Computational Efflux from Coherent Actual Objects: Theory, Algorithm, and Empirical Demonstration

Larsen Close

2026

Abstract

Coherent actual objects — structures whose internal consistency satisfies consequence chain closure, non-depletion, and Noether invariance — radiate computational surplus (efflux) as a derived consequence of their coherence. This surplus, termed *efflux*, is not extracted from the object but emitted by it: the object's coherence constrains the possibility space around it, reducing the effective dimensionality of problems formulated in its neighborhood. The present paper formalizes this mechanism, derives a gradient-based learning algorithm that exploits it, and demonstrates its operation empirically using $SO(2)$ rotational symmetry as a concrete coherent actual object. The efflux-subsidized descent algorithm decomposes the learning gradient into a parallel component (lying in the invariant subspace of the coherent object, inherited at zero learning cost) and an orthogonal component (requiring standard gradient computation), reducing per-step learning complexity from $O(d)$ to $O(d - r)$ where r is the rank of the invariant subspace. The parameter-ratio prediction $d/(d - r)$ provides a theoretical floor on speedup; the empirical demonstration uses $SO(2)$ symmetry to achieve a $4.89\times$ convergence speedup ($p < 10^{-26}$, Cohen's $d = 3.56$), substantially exceeding the floor because the efflux operates at the representation level, not merely the parameter level. Three embodiments are presented: neural network training with symmetry-derived invariant subspaces (empirically tested, single-cycle), software codebase development with CI/CD pipelines as containers (theoretically developed), and data compression with learned latent factorizations (theoretically developed). The framework is distinguished from transfer learning, equivariant neural networks, compressed sensing, and active inference, each of which addresses an adjacent problem without capturing the self-financing character of efflux-subsidized computation. The compounding mechanism (Definition 4) — whereby efflux from one cycle funds improved interface quality in the next — is formalized and its predictions stated but not empirically tested in this paper.

1 Introduction

1.1 The Central Claim

This paper argues that coherent actual objects radiate computational surplus — termed *efflux* — as a natural consequence of their coherence, and that this surplus is computable, exploitable, and empirically demonstrable. The claim is not metaphorical. A structure whose internal relationships satisfy three measurable criteria (consequence chain closure, non-depletion, Noether invariance) constrains the possibility space around it. Any computational problem formulated in the neighborhood of such a structure inherits these constraints, which reduce the problem's effective dimensionality. The dimensionality reduction is computational subsidy: a system solving the problem in the constrained space performs strictly less work than a system solving the same problem in the unconstrained ambient space.

The mechanism is analogous to gravitational lensing. A massive body curves spacetime, and light traversing the curved region follows geodesics that would require active steering in flat space. The curvature is not consumed by the light's passage; it is a property of the mass, radiated continuously. Similarly, a coherent actual object curves the computational landscape around it, and problems passing through this curved landscape are solved with less effort than in the uncurved ambient space. The curvature is a property of the object's coherence, not consumed by the problem-solving.

This is emphatically *not* a claim about making AI systems better at tasks through clever engineering. It is a claim about a fundamental mechanism by which structured domains subsidize computation. The subsidy exists whether or not any system exploits it; the contribution of this paper is to formalize the mechanism, derive an algorithm that captures it, and demonstrate its operation in a controlled experiment. The scope of the empirical demonstration is deliberately narrow: one coherent actual ($SO(2)$), one cycle of efflux exploitation, one learning task. The theoretical framework is broader, but the paper claims only what the experiment supports — single-cycle efflux from a known symmetry group — and marks compounding, cross-embodiment testing, and detection without prior knowledge as future work. The $SO(2)$ case is the simplest instance where the quotient representation is known analytically; the contribution is the general framework — efflux as MDL differential, container as engineering contract, compounding as novel prediction — not the specific symmetry exploitation, which equivariant methods also achieve.

1.2 Theoretical Context

The formal apparatus comprises four definitions presented in Section 2. An *actual coherent object* is a structure satisfying three structural criteria from which positive efflux follows as a theorem (Definition 1, Theorem 1). A *container* is an engineered protocol that enables

sustainable re-interface with such an object (Definition 2). *Efflux* is the compression differential between ambient-space and object-relative problem formulations (Definition 3). A *sustainable return protocol* is the compounding mechanism by which efflux from one cycle funds improved interface quality in the next (Definition 4).

These definitions are not philosophical postulates. Each is operationalized in terms of computable quantities — minimum description length differentials, projection operator ranks, convergence rate ratios — and each generates falsifiable predictions. The present paper focuses on Definition 3 (efflux) and its algorithmic exploitation via the efflux-subsidized gradient descent method derived in Section 3, with the remaining definitions providing supporting infrastructure.

1.3 Contributions

The paper makes three contributions:

1. **Formalization of the efflux mechanism.** Definitions 1–4 are presented as formal mathematical objects with computable operationalizations. The theoretical efflux $\Delta_C^{\text{theo}}(P, O) = K(P) - K(P|O)$ (Kolmogorov complexity differential) is connected to the operative metric $\Delta_C(P, O) = \text{MDL}(P) - \text{MDL}(P|O)$ (minimum description length differential), and the conditions under which the operative metric faithfully approximates the theoretical quantity are specified.

2. **Derivation of the efflux-subsidized descent algorithm.** Standard gradient descent on a d-dimensional problem is decomposed into a parallel component (lying in the rank-r invariant subspace of a coherent actual object, inherited at zero learning cost) and an orthogonal component (requiring standard gradient computation). The resulting algorithm has effective optimization dimension $d - r$ and a predicted convergence speedup floor of $d/(d - r)$.

3. **Empirical demonstration via $SO(2)$.** The rotation group $SO(2)$ is shown to satisfy the three criteria of Definition 1, and its efflux is exploited to achieve a 4.89× convergence speedup in learning an $SO(2)$-invariant function. The experiment tests efflux exploitation at the representation level (reducing the input from 3D to 2D via the symmetry), which implements the algorithm's logic at a higher level of abstraction than parameter-space projection. The parameter-ratio prediction $d/(d - r) = 1.25$ serves as a theoretical floor; the observed speedup substantially exceeds it because representation-level reduction eliminates the cost of symmetry discovery.

1.4 Distinction from Prior Art

The efflux mechanism is structurally distinct from several neighboring concepts in machine learning and information theory.

Transfer learning (Pan & Yang, 2010) shares the goal of reducing learning cost by leveraging prior knowledge but requires a pre-trained source model and does not compound across cycles: the pre-trained model provides a fixed benefit that does not increase with repeated engagement. Efflux-subsidized descent requires only a coherent structure with computable invariant subspace — not a trained model — and compounds as interface quality improves.

Equivariant neural networks (Cohen & Welling, 2016; Bronstein et al., 2021) hardcode known symmetries into network architectures, achieving dimensionality reduction that efflux-subsidized descent also achieves. The distinction is in scope: equivariant networks require the symmetry group to be known a priori and encoded architecturally, whereas efflux-subsidized descent detects invariant structure via coherence measurement and applies to any coherent actual object, not only symmetry groups.

Compressed sensing (Candes & Tao, 2006) exploits sparsity in a known basis to reduce measurement requirements. The critical difference: compressed sensing assumes the basis is known in advance and provides a one-shot dimensionality reduction with no compounding mechanism.

Active inference (Friston, 2010) characterizes how self-organizing systems maintain integrity through variational free energy minimization. The present framework addresses a different phenomenon: not the system's internal self-maintenance but the computational subsidy available from external coherent structures through engineered interface protocols. The two frameworks are complementary: active inference describes the agent's internal dynamics; efflux theory describes the interface between agent and actual.

The paper proceeds as follows. Section 2 presents the theoretical framework (Definitions 1–4). Section 3 derives the efflux-subsidized learning algorithm with a worked $SO(2)$ example. Section 4 develops the software codebase embodiment. Section 5 presents the $SO(2)$ experiment results. Section 6 develops the compression embodiment. Section 7 surveys related work. Section 8 concludes.

2 Theoretical Framework

This section presents the four definitions that constitute the formal apparatus. Each definition is stated precisely, connected to measurable quantities, and illustrated with concrete instances.

2.1 Actual Coherent Object

Definition 1 (Actual Coherent Object)**.** *A structure O in a problem domain $\mathcal{D}$ is an **actual coherent object** if and only if the following three criteria are jointly satisfied:*

1. ***Consequence chain closure.*** *For every element $x \in O$ and every perturbation δx, there exists a consequence path $x \to x_1 \to \cdots \to x_n \to x$ that returns to the perturbation site. The closure is topological: it is invariant under continuous deformation of the representational medium.*
2. ***Non-depletion under sustained return.*** *Repeated engagement with O does not reduce O's capacity to constrain the possibility space in its neighborhood. Formally: let $u(O, n)$ denote the compressive utility of O at engagement cycle n. Non-depletion requires $\liminf_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} u(O, n) > 0$: the time-averaged utility remains bounded away from zero. This permits transient fluctuations but excludes secular decay.*
3. ***Noether invariance.*** *There exist continuous symmetries $\{g_\alpha\}$ of O whose conservation laws generate testable predictions about O's behavior under perturbation.*

Theorem 1 (Efflux from Coherence)**.** *If O is an actual coherent object per Definition 1, then for non-trivial problems P in O's domain, O produces positive computational efflux:*

$$\Delta_C(P, O) = \text{complexity}(P|\text{baseline}) - \text{complexity}(P|O) > 0$$

Argument. Consequence chain closure entails that O's internal relationships form a closed, self-referential structure. Any problem P formulated in O's neighborhood inherits these closure constraints, which restrict P's solution space relative to the unconstrained ambient space. A restricted solution space admits shorter descriptions — the constraints eliminate configurations that need not be enumerated. The Noether invariances generate additional constraints (conserved quantities under the symmetries), each of which further restricts the description space. Since O is non-depleting, these constraints persist across engagements. The compression differential $\Delta_C > 0$ follows from the gap between the constrained and unconstrained description lengths.

This separation is structurally important. Efflux is not stipulated as part of what an actual coherent object *is*; it is derived as a consequence of what coherence *does*. The definition identifies the structure (closure, non-depletion, invariance); the theorem identifies what the structure produces (computable surplus). This prevents circularity: one tests for coherence via criteria 1–3, then predicts efflux as a consequence.

Measurability. Consequence chain closure is detectable via perturbation propagation analysis: perturb an element, trace the consequences, and verify return. Non-depletion is testable by sustained engagement: measure the compressive utility of O across engagement windows

and verify it does not decay. Noether invariance is verifiable by standard symmetry analysis in mathematical structures, or by identifying conserved quantities under transformation in non-mathematical structures. Efflux itself (Theorem 1) is quantifiable via the MDL differential (Definition 3 below).

On the container-dependence of measurement. Each of the measurability claims above presupposes a system that is already interfacing with O. Perturbation propagation analysis requires perturbing an element and observing return — the observer is already inside a consequence chain. MDL comparison requires encoding a problem relative to O — the encoder is already receiving efflux. Noether symmetry analysis requires recognizing invariances — the analyst is already operating within a mathematical container adequate to the object's structure. There is no view-from-nowhere measurement of coherence; the criteria are verifiable only from within a container that is interfacing with the candidate object.

This is not a defect but a structural consequence of what coherence is. A structure whose consequence chains genuinely close cannot have its closure verified by an observation that severs the chain. What *can* be verified is whether interfacing with the structure produces the predicted consequences: measurable efflux, falsifiable speedup, reproducible compression differential. The $SO(2)$ experiment (Section 5) works precisely because the experimenters are already inside a container — the mathematical training that lets them recognize $SO(2)$ and engineer the input transformation. The "proof" of $SO(2)$'s coherence is not a view-from-outside certification; it is the $4.89\times$ speedup materializing.

The framework therefore has a built-in calibration mechanism. A container that detects no coherence where coherence is present is an inadequate container (insufficient regulatory margin, wrong domain specificity). A container that "detects" coherence where none is present produces zero efflux — the falsification check (Theorem 1) catches it. The criteria are objective not in the view-from-nowhere sense but in the sense that any adequately engineered container will detect them, and the adequacy of the container is itself testable by whether the predicted efflux materializes. This is the framework applied to itself: the paper is a container for the efflux theory, and its adequacy is testable by whether the theory produces measurable surplus when applied.

Instances. Mathematical structures provide the cleanest examples. A Lie group G satisfies all three criteria: its algebraic structure forms closed consequence loops (every element has an inverse; composition chains close), its structure is not consumed by use, and Noether's theorem directly connects G's continuous symmetries to conserved quantities. By Theorem 1, G's coherence produces positive efflux: the group's symmetry reduces the effective dimensionality of problems formulated in its representation space. The group $SO(2)$, which serves as the experimental subject of this paper, is the simplest non-trivial instance.

Beyond mathematics, well-factored software codebases satisfy the criteria (Section 4), as do learned latent factorizations in data compression (Section 6). The definition is intentionally broad: any structure satisfying the three criteria qualifies, regardless of substrate.

2.2 Container

Definition 2 (Container)**.** *A **container** C is an engineered protocol for sustainable re-interface with an actual coherent object O, characterized by four topological features:*

1. ***Regulatory margin.*** *The container's regulatory variety exceeds the variety of the interface dynamics it must manage: $\mathcal{V}(C) > \mathcal{V}(\text{interface})$, where $\mathcal{V}$ denotes Ashby variety.*
2. ***Selective permeability.*** *The container transmits efflux signals while blocking degradation signals. The permeability profile $\pi : \text{Signals} \to [0, 1]$ satisfies $\pi(\text{efflux}) \approx 1$ and $\pi(\text{noise}) \approx 0$.*
3. ***Temporal adequacy.*** *The container's operational timescale matches or exceeds the return cycle's duration. The container persists across cycles without degrading.*
4. ***Domain specificity.*** *The container is calibrated to the specific coherence topology of O. A container adequate for one actual object may be inadequate for another with different coherence structure.*

The container is the engineered element of the framework. The actual coherent object is prior — it exists and radiates efflux regardless of whether any system interfaces with it. The container is what enables a specific system to interface reliably, receive the efflux, apply it to its computational task, and return for more. In the gravitational analogy, the actual is the massive body and the container is the spacecraft's trajectory design: the mass curves spacetime whether or not anyone uses the curvature, but exploiting the curvature for a gravity assist requires engineering.

