TensorType:

Implementing and Extending Deep Learning with Types

Bruno Gavranović

March 31, 2026

1 Introduction

The categorical deep learning programme ([GLD+24, CGG+22, FST21]) has provided a systematic way to reason about and implement neural network architectures, loss functions, optimisers, automatic differentiation, and entire supervised learning systems. However, practical implementation of these ideas remains confined to frameworks such as TensorFlow, PyTorch and JAX in dynamically-typed languages. This is because attempts to build frameworks for deep learning in statically-typed languages have struggled with expressiveness and ergonomics, typically only replicating what exists without reimagining what can be.

At the same time, an increasing number of AI systems operate on structured data: proof terms, algebraic expressions, tactics, and lambda terms. Despite this, researchers must flatten structured data into sequences and serialise multi-step interaction protocols, causing information that should guide learning to be lost, and making experimentation with structure prohibitively difficult.

In this talk we introduce TensorType ([Gav25, Č26]), a tensor processing framework implemented in Idris 2. It addresses these shortcomings by reimagining tensors over trees, inductive types, and interaction protocols via a novel type-safe container-based interface, and aims to demonstrate that ergonomics and rigor need not be at odds. It provides tensor operations checked at compile-time while enabling fundamentally new capabilities: tensors that branch and recurse instead of being confined to rectangular shapes, as well as a native support of multi-step interaction protocols.

2 Tensors as Containers

The well-known construction of containers ([AAG03]) captures the idea that datatypes consist of groups of memory locations where data can be stored. They are a central construction in TensorType because of their *composition product* ([Spi23]). Surprisingly, via a simple fold over a list of containers, this composition product gives rise to exactly the tensors in widespread use in NumPy/PyTorch and similar deep learning frameworks. We showcase this in Idris 2 directly:[1]

```
Tensor : List Cont -> Cont
Tensor cs = foldr (>@) Scalar
```

From this single definition, all standard tensor processing operations arise:

Type-safe indexing Indexing into a tensor is given by the positions of the composed container. An index into a `Tensor [BinTree] Double` is a path in the tree, and supplying a path that does not exist in a given tree is a compile-time type error.

Generalised linear algebra as tensor monoids Matrix multiplication and other linear algebra operations are given by a monoid structure of a container with respect to the Hancock tensor product, and transposition is possible when a container is Naperian, i.e. its shape is a singleton. This exactly recovers applicative programming ([Gib16]) within a purely categorical setting, and allows us to compute dot products over not just vectors, but also trees, lists, and many other datatypes.

Reshaping as container morphisms A reshape operation transforms the shape of a tensor without altering its content. In TensorType, this is precisely a dependent lens, a morphism of containers. This recovers standard reshapes, views, but also traversals of structures, with all of their well-typedness checked at compile-time.

Efficient representation By modelling reshapes as container morphisms, we recover a mechanism to perform reshapes in $O(1)$ time without a need for a separate "strided" representation, as is done in frameworks like NumPy/PyTorch. This optimisation emerged as a byproduct of principled library design, instead of as a performance workaround sacrificing compositionality.

[1] This is exactly how it is implemented in code, see here.

References

[AAG03] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Categories of Containers. In Andrew D. Gordon, editor, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, pages 23–38, Berlin, Heidelberg, 2003. Springer. doi:10.1007/3-540-36576-1_2.

[CGG+22] Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical Foundations of Gradient-Based Learning. In Ilya Sergey, editor, *Programming Languages and Systems*, Lecture Notes in Computer Science, pages 1–28, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-99336-8_1.

[FST21] Brendan Fong, David Spivak, and Rémy Tuyéras. Backprop as functor: a compositional perspective on supervised learning. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '19, pages 1–13, Vancouver, Canada, June 2021. IEEE Press.

[Gav25] Bruno Gavranović. TensorType: Implementing and extending deep learning with types, 2025. URL: https://github.com/bgavran/TensorType.

[Gib16] Jeremy Gibbons. Aplicative programming with naperian functors (extended abstract). In *Proceedings of the 1st International Workshop on Type-Driven Development*, TyDe 2016, page 13–14, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976022.2976023.

[GLD+24] Bruno Gavranović, Paul Lessard, Andrew Dudzik, Tamara von Glehn, João G. M. Araújo, and Petar Veličković. Position: Categorical Deep Learning is an Algebraic Theory of All Architectures. *arXiv e-prints*, page arXiv:2402.15332, February 2024. arXiv:2402.15332, doi:10.48550/arXiv.2402.15332.

[Spi23] David I. Spivak. A reference for categorical structures on $\mathbf{Poly}$, December 2023. arXiv:2202.00534 [math]. URL: http://arxiv.org/abs/2202.00534.

[Č26] Ieva Čepaitė. "tensortype is great", 2026.