operations are considered.

`FunFact` solves this problem and fills the gap. It offers an embedded Domain Specific Language (eDSL) in Python for creating nonlinear tensor algebra expressions that use generalized Einstein operations. Using the eDSL, users can create custom tensor expressions and immediately use them to solve the corresponding inverse factorization problem. `FunFact` solves this inverse problem by combining stochastic gradient descent, automatic differentiation, and model vectorization for multi-replica learning. This combination achieves instantaneous time-to-algorithm for all conceivable tensor factorization models. It allows the user to explore the entire universe of nonlinear tensor factorization models.

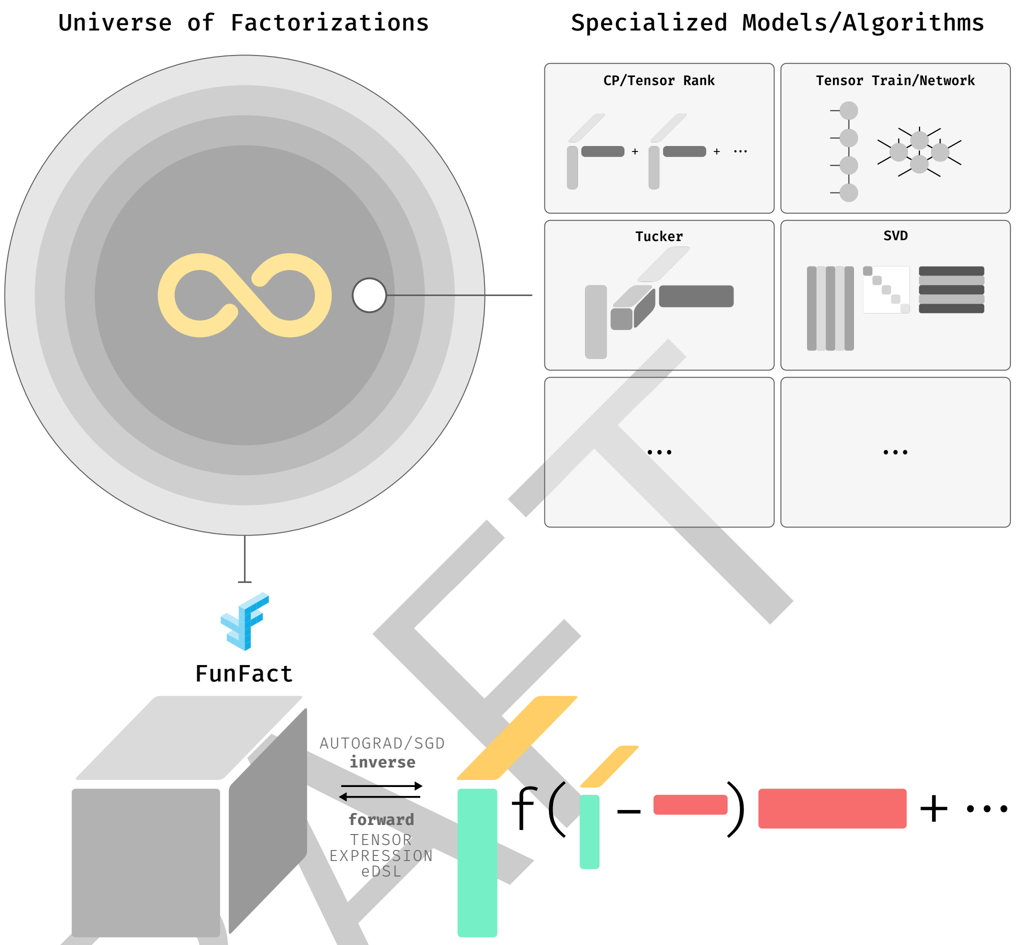

Figure 1: Tensor rank, Tucker, tensor network, and singular value decompositions are among the most popular factorization models that have found numerous applications. However, the popular models studied in the literature only form a small subset of all possible tensor factorization models that can be constructed from generalized contractions, semiring operations, nonlinearities, and more. `FunFact` allows users to probe this vastly larger universe of models through an eDSL for tensor expressions. From the forward computation defined by a tensor expression, `FunFact` can solve the inverse factorization problem using a combination of techniques such as lazy evaluation, automatic differentiation, and stochastic gradient descent.

Functionality

`FunFact`'s core functionality consists of three parts:

1. a rich and flexible eDSL to express complicated tensor factorization models with a concise notation,
2. an interpreter for forward evaluation of user-defined tensor expressions, and
3. algorithms that use backpropagation and automatic differentiation to compute the model gradients and to optimize the factorization model for a target tensor using stochastic gradient descent.

eDSL for tensor expressions

The central notions in the `FunFact` eDSL are `tensor` and `index` objects that can be used to construct tensor expressions. Indices are used to label the dimensions of tensor expressions.

Repeated indices are contracted over in an Einstein operation between two tensor expressions. Abstract `tensor` objects can be initialized either (1) based on their shape or (2) from concrete numerical data. Optional arguments for `tensor` objects include

- a human-readable label,
- an `initializer` that provides a generator for a particular distribution,
- a `condition` that the tensor is expected to satisfy, such as nonnegativity and orthogonality, which is implemented as a penalty term during the optimization process, and
- an `optimizable` flag that indicates if the tensor should be updated during the optimization process.

Tensor expressions can be indexed, indexless, or hybrid. `FunFact` implements a tensor algebra language model based on a context-free grammar. Index decorators, explicit output index specification, generalized contractions with semiring operations, nonlinearities, and other features make the `FunFact` language rich and flexible.

Forward evaluation

In `FunFact`, tensor expressions are handled by a lazy evaluation model. Only basic analyses are performed after the user defines a tensor expression, such as shape and dimensionality checking. After that, the computational graph of the expression is stored for later use. A tensor expression can be explicitly evaluated in the forward direction, *i.e.,* from leaf tensors to the result, using the `Factorization` class, which serves as an interpreter for tensor expressions.

Optimizing for a target tensor

The central capability of `FunFact` is implemented in the `factorize` method, which can:

1. run the model in the forward direction, and then
2. run a backpropagation pass with automatic differentiation to find the gradients of a given cost function with regard to the leaf tensors in the tensor expression, and then
3. update the leaf tensors using a stochastic gradient descent algorithm.

The `factorize` method allows a user to optimize a model as defined by *any* tensor expression towards a target tensor, thereby solving the inverse problem. The method has many knobs that the user can fine-tune for the problem at hand to achieve faster and better convergence. These include the learning rate, the optimization integrator, the cost function, the weights of the penalty terms, and any of the numerous hyperparameters.

Example

We illustrate the use and flexibility of `FunFact` by providing reference tensor expressions for a few matrix and tensor decomposition models. Upper-case symbols are assumed to be abstract tensors of the appropriate dimensions, while lower-case symbols are abstract indices.

Tensor Expression	Description
`U[i, r] * V[j, r]`	Rank-r decomposition of matrix $A(i,j)$
`Z[r1, r2, r3] * S1[r1, i1] * S2[r2, i2] * S3[r3, i3]`	Tucker decomposition of tensor $T_{i_1 i_2 i_3}$ (Kolda & Bader, 2009)
`(A[i1, ~r] * B[i2, r]) * C[i3, r]`	Tensor rank decomposition of tensor $T_{i_1 i_2 i_3}$ (Kolda & Bader, 2009)
`G1[i1, r1] * G2[i2, r1, r2] * G3[i3, r2, r3] * G4[i4, r3]`	Tensor train decomposition of tensor $T_{i_1 i_2 i_3 i_4}$ (Oseledets, 2011)

Tensor Expression	Description
`ff.exp(-(U[i, ~k] - V[j, ~k])**2) * A[k] + B[[ ]]`	RBF kernel decomposition of matrix $A(i,j)$ (Rebrova & Tang, 2021)
`ff.eye(2**i) & ff.tensor(4, 4, prefer=cond.Unitary)`	Two-qubit unitary quantum gate (Nielsen & Chuang, 2000)

Related research and software

FunFact is closely related to several other software packages that provide Einstein notations and Domain Specific Languages (DSL) for tensor algebra. Notable examples are TensorOperations.jl (Haegeman, 2021) that provides Einstein index notations in julia, Tensor Comprehensions (Facebook Research, 2018) that provides a DSL to automatically synthesize high-performance machine learning kernels, einops (Rogozhnikov, 2022) that enables tensor operations through readable and reliable code, TACO (Kjolstad et al., 2017): the tensor algebra compiler, and COMET (Mutlu et al., 2021) which is designed for high-performance contractions of sparse tensors. FunFact is distinct from all of the aforementioned projects in that it aims to solve the inverse decomposition problem from the model description as a nonlinear tensor algebra expression. Additionally, FunFact offers increased generality compared to other tensor decomposition software libraries such as Tensorly (Kossaifi et al., 2019), Tensor Toolbox (Bader et al., 2021) or Tensorlab (Vervliet et al., 2016) which only provide specialized implementations for computing fixed-form tensor decompositions such as Tucker or tensor rank decompositions.

Acknowledgement

The authors thank Liza Rebrova for her input on this work. This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231.

References

Bader, B. W., Kolda, T. G.others. (2021). *Tensor toolbox for MATLAB* (Version 3.2.1) [Computer software]. https://www.tensortoolbox.org

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.2.5) [Computer software]. http://github.com/google/jax

Facebook Research. (2018). *Tensor comprehensions* (Version 0.1.1) [Computer software]. https://github.com/facebookresearch/TensorComprehensions

Haegeman, J. (2021). *TensorOperations.jl* (Version 3.2.3) [Computer software]. https://github.com/Jutho/TensorOperations.jl

Hunyadi, B., Camps, D., Sorber, L., Paesschen, W. V., Vos, M. D., Huffel, S. V., & De Lathauwer, L. (2014). Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, *2014*(1), 139. https://doi.org/10.1186/1687-6180-2014-139

Kjolstad, F., Kamil, S., Chou, S., Lugato, D., & Amarasinghe, S. (2017). The tensor algebra compiler. *Proc. ACM Program. Lang.*, *1*(OOPSLA). https://doi.org/10.1145/3133901

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Rev.*, *51*(3), 455–500. https://doi.org/10.1137/07070111X

Kossaifi, J., Panagakis, Y., Anandkumar, A., & Pantic, M. (2019). TensorLy: Tensor learning in python. *Journal of Machine Learning Research (JMLR)*, *20*(26).

Liu, Y. (2021). *Tensors for data processing: Theory, methods, and applications*. Elsevier Science. ISBN: 9780323859653

Markov, I. L., & Shi, Y. (2008). Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, *38*(3), 963–981. https://doi.org/10.1137/050644756

Mutlu, E., Tian, R., Ren, B., Krishnamoorthy, S., Gioiosa, R., Pienaar, J., & Kestor, G. (2021). *COMET: A domain-specific compilation of high-performance computational chemistry*. https://arxiv.org/abs/2102.06827

Nielsen, M. A., & Chuang, I. L. (2000). *Quantum computation and quantum information*. Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

Orús, R. (2019). Tensor networks for complex quantum systems. *Nature Reviews Physics*, *1*(9), 538–550. https://doi.org/10.1038/s42254-019-0086-7

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, *33*(5), 2295–2317. https://doi.org/10.1137/090752286

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Rebrova, E., & Tang, Y.-H. (2021). *Nonlinear matrix approximation with radial basis function components*. https://arxiv.org/abs/2106.02018

Rogozhnikov, A. (2022). *Einops* (Version 0.4.0) [Computer software]. https://github.com/arogozhnikov/einops

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., & Faloutsos, C. (2017). Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, *65*(13), 3551–3582. https://doi.org/10.1109/TSP.2017.2690524

Vervliet, N., Debals, O., & De Lathauwer, L. (2016). Tensorlab 3.0 — numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization. *2016 50th Asilomar Conference on Signals, Systems and Computers*, 1733–1738. https://doi.org/10.1109/ACSSC.2016.7869679