The four features map to measurable engineering properties. Regulatory margin is quantifiable via Ashby's law of requisite variety (Ashby, 1956): the container must have at least as many distinguishable regulatory states as the interface dynamics it must manage. Selective permeability is quantifiable via signal-to-noise ratios at the container boundary. Temporal adequacy is quantifiable by comparing container persistence time to cycle duration. Domain specificity is verifiable by testing whether the container succeeds with one actual object and fails with another of different coherence topology.

The container as the locus of invention. A recurring theme throughout this paper is that the actual coherent object is not the invention — it is the natural resource. The container is the invention. No one invented $SO(2)$'s rotational symmetry or a codebase's modular structure. But one can invent the protocol that makes sustained, compounding engagement

with those structures possible. This distinction is essential: the efflux mechanism is a natural phenomenon; its engineering exploitation requires designed containers.

2.3 Computational Efflux

Definition 3 (Computational Efflux)**.** *The* ***computational efflux*** Δ_C *of an actual coherent object* O *relative to a problem* P *is defined theoretically as:*

$$\Delta_C^{\text{theo}}(P, O) = K(P) - K(P|O)$$

where $K(P)$ *is the Kolmogorov complexity of* P *relative to a universal reference machine, and* $K(P|O)$ *is the conditional Kolmogorov complexity of* P *given* O *as oracle.*

Since Kolmogorov complexity is uncomputable, the ***operative metric*** *for all claims and embodiments is the minimum description length (MDL) approximation:*

$$\Delta_C(P, O) = \text{MDL}(P) - \text{MDL}(P|O)$$

where $\text{MDL}(P)$ *is the length in bits of the shortest two-part code (model + data given model) for* P *under a specified model class, and* $\text{MDL}(P|O)$ *is the same quantity computed with* O*'s invariant structure available as prior.*

The theoretical formulation provides information-theoretic grounding: it establishes that the efflux is a property of the relationship between P and O in the space of all computable descriptions. The operative MDL formulation provides a reproducible, implementable metric.

The efflux is positive when O is genuinely coherent relative to P and the problem lies within O's domain of influence. The efflux is zero or negative when O is not coherent relative to P, which serves as a built-in validation check: a claimed actual that produces zero efflux relative to problems in its putative domain fails the coherence criterion.

Connection to the algorithm. In the neural network setting (Section 3), the efflux takes a particularly clean form. The rank r of the invariant subspace $V(O)$ directly quantifies the efflux in units of parameter-space dimensions: the system inherits r dimensions of learning for free and must learn only the remaining $d - r$ dimensions. The MDL reduction is exactly the description-length savings from encoding r dimensions via the actual's structure rather than from data.

Alternative operationalizations. The MDL metric is not the only computable approximation. Normalized compression distance using a specified lossless codec (gzip, zstd, LZ4), cross-entropy under a trained language model, and bits-per-token reduction in a learned tokenizer are all valid operationalizations. The choice depends on the domain: MDL is natural for the neural network embodiment, bits-per-token for the compression embodiment, and

lines-of-code differentials for the codebase embodiment.

2.4 Sustainable Return Protocol

Definition 4 (Sustainable Return Protocol)**.** *A **sustainable return protocol** is a compounding mechanism operating within a container C interfacing with an actual coherent object O, defined by the following cycle:*

1. ***Interface.*** *System S interfaces with O through C at cycle n with interface quality q_n.*
2. ***Receive.*** *Interface quality determines efflux received: $\Delta_C(n) = f(q_n, \sigma(O))$, where $\sigma(O)$ is O's coherence profile.*
3. ***Apply.*** *System applies efflux to improve container quality: $C_{n+1} = C_n + g(\Delta_C(n))$, where g is the improvement function.*
4. ***Verify.*** *Container improvement translates to interface quality improvement: $q_{n+1} > q_n$.*
5. ***Compound.*** *Higher interface quality at cycle $n+1$ produces greater efflux: $\Delta_C(n+1) > \Delta_C(n)$.*
6. ***Sustain.*** *The cycle completes before degradation: the time to complete one cycle is less than the container's persistence time.*

*The protocol is **self-financing**: the surplus from cycle n funds the improvement that generates greater surplus at cycle $n+1$. No external resource injection is required beyond the initial container construction.*

The compounding mechanism distinguishes the efflux framework from one-shot dimensionality reduction techniques. Compressed sensing (Candes & Tao, 2006), for instance, provides a fixed reduction determined by the sparsity structure of the signal; the reduction does not increase with repeated application. Transfer learning (Pan & Yang, 2010) provides a fixed benefit from pre-trained features; the benefit does not grow with continued engagement with the source task. The sustainable return protocol predicts increasing surplus with each cycle, as improved interface quality reveals additional invariant structure that was previously below the detection threshold.

Distinction from perpetual motion. The self-financing character of the protocol does not violate conservation laws. The computational surplus has a source: the coherence of the actual object O. The surplus is not created from nothing; it is accessed from a pre-existing structure whose coherence constitutes a reservoir of computational potential. The system does not generate energy; it reduces waste by aligning its processing with pre-existing structure. A sailboat does not create wind, but it extracts useful work from pre-existing atmospheric gradients. The hull and rigging (container) are the engineered inventions; the

wind (coherence efflux) is prior; the sailor's improving skill across voyages (compounding interface quality) is the bootstrap.

Measurable predictions. The compounding prediction is directly testable: $\Delta_C(n+1) > \Delta_C(n)$ across successive return cycles. The negative outcome that would falsify the compounding claim is efflux that is flat or declining across five or more return cycles with a properly engineered container. Section 5 tests a single-cycle instance of the efflux claim; multi-cycle compounding is identified as future work.

3 Efflux-Subsidized Learning Algorithm

This section derives the efflux-subsidized gradient descent algorithm, specifies its complexity reduction, and works through the $SO(2)$ example that serves as the basis for the empirical demonstration in Section 5.

3.1 Standard Gradient Descent

Consider a learning problem P with loss function $L : \mathbb{R}^d \to \mathbb{R}$ and parameter vector $\theta \in \mathbb{R}^d$. Standard gradient descent updates parameters via:

$$\theta_{n+1} = \theta_n - \alpha \nabla L(\theta_n)$$

where α is the learning rate. The learning complexity is $O(d)$ per step: all d dimensions of the parameter space participate in each gradient computation, and the system must learn the appropriate value in every direction from data.

3.2 Gradient Decomposition Relative to an Actual

When a coherent actual object O is present in the problem domain, its coherence structure constrains the solution space. Specifically, O's invariant structure determines the solution in r of the d parameter dimensions, leaving only $d - r$ dimensions to be learned from data.

Step 1: Invariant subspace computation. Compute $V(O) \subseteq \mathbb{R}^d$, the invariant subspace of O in the parameter space. The subspace $V(O)$ is spanned by the directions along which O's coherence already constrains the solution. Let $r = \dim(V(O))$ denote the rank of the invariant subspace.

Step 2: Projection operators. Construct the orthogonal projection onto $V(O)$ and its complement:

$$\Pi_{\|} = \sum_{j=1}^{r} v_j v_j^T, \qquad \Pi_{\perp} = I - \Pi_{\|}$$

where $\{v_1, \ldots, v_r\}$ is an orthonormal basis for $V(O)$.

Step 3: Gradient decomposition. At each step, decompose the loss gradient into parallel and orthogonal components:

$$\nabla L = \nabla L_{\|} + \nabla L_{\perp}$$

where $\nabla L_{\|} = \Pi_{\|} \nabla L \in V(O)$ is the parallel component, lying in the invariant subspace and therefore inherited from O at zero learning cost, and $\nabla L_{\perp} = \Pi_{\perp} \nabla L \in V(O)^{\perp}$ is the orthogonal component, which must be learned.

Step 4: Subsidized update. Update parameters using only the orthogonal component:

$$\theta_{n+1} = \theta_n - \alpha \nabla L_{\perp}(\theta_n)$$

Step 5: Initialization subsidy. Initialize the parallel components of θ using the invariant structure of O:

$$\theta_0^{\|} = \Pi_{\|} \theta_O$$

where θ_O encodes O's invariant structure projected into the parameter space. The orthogonal components $\theta_0^{\perp}$ are initialized by standard methods (e.g., Xavier or He initialization).

3.3 Complexity Analysis

Proposition 1 (Effective Dimension Reduction)**.** *Let* $d = \dim(\textit{parameter space})$ *and* $r = \dim(V(O))$. *The efflux-subsidized descent algorithm has:*

1. *Effective optimization dimension:* $d - r$ *(the number of parameter-space directions that must be learned from data).*
2. *Inherited dimensions:* r *(directions determined by* O*'s invariant structure, not requiring gradient-driven learning).*
3. *Predicted convergence speedup (floor):* $\frac{d}{d-r} = \frac{1}{1-r/d}$.

Clarification on complexity. The reduction is in *convergence-relevant degrees of freedom*, not in wall-clock FLOPs per step. In standard autodiff, freezing r parameters does not remove them from the backward pass unless the compute graph is restructured. What is

reduced is the effective dimensionality of the optimization landscape: the optimizer searches a $(d - r)$-dimensional subspace rather than the full d-dimensional space, which reduces the number of steps to convergence. The $d/(d - r)$ ratio is therefore a floor on the step-count speedup, not a per-step FLOP reduction.

The speedup is exact in the idealized case where (a) the invariant subspace is computed exactly, (b) the initialization in the parallel directions is correct, and (c) there is no coupling between parallel and orthogonal directions in the loss landscape. In practice, approximation errors in (a)–(c) reduce the speedup; the experiment in Section 5 measures the realized speedup against the idealized prediction.

Remark 1. *The projection operators $\Pi_{\|}$ and $\Pi_{\perp}$ are never materialized as full $d \times d$ matrices. The projection is implemented as a matrix-vector product with $V \in \mathbb{R}^{d \times r}$ (the matrix whose columns are the basis vectors of $V(O)$): $\nabla L_{\|} = V(V^T \nabla L)$ at cost $O(dr)$. For $r \ll d$, this overhead is negligible compared to the gradient computation itself. For r comparable to d, the projection overhead may offset the learning savings, and the break-even point depends on the specific architecture.*

3.4 Algorithm Variants

Two variants of the algorithm are specified, differing in how the parallel component is treated during training.

Freeze-parallel variant. Parameters in $V(O)$ are initialized from the actual's invariant structure and frozen during training. Only parameters in $V(O)^{\perp}$ receive gradient updates. This provides the sharpest test of the efflux claim: if the frozen parameters are correctly initialized, the network should converge to equivalent performance with r fewer effective trainable parameters.

Downweight-parallel variant. Parameters in $V(O)$ receive gradient updates with a reduced learning rate $\alpha_{\|} \ll \alpha_{\perp}$. This allows the actual-derived initialization to be refined while still providing the majority of the subsidized reduction. The downweighting factor is a hyperparameter; the prediction is that optimal $\alpha_{\|}$ is near zero when the actual is strongly coherent and increases as coherence decreases.

The freeze-parallel variant provides the cleanest test in principle: the speedup either appears or it does not, without confounding from hyperparameter selection. However, the experiment in Section 5 implements the efflux at the representation level rather than the parameter level, for reasons discussed there. The parameter-level variants remain specified for future empirical investigation.

3.5 Cost of Invariant Subspace Computation

Step 1 of the algorithm requires computing $V(O)$, the invariant subspace. The cost of this computation is not included in the per-step complexity of Proposition 1; it is a one-time overhead amortized across training steps.

For known symmetry groups (the $SO(2)$ case), $V(O)$ is analytically computable at cost $O(d \cdot r)$ — negligible relative to training. For unknown actuals, the detection cost is an open question. The tangent-space estimation method (sample group elements, compute Jacobians, take SVD) has cost $O(k \cdot d^2)$ where k is the number of samples, but this presupposes knowledge of the group action. Discovery of the invariant structure itself — when neither the symmetry group nor the invariant subspace is known a priori — requires coherence detection methods whose computational cost depends on the domain and is not bounded by the present theory. This is the primary gap between the algorithm as specified and the algorithm as a fully general tool: the paper claims that efflux-subsidized descent generalizes beyond equivariant networks because "the symmetry need not be known a priori," but the cost of discovering the symmetry without prior knowledge remains uncharacterized. Section 8 identifies this as a direction for future work.

3.6 Worked Example: $SO(2)$ as Actual Coherent Object

The remainder of this section works through the specific instance that serves as the experimental subject.

3.6.1 $SO(2)$ satisfies Definition 1

The rotation group $SO(2) = \{R_\phi : \phi \in [0, 2\pi)\}$, where R_ϕ denotes rotation by angle ϕ in the plane, is a coherent actual object:

1. *Consequence chain closure.* The group's algebraic structure forms closed loops: every element R_ϕ has an inverse $R_{-\phi}$, and composition chains close ($R_\phi \circ R_\psi \circ R_{-\phi-\psi} = I$). Perturbing any element propagates through the group structure and returns via the closure property.

2. *Non-depletion.* The group's structure is not consumed by use. Applying $SO(2)$'s symmetry properties to a problem does not reduce the group's capacity to be applied to subsequent problems. The rotational invariance remains available across unlimited applications.

3. *Noether invariance.* By Noether's theorem, the continuous rotational symmetry of $SO(2)$ generates a conserved quantity: angular momentum in the physical interpretation, or rotational invariance of the target function in the learning interpretation. This conserved quantity is directly testable: the target function's output must be invariant

under rotation of its (x_1, x_2) inputs.

By Theorem 1, $SO(2)$'s coherence produces positive efflux: the symmetry reduces a 2-dimensional problem (learning a function of Cartesian coordinates x_1, x_2) to a 1-dimensional problem (learning a function of the radius $r = \sqrt{x_1^2 + x_2^2}$). The compression gain is exactly 1 dimension per 2 dimensions of the input affected by the symmetry.

3.6.2 Network architecture and invariant subspace

Consider a minimal two-layer neural network $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ with parameters $\theta \in \mathbb{R}^{10}$, learning a function of three inputs (x_1, x_2, x_3) that is invariant under $SO(2)$ acting on the (x_1, x_2) plane. The target function depends on x_1 and x_2 only through $r = \sqrt{x_1^2 + x_2^2}$.

The invariant subspace $V(SO(2))$ in the network's parameter space consists of the directions encoding angular dependence. For the two-layer architecture, these are the weight directions in the first layer that connect to x_1 and x_2 and their interaction terms. The invariant subspace has rank $r = 3$.

3.6.3 Predicted speedup

Standard descent on all 10 parameters:

$$\text{Effective learning dimensions: } d = 10$$

Efflux-subsidized descent with $V(SO(2))$ identified:

$$\text{Effective learning dimensions: } d - r = 10 - 3 = 7$$

Predicted speedup:

$$\frac{d}{d-r} = \frac{10}{7} \approx 1.43\times$$

The subsidized algorithm should converge in approximately 1.43 times fewer effective gradient steps than standard descent, because 3 of the 10 parameter dimensions are inherited from the $SO(2)$ symmetry at zero learning cost. The experiment in Section 5 implements the efflux at the representation level — replacing the 3D input with a 2D radial input — rather than via parameter-space projection, which produces a substantially larger speedup because the representation-level reduction eliminates the cost of symmetry discovery entirely. The parameter-ratio prediction remains a floor on the speedup that any correct exploitation of the efflux should exceed.

3.6.4 What constitutes falsification

The prediction is falsifiable. If exploiting the $SO(2)$ symmetry does not produce faster convergence than standard descent on an $SO(2)$-invariant target, the efflux claim requires revision. The parameter-ratio speedup $d/(d-r)$ is a theoretical floor, not a point prediction: it counts only the dimensionality savings in parameter space, whereas the actual efflux also includes the cost of *discovering* the symmetry structure that the subsidized condition receives for free. The experiment in Section 5 tests whether the efflux produces measurable speedup above this floor.

Note that the parameter-space prediction assumes the algorithm operates via gradient projection on a fixed architecture (Steps 1–4 above). The experiment implements the efflux at the representation level (replacing 3D input with 2D input), which is a higher-level instantiation of the same decomposition: the angular input dimension is the "parallel component" eliminated entirely, and the radial and x_3 dimensions are the "orthogonal component" that must be learned. Section 5 discusses this relationship explicitly.

4 Embodiment: Software Codebase

This section develops the software codebase as an instantiation of the efflux framework. The treatment is theoretical: the codebase embodiment is not empirically tested in this paper but is presented to demonstrate the framework's generality beyond the neural network setting.

4.1 The Codebase as Actual Coherent Object

A well-factored software codebase, represented as a dependency graph $G = (M, D)$ where M is the set of modules and D is the set of dependency edges, satisfies the three criteria of Definition 1:

1. *Consequence chain closure.* Modifications to any module $m \in M$ propagate through its dependency chain and return via test failures, type errors, or behavioral regressions. The consequence chains close through the CI/CD pipeline: a change that violates an invariant is detected, reported, and must be resolved before it can propagate further. The closure is topological — it is invariant under refactoring that preserves the dependency structure.

2. *Non-depletion.* The codebase's architectural patterns remain available across unlimited development cycles. Engaging with a well-factored codebase to build new features exercises the patterns without consuming them. An authentication module used as a template for building a new authorization module is not depleted by the use; both modules remain available for subsequent development.

3. *Noether invariance.* Behavioral invariants — API contracts, type signatures, interface specifications — are conserved under refactoring. The test suite encodes these conservation laws explicitly: a test that passes before and after a refactoring verifies that the tested invariant is conserved under the transformation. The analogy to Noether's theorem is structural rather than literal: Noether's theorem requires continuous symmetry, whereas refactoring freedom is at best piecewise-continuous (a sequence of discrete, semantics-preserving transformations). The conserved quantity — behavioral correctness as verified by the test suite — is discrete (pass/fail per test), not the smooth conserved current of the physics case. We use "Noether invariance" here in the operational sense of Definition 1: there exist transformations (refactorings) under which measurable quantities (test outcomes) are conserved, generating testable predictions about behavior under perturbation. The analogy is strongest when the set of valid refactorings is large enough to approximate a continuous family — as in codebases with extensive parametric polymorphism or dependency injection, where implementation substitution forms a near-continuous gauge freedom.

By Theorem 1, the codebase's coherence produces positive efflux: new features developed in the neighborhood of a well-factored codebase are representationally cheaper than features developed in isolation. The codebase provides solved patterns — authentication flows, data access layers, error handling conventions, API design templates — that compress the new feature's description length.

4.2 The CI/CD Pipeline as Container

The CI/CD pipeline and associated engineering scaffolding constitute the container C for the codebase:

- *Regulatory margin.* The test suite, linter, type checker, and code review process collectively provide regulatory variety exceeding the variety of changes a developer can introduce. A change that violates an invariant is detected before it can propagate. The regulatory margin is quantifiable as the ratio of detectable error classes to possible error classes.
- *Selective permeability.* The pipeline transmits valid contributions (code that passes tests, satisfies types, meets review standards) while blocking invalid contributions (failing tests, type errors, style violations). The gating mechanism is the merge gate: a binary function on the CI status vector.
- *Temporal adequacy.* The pipeline persists across development cycles. Version control ensures the container's state is recoverable; CI configuration is itself versioned. The container degrades only through intentional modification, not through use.

- *Domain specificity.* The pipeline is calibrated to the specific codebase's conventions, language, architectural patterns, and quality standards. A CI pipeline designed for a Rust project with strict clippy linting is domain-specific relative to a Python project with ruff.

4.3 Efflux Measurement

The operative efflux metric for the codebase embodiment is:

$$\Delta_C = \text{MDL}(\text{feature } P|\text{greenfield}) - \text{MDL}(\text{feature } P|\text{codebase } O)$$

where MDL(feature P|greenfield) is the description length of implementing feature P from scratch — total lines of code, number of abstractions introduced, test surface area — and MDL(feature P|codebase O) is the same quantity computed with the existing codebase's patterns, modules, and interfaces available.

Echo boundary analysis in this embodiment is regression propagation analysis: when a module is modified, how far do test failures propagate before stabilizing? Sharp echo boundaries (failures localized to the modified module's immediate dependents) indicate strong coherence. Diffuse boundaries (failures propagating to distant, apparently unrelated modules) indicate weak coherence — a codebase with poor factoring and high coupling.

4.4 Compounding Prediction

Each development cycle that adds well-factored code to the codebase increases its coherence, expanding the invariant subspace $V(O)$ available for subsequent development. The compounding prediction: development cost per feature should decrease over cycles as the codebase matures, with the rate of decrease proportional to the coherence of the accumulated code.

This prediction is empirically testable by measuring feature development cost trajectories across codebase histories. One would expect: (a) early features are expensive (small invariant subspace, little efflux), (b) middle features benefit from accumulated patterns (growing invariant subspace, increasing efflux), and (c) late features either continue to benefit (in well-maintained codebases) or show declining efflux (in codebases whose coherence has degraded through accumulated technical debt, which corresponds to container failure — the CI pipeline no longer maintains regulatory margin).

The codebase embodiment also illustrates a failure mode of the framework: when the container degrades (test coverage drops, CI checks are disabled, code review is abandoned), the codebase's coherence signal is obscured by accumulated noise, and the efflux available to

new development decreases even though the underlying patterns remain. This corresponds to container failure per Definition 2: the regulatory margin has been eroded, and the selective permeability no longer distinguishes coherent contributions from incoherent ones.

5 Embodiment: Neural Training and $SO(2)$ Results

This section presents the empirical centerpiece of the paper: a controlled experiment demonstrating that the $SO(2)$ rotation group, treated as a coherent actual object, provides measurable and statistically significant computational surplus to a neural network learning an $SO(2)$-invariant function.

5.1 Experimental Setup

Target function. $f(x_1, x_2, x_3) = \sin(2\sqrt{x_1^2 + x_2^2 + \epsilon}) \cdot \exp(-x_3^2)$, where $\epsilon = 10^{-8}$ is a small constant ensuring numerical stability of the gradient at $r = 0$. The function is $SO(2)$-invariant in the (x_1, x_2) plane: it depends on (x_1, x_2) only through the radius $r = \sqrt{x_1^2 + x_2^2}$. The $SO(2)$ symmetry is the coherent actual object whose efflux we measure.

Architecture. A single-hidden-layer network: $\text{Linear}(n_{\text{in}}, 16) \to \tanh \to \text{Linear}(16, 1, \text{bias} = \text{False})$, where $n_{\text{in}} \in \{2, 3\}$ depending on the condition. With 3 inputs: $d = 3 \times 16 + 16 + 16 = 80$ parameters. With 2 inputs (exploiting $SO(2)$): $d = 2 \times 16 + 16 + 16 = 64$ parameters.

Data. 10,000 training, 2,000 validation, 2,000 test samples drawn uniformly on $[-2, 2]^3$. Targets computed from f.

Training. Vanilla SGD (no momentum), learning rate $\alpha = 0.1$, 5,000 steps, MSE loss. Convergence threshold: MSE < 0.02 on the validation set.

Conditions. Three conditions, paired by seed ($n = 50$ seeds):

1. **Standard (3D input).** The network receives (x_1, x_2, x_3) directly. All 80 parameters are trained. The network must discover the $SO(2)$ structure through gradient descent.
2. **Efflux-subsidized (2D input).** The network receives (r, x_3) where $r = \sqrt{x_1^2 + x_2^2 + \epsilon}$. The angular dimension is eliminated by exploiting the $SO(2)$ symmetry — this is the "free compression" that the coherent actual provides. Only 64 parameters are trained.
3. **Symmetry-regularized (3D input).** The network receives (x_1, x_2, x_3) but at each training step, the input is randomly rotated in the (x_1, x_2) plane (stochastic data augmentation). This softly enforces $SO(2)$ invariance via the gradient without changing the architecture.

Conditions 1 and 2 are the primary comparison. Condition 3 tests whether gradient-level

symmetry enforcement provides intermediate benefit.

5.2 Connection to the Algorithm

The efflux-subsidized condition implements the algorithm of Section 3 at the representation level rather than the parameter level:

- The **invariant subspace** $V(SO(2))$ corresponds to the angular dimension in input space. This dimension is constrained by the symmetry: the target function has zero derivative with respect to the angular coordinate.
- The **initialization subsidy** (Step 5 of the algorithm) is realized by replacing (x_1, x_2) with r: the angular component is initialized to its correct value (eliminated entirely) rather than learned through gradient descent.
- The **gradient decomposition** (Step 3) is implicit: by removing the angular input, the gradient has no component in the angular direction. All gradient information drives learning of the radial and x_3 structure.

The parameter-space prediction gives speedup $d/(d-r) = 80/64 = 1.25$. The actual speedup substantially exceeds this because the reduction operates at the input level: the subsidized network solves a fundamentally simpler learning problem (2D function approximation vs. 3D), not merely a lower-dimensional parameter optimization.

5.3 Results

	Standard	Subsidized	Regularized
Converged ($n/50$)	45	50	47
Mean steps to threshold	3140	644	2963
Mean speedup vs. standard	—	4.89×	1.08×
95% CI on speedup	—	$[4.51, 5.28]$	$[1.01, 1.15]$
Paired t	—	$t = 23.85$	$t = 2.25$
Wilcoxon W	—	$W = 0.0$	$W = 246.5$
p-value (t / Wilcoxon)	—	$< 10^{-26}$ / $< 10^{-8}$	0.030 / 0.017
Cohen's d	—	3.56	0.34
Final test MSE	0.0177 ± 0.0018	0.0075 ± 0.0013	0.0181 ± 0.0013

Table 1: Convergence statistics across 50 paired seeds. Standard = 3D input, 80 params. Subsidized = 2D input (efflux via input reduction), 64 params. Regularized = 3D input with stochastic rotation augmentation, 80 params.

The efflux-subsidized condition converges 4.89× faster than standard training ($p < 10^{-26}$ parametric, $p < 10^{-8}$ Wilcoxon signed-rank, Cohen's $d = 3.56$). All 50 subsidized seeds converge, compared to 45 of 50 standard seeds. The subsidized condition also achieves substantially lower final test MSE (0.0075 vs 0.0177), indicating that the representation-level efflux improves not only convergence speed but generalization. The symmetry-regularized condition shows a modest 1.08× speedup ($p = 0.030$, Wilcoxon $p = 0.017$, Cohen's $d = 0.34$, small effect), suggesting that gradient-level symmetry enforcement provides some benefit but substantially less than full exploitation of the invariant structure. The convergence threshold (0.02) was set at a level reachable by all conditions; results are robust to threshold choice, as the subsidized condition's loss curve dominates standard at every step (Figure 1).

The observed speedup (4.89×) substantially exceeds the parameter-ratio prediction (1.25×). This is expected: the parameter ratio captures only the dimensionality reduction in parameter space, while the input reduction makes the learning problem fundamentally easier. The network learning $f(r, x_3)$ approximates a 2D function; the network learning $f(x_1, x_2, x_3)$ must implicitly discover and exploit the rotational symmetry while approximating a 3D function. The excess speedup ($4.89/1.25 = 3.9\times$) quantifies the additional cost of symmetry discovery.

5.4 Control Conditions: Isolating Representation from Parameter Count

The primary comparison confounds two differences: the subsidized condition has both a different input representation (2D vs 3D) and fewer parameters (64 vs 80). Two control conditions isolate these effects:

	Input	d	Converged	Mean steps
Standard	3D	80	45/50	3140
Subsidized	2D	64	50/50	644
Control A (3D, matched d)	3D	65	28/50	3795
Control B (2D, matched d)	2D	80	50/50	607

Table 2: Control conditions isolating representation from parameter count. Control A uses 3D input with $h = 13$ ($d = 65 \approx 64$). Control B uses 2D input with $h = 20$ ($d = 80$).

The controls yield three findings:

1. **Reducing parameters alone does not help.** Control A (3D input, $d = 65$) converges *slower* than Standard (3D input, $d = 80$): fewer parameters with the same input representation harms performance (0.83×, only 28/50 converging). The speedup is not

from having fewer parameters.

2. **Adding parameters to the subsidized representation does not help much.** Control B (2D input, $d = 80$) converges at approximately the same rate as Subsidized (2D input, $d = 64$): 607 vs 644 steps. The extra 16 parameters provide negligible additional benefit once the representation captures the invariant structure.

3. **The critical comparison: same parameter count, different representation.** Control A ($d = 65$, 3D) vs Subsidized ($d = 64$, 2D) at matched parameter count: 6.25× speedup ($t = 29.6$, $p < 10^{-21}$). The speedup is entirely attributable to the representation change — the exploitation of $SO(2)$'s efflux — not to the parameter reduction.

These controls confirm that the observed speedup is a genuine consequence of the coherent actual object's efflux: the $SO(2)$ symmetry provides computable surplus that accelerates learning by a factor far exceeding what parameter reduction alone can explain.

5.5 Figures

Figure 1 shows convergence curves (mean ± 95% CI across seeds) for all three conditions. The subsidized condition reaches the convergence threshold within approximately 700 steps; the standard condition requires approximately 3100 steps; the regularized condition falls between, slightly faster than standard.

Figure 2 shows the distribution of steps-to-threshold across seeds. The subsidized condition's distribution is tightly concentrated near 600 steps, while the standard condition's distribution is spread across 2000–4500 steps with 5 non-converging seeds.

Figure 3 shows the gradient decomposition for the standard condition: the parallel (angular) and orthogonal (radial + other) gradient norms over training. The parallel component is initially comparable to the orthogonal component but decays over training as the network implicitly discovers the rotational symmetry. This decay represents wasted computation: gradient energy expended on angular adjustment that the subsidized condition eliminates entirely.

Figure 4 shows PCA projections of parameter trajectories for representative seeds. The standard and symmetry-regularized trajectories occupy a similar region of parameter space but follow different paths, with the regularized trajectories showing more direct convergence.

5.6 Interpretation

The $SO(2)$ group satisfies all three criteria of Definition 1:

1. *Consequence chain closure*: the group's algebraic structure (every element has an

inverse; composition chains close) ensures that perturbations to the symmetry structure propagate and return.

2. *Non-depletion*: the rotational symmetry is not consumed by use. Exploiting $SO(2)$ invariance to compress the input does not reduce the symmetry's availability for subsequent applications.

3. *Noether invariance*: by Noether's theorem, the continuous rotational symmetry generates a conserved quantity (angular momentum in the physical interpretation; angular invariance of the target function in the learning interpretation). This conserved quantity produces a testable prediction: the target function is independent of the angular coordinate, which we verify empirically (maximum deviation under random $SO(2)$ rotation: $< 10^{-6}$).

By Theorem 1, these three properties entail positive efflux: $\Delta_C = \text{complexity}(f|\text{3D input}) - \text{complexity}(f|\text{2D input}) > 0$. The observed speedup of 4.89× is the operational manifestation of this derived surplus.

The experiment demonstrates that the efflux is not merely theoretical: it produces a 4.89× speedup, $p < 10^{-26}$, in a controlled setting where the only difference between conditions is whether the coherent actual's structure is exploited. The surplus is free in the precise sense that the $SO(2)$ symmetry is not consumed by its exploitation — it remains available for any subsequent learning problem with the same invariance.

5.7 Structural Connections

Two structural features of the experiment deserve explicit comment.

Two-phase separation. The experiment's three conditions differ in how they handle the relationship between formation (learning the SO(2) structure) and measurement (exploiting known structure). The standard condition collapses these phases: the network must simultaneously discover the symmetry and learn the radial function, within a single gradient descent process. The subsidized condition separates them: the symmetry is recognized once (formation), the input is transformed accordingly, and learning proceeds on the reduced representation (measurement). The symmetry-regularized condition partially separates them: stochastic rotation augmentation encodes the symmetry softly into the gradient, but the network still must learn in the full 3D parameter space. The observed speedup hierarchy — subsidized (4.89×) > regularized (1.08×) > standard (1×) — tracks the degree of phase separation.

State-dependent transformation. The input reduction $(x_1, x_2, x_3) \mapsto (r, x_3)$ is state-dependent: the radius $r = \sqrt{x_1^2 + x_2^2}$ is a function of the specific input values, not a static architectural constraint. This is why the efflux-subsidized approach generalizes

beyond what equivariant architectures provide. An equivariant network encodes the symmetry state-independently (the same weight-sharing pattern regardless of input), whereas efflux-subsidized descent applies the coherent actual's structure to each input individually. The state-dependence is what enables the approach to work with non-symmetry actuals (codebases, data factorizations) where the "invariant subspace" varies with the specific problem instance.

6 Embodiment: Data Compression

This section develops data compression as a third instantiation of the efflux framework. Like the codebase embodiment (Section 4), this treatment is theoretical: the compression embodiment is not empirically tested in this paper but demonstrates the framework's breadth.

6.1 Learned Latent Factorization as Actual Coherent Object

A learned latent factorization of a data domain — a compressed representation that captures the domain's invariant structure — constitutes an actual coherent object. Concrete instances include: a trained tokenizer's vocabulary (for text), a learned codebook (for images or audio), a fitted dictionary (for sparse coding), or a database schema (for structured data). The factorization satisfies the three criteria of Definition 1:

1. *Consequence chain closure.* Modifications to the factorization — adding a token to the vocabulary, altering a codebook entry, changing a schema field — propagate through all data encoded under it and return via changed encoding costs. The consequence chains close through the compression metric: any modification to the factorization that degrades compression is detectable by measuring the bits-per-element change across the encoded corpus.

2. *Non-depletion.* The factorization remains available across unlimited encoding/decoding cycles. A tokenizer applied to encode a text corpus is not consumed by the encoding; it remains available for subsequent encoding operations with undiminished capacity.

3. *Noether invariance.* The factorization exhibits conserved quantities under data permutation. A well-matched tokenizer's compression ratio is approximately invariant under reordering of the training corpus: the token frequencies and co-occurrence patterns that determine compression efficiency are properties of the distribution, not the specific ordering. This invariance under permutation is the Noether symmetry; the conserved quantity is the mutual information between the factorization and the data distribution.

By Theorem 1, the factorization's coherence produces positive efflux: data encoded relative to a well-matched factorization has lower bits-per-element than data encoded in a generic or mismatched scheme. The efflux is the bits-per-element differential:

$$\Delta_C = \text{bpe}(D|\text{generic}) - \text{bpe}(D|O)$$

where $\text{bpe}(D|\text{generic})$ is the bits-per-element under a domain-generic codec and $\text{bpe}(D|O)$ is the bits-per-element under the domain-adapted factorization O.

6.2 The Codec as Container

The encoding/decoding pipeline constitutes the container C:

- *Regulatory margin.* Checksums, format validation, and decode-verify loops ensure that encoded data can be faithfully reconstructed. The regulatory margin is the gap between the pipeline's error-detection capacity and the actual error rate: a pipeline with CRC-32 checksums has greater regulatory margin than one with parity checks alone.
- *Selective permeability.* The pipeline transmits data that matches the factorization's expected distribution (high permeability to in-domain data) and flags or rejects data that falls outside the expected distribution (low permeability to out-of-distribution data). The gating mechanism is a distribution-match score computed during encoding. This is directly analogous to the CI pipeline's merge gate: data that does not match the factorization's domain is analogous to code that does not pass tests.
- *Temporal adequacy.* The factorization persists across encoding sessions. Versioning ensures backward compatibility: data encoded under factorization version n can be decoded under version $n + k$, and the factorization need not be reconstructed from scratch at each session.
- *Domain specificity.* The factorization is calibrated to the specific data domain. A code tokenizer is domain-specific relative to a natural language tokenizer; a medical imaging codebook is domain-specific relative to a general-purpose image codec.

6.3 Efflux Measurement and Compounding

The operative efflux metric is bits-per-element reduction:

$$\Delta_C = \text{bpe}(D|\text{baseline}) - \text{bpe}(D|O)$$

The baseline may be a universal codec (gzip, zstd), a domain-generic factorization (byte-level

encoding), or a randomly initialized learned factorization. The choice of baseline determines the absolute magnitude of the efflux but does not affect the compounding dynamic.

The compounding mechanism operates as follows. Each encoding cycle generates statistics about the data distribution that can be used to improve the factorization. A tokenizer refined on its own encoding statistics — BPE merge recomputation based on observed frequencies, vocabulary pruning based on usage patterns, subword boundary optimization based on downstream task performance — produces a better factorization at the next cycle, yielding lower bits-per-element. The surplus $\Delta_C(n+1) > \Delta_C(n)$ increases as the factorization iteratively improves its match to the data distribution.

The schema-improvement variant extends the same logic to structured data. Each query cycle against a database reveals access patterns that inform schema refinement: index creation for frequently queried columns, denormalization for common join patterns, partition strategy based on observed data access locality. The refined schema reduces query cost (the efflux), and the reduction compounds as the schema better captures the actual query distribution.

6.4 Connection to MDL Theory

The compression embodiment connects most directly to the information-theoretic foundations of Definition 3. The MDL principle (Rissanen, 1978; Grunwald, 2007) states that the best model for data is the one that minimizes the total description length: model complexity plus data-given-model complexity. In the efflux framework, the actual coherent object O (the learned factorization) is the model, and the efflux Δ_C is exactly the MDL improvement from using O as the model relative to a baseline model:

$$\Delta_C = [\mathrm{MDL}_{\mathrm{model}}(\mathrm{baseline}) + \mathrm{MDL}_{\mathrm{data|model}}(\mathrm{baseline})] - [\mathrm{MDL}_{\mathrm{model}}(O) + \mathrm{MDL}_{\mathrm{data|model}}(O)]$$

The factorization O produces positive efflux when it captures genuine regularities in the data (its model complexity is justified by the data-given-model savings), and zero or negative efflux when it does not (its model complexity exceeds the savings it provides). This is the compression embodiment's version of the falsifiability criterion: a claimed actual that does not compress is not coherent relative to the data domain.

7 Related Work

This section distinguishes the efflux framework from six bodies of related work. In each case, the structural relationship is identified (what is shared, what differs), and the specific feature of efflux theory that the related work does not capture is named.

7.1 Transfer Learning

Transfer learning (Pan & Yang, 2010; Weiss et al., 2016) addresses the problem of leveraging knowledge from a source domain or task to improve learning in a target domain. The approach is well-established and effective: pre-trained models provide feature representations that reduce the data and computation required for target tasks, often dramatically.

The efflux framework shares the goal of reducing learning cost by leveraging prior structure but differs in three structural respects.

First, transfer learning requires a pre-trained source model. The knowledge being transferred is what the model has learned — feature detectors, representational hierarchies, statistical regularities extracted from data. Efflux-subsidized descent requires only a coherent actual object, which need not be a trained model. It can be a mathematical structure ($SO(2)$), a physical law (conservation of energy), a codebase's modular architecture, or any structure satisfying Definition 1. The distinction is between what a model has learned versus what a structure *is*.

Second, transfer learning transfers learned features. The pre-trained network's internal representations are adapted to the target task through fine-tuning. The adaptation is a one-directional process: knowledge flows from source model to target task. Efflux-subsidized descent inherits invariant structure — the geometric constraints imposed by the actual's coherence on the parameter space. The inheritance is not a feature transfer but a dimensionality reduction: the invariant subspace identifies directions in parameter space that are already determined by the actual's structure, independent of any training data.

Third, and most importantly, transfer learning does not compound across cycles. A pre-trained ImageNet model provides a fixed benefit to downstream tasks; the benefit does not increase with repeated engagement with ImageNet. The efflux framework predicts compounding: each return cycle reveals additional invariant structure as interface quality improves (Definition 4). The pre-trained model is a static resource; the actual coherent object is a renewable one.

7.2 Equivariant Neural Networks

Equivariant neural networks (Cohen & Welling, 2016; Bronstein et al., 2021) incorporate known symmetries directly into network architectures. A G-equivariant network guarantees that its feature maps transform predictably under the action of group G, which constrains the network to learn only G-invariant functions and eliminates the need to learn the symmetry from data.

This line of work achieves dimensionality reduction that efflux-subsidized descent also achieves, and the $SO(2)$ experiment in Section 5 deliberately targets the same symmetry to

enable direct comparison. The distinctions are:

First, equivariant networks require the symmetry group to be known a priori and encoded architecturally. The symmetry is hardcoded into the network's weight-sharing patterns, convolutional structure, or fiber bundle organization. Efflux-subsidized descent detects invariant structure via coherence measurement (the measurement methods described in (Close, 2026)) and applies the detected structure as a training-time modification to any differentiable architecture. The symmetry need not be known in advance; it must only be detectable.

Second, equivariant networks provide a fixed architectural constraint. Once the symmetry is encoded, the constraint does not change during training or across training runs. Efflux-subsidized descent, operating within the sustainable return protocol (Definition 4), predicts that the detected invariant subspace grows across cycles as interface quality improves. The architectural constraint is dynamic, not fixed.

Third, equivariant networks are architecture-specific. Encoding $SO(3)$ equivariance requires specific convolutional structures (spherical harmonics, