

Why DTDR Is Not Compression

Jonathan West

February 4, 2026

1 Introduction

The term *compression* is frequently used to describe any method that reduces the storage footprint of numerical data. As a result, techniques that apply transforms, quantisation, or coefficient redistribution are often grouped together under a single conceptual umbrella. Distributed Transform–Domain Representation (DTDR) may, at a superficial level, appear to fall into this category, since it can reduce the persistent storage size of numerical model parameters relative to conventional floating–point representations.

This document clarifies why DTDR should *not* be understood as a compression scheme in the conventional sense. While DTDR shares certain mathematical tools with transform–based compression methods, it differs fundamentally in purpose, optimisation target, and operational role. Compression schemes are designed to minimise storage or transmission cost subject to a distortion criterion. DTDR, by contrast, is designed as a *persistent transform–domain representation* that preserves computational functionality and semantic structure.

The distinction is not merely semantic. It has direct implications for how the representation is used, what properties are preserved, and what forms of computation are supported. Confusing DTDR with compression obscures the architectural role it is intended to play and leads to incorrect comparisons with conventional codecs.

This document aims to make that distinction precise.

2 What Compression Means in Practice

In practical terms, compression refers to a class of techniques whose primary objective is to reduce the number of bits required to represent data while maintaining acceptable reconstruction quality. The defining feature of compression systems is that they optimise a *rate–distortion trade-off*: storage or transmission cost is minimised subject to a constraint on reconstruction error, typically measured in a signal domain.

Most widely deployed compression schemes follow a common pattern:

- transformation of the data into a basis that concentrates energy or variance,
- quantisation of transform coefficients to reduce precision, and
- entropy coding to eliminate statistical redundancy.

Classical image codecs, such as those based on the discrete cosine transform, and more recent learned codecs based on autoencoders both follow this paradigm. Although the mathematical tools differ, the optimisation objective remains the same: minimise storage cost while producing an acceptable approximation of the original signal after reconstruction.

Two properties are characteristic of such systems. First, the compressed representation is not intended to be used directly for computation; it is an intermediate artefact whose sole purpose is efficient storage or transmission. Second, reconstruction to the original or an approximate signal is the *end goal* of the process. Any computation performed on the compressed representation itself is incidental and typically unstable or heuristic.

From this perspective, compression schemes are evaluated by metrics such as compression ratio, reconstruction error, perceptual quality, or bitrate. They are not designed to preserve the computational behaviour of the data under downstream operations.

3 What DTDR Optimises Instead

DTDR is motivated by a different set of constraints. Rather than optimising a rate–distortion objective, DTDR is designed to preserve *computational utility* under storage, partial loss, and reconstruction. Its primary goal is to maintain the functional behaviour of numerical data when used in downstream computations, such as similarity evaluation or model inference.

In DTDR, numerical values are transformed and distributed across coefficients in a structured manner, typically using orthogonal transforms. The resulting representation is intended to serve as a *persistent memory format*, not merely as a compressed artefact. Reconstruction to a working numerical precision may occur, but it is not the defining endpoint of the representation. In some embodiments, computation may be performed directly in the transform domain without full reconstruction.

Several properties follow from this design choice. Information is distributed rather than localised, leading to smooth and proportional degradation under partial corruption. Similarity structure and inner products are approximately preserved, enabling meaningful comparison operations in the transformed domain. When reconstruction is performed, it is to a *working precision* sufficient to restore computational functionality rather than to reproduce the original numerical values exactly.

These characteristics are not accidental side effects of storage reduction. They are the result of treating the transform–domain representation as a first–class computational object. DTDR therefore occupies a different conceptual category from conventional compression: it is a representation designed for persistent storage *and* continued use, rather than a codec designed solely for efficient encoding and decoding.

4 Representation Versus Codec: A Structural Distinction

The fundamental distinction between DTDR and conventional compression schemes can be expressed in terms of *architectural role*. Compression methods function as *codecs*: they encode data into a compact form for storage or transmission and decode it back into an approximation of the original signal when needed. DTDR, by contrast, functions as a *persistent representation* intended to remain computationally meaningful over its lifetime.

This distinction can be made explicit by comparing the structural properties and intended use of each approach, as summarised in Table 1.

Several points in this comparison merit emphasis. First, while both approaches may employ transforms and quantisation, the optimisation objectives are orthogonal. Compression schemes aim to reduce the number of bits required to represent a signal, accepting loss of information so long as reconstruction quality remains within acceptable bounds. DTDR instead accepts numerical non-identity in service of preserving downstream computational behaviour.

Property	Compression Schemes	DTDR
Primary objective	Minimise storage or transmission cost	Preserve computational functionality
Optimisation target	Rate–distortion trade-off	Functional fidelity under computation
Role of transform domain	Intermediate encoding stage	Persistent memory representation
Reconstruction	End goal of the process	Optional or intermediate step
Direct computation on representation	Not intended or unstable	Supported in certain embodiments
Behaviour under partial loss	Often catastrophic or non-linear	Designed to degrade smoothly
Similarity structure preservation	Incidental	Explicitly preserved
Typical evaluation metrics	Bitrate, distortion, perceptual quality	Functional equivalence, robustness

Table 1: Structural comparison between conventional compression schemes and Distributed Transform–Domain Representation (DTDR).

Second, the transform domain in compression systems is a transient artefact: coefficients exist only to facilitate efficient encoding and are not designed to support meaningful computation. In DTDR, the transform domain is the *primary storage format*. As a result, properties such as distributed information content, robustness to partial corruption, and approximate preservation of inner products are design goals rather than side effects.

Finally, reconstruction occupies a fundamentally different role. In a codec, reconstruction completes the process; the compressed form has no further utility once decoded. In DTDR, reconstruction to a working numerical precision is a means of restoring compatibility with existing inference infrastructure, not the defining purpose of the representation itself. In some embodiments, reconstruction may be bypassed entirely.

These structural differences explain why DTDR cannot be meaningfully classified as a compression scheme, despite superficial similarities in the mathematical tools employed. DTDR should instead be understood as a memory architecture in which transform–domain persistence is a first–class computational feature.

5 Why JPEG and Related Codecs Are Not Prior Art for DTDR

Given that DTDR employs transform–domain representations and quantisation, it is natural to ask whether existing transform–based compression schemes, such as JPEG, JPEG–2000, or more recent learned codecs, constitute prior art for DTDR. This section explains why they do not, despite superficial similarities in mathematical technique.

The defining purpose of JPEG and related codecs is efficient encoding for storage or transmission. In these systems, transform–domain coefficients are explicitly designed as an intermediate representation whose sole role is to facilitate compression. The coefficients are optimised for energy compaction and entropy coding, not for preserving semantic structure or computational behaviour. As a result, the compressed representation is not intended to support meaningful computation, and any attempt to do so is incidental and unstable.

In particular, similarity relationships, inner products, and inference behaviour are not preserved in JPEG coefficient space. Operations performed directly on compressed coefficients generally require heuristic correction, full reconstruction, or retraining to recover useful results. This is not a limitation of specific implementations but a consequence of the optimisation objective: JPEG coefficients are chosen to minimise reconstruction error, not to preserve functional equivalence under downstream computation.

DTDR differs fundamentally in this respect. Its transform–domain representation is constructed explicitly to retain computational meaning. Similarity structure is approximately preserved, information is distributed across coefficients rather than localised, and degradation under partial loss is smooth rather than catastrophic. These properties are observed experimentally and arise from the architectural role of the representation, not as incidental byproducts.

Learned codecs and autoencoder–based compression systems share the same conceptual limitation. Although they may learn sophisticated nonlinear transforms, the latent representations they produce are optimised for reconstruction quality and bitrate. They are not designed to function as persistent, computation–ready representations, and their latent spaces do not generally support stable or interpretable computation without additional training or task–specific adaptation.

Crucially, none of these systems treat the transform domain as a first–class memory representation intended for continued use. Reconstruction is always the terminal step, and computational functionality is restored only after decoding. DTDR inverts this relationship by treating the transform–domain representation itself as the primary stored form, with reconstruction serving as an optional compatibility mechanism rather than the defining objective.

For these reasons, transform–based compression schemes, including JPEG and learned codecs, do not anticipate or subsume DTDR. While they may employ related mathematical tools, they address a different problem, optimise a different objective, and occupy a different architectural role. DTDR should therefore be evaluated as a memory representation rather than as a compression technique.

6 Experimental Evidence

The conceptual distinctions described in the preceding sections are supported by a set of focused experimental demonstrations. These experiments are designed not as performance benchmarks, but as evidence that DTDR behaves as a persistent, computation–preserving representation rather than as a conventional compression scheme.

First, DTDR representations of model parameters were shown to be reconstructible to a working numerical precision sufficient to restore functional inference behaviour. Although the reconstructed parameters are numerically non-identical to the original floating–point values, model outputs remain closely aligned, indicating preservation of computational functionality rather than bitwise fidelity.

Second, similarity search was demonstrated directly in the DTDR domain for embedded representations, without reconstruction to floating–point prior to evaluation. This behaviour is inconsistent with conventional compression schemes, whose compressed representations are not intended to support stable or semantically meaningful computation.

Third, controlled corruption of DTDR–stored parameters was observed to produce smooth, proportional degradation after reconstruction, rather than catastrophic failure. This behaviour reflects the distributed nature of the representation and contrasts with the fragility typically associated with localised encodings.

Finally, storage accounting demonstrates that DTDR can reduce the persistent storage footprint of numerical data while decoupling storage format from execution format. Reconstruction to a working numerical precision occurs as an explicit, one–time step and does not require specialised execution kernels.

Taken together, these experiments support the interpretation of DTDR as a memory representation designed to preserve computational utility under storage, partial loss, and reconstruction. They are difficult to reconcile with an interpretation of DTDR as a conventional compression scheme.

7 Conclusion

DTDR occupies a distinct conceptual and architectural category from transform-based compression techniques. While it may employ similar mathematical tools, such as orthogonal transforms and quantisation, it optimises a fundamentally different objective: the preservation of computational functionality rather than minimisation of storage subject to reconstruction error.

The key distinction lies in the role of the transform domain. In compression systems, the transform domain is a transient encoding stage whose purpose is efficient representation prior to decoding. In DTDR, the transform domain is the persistent storage format itself, designed to remain computationally meaningful over time.

This perspective clarifies why DTDR supports properties such as direct transform-domain computation, graceful degradation under partial loss, and reconstruction to a working numerical precision rather than exact numerical identity. These properties are not incidental side effects of compression, but consequences of treating the transform-domain representation as a first-class computational object.

DTDR should therefore be understood not as a compression scheme, but as a memory architecture in which transform-domain persistence enables robust, flexible, and computation-preserving storage of numerical data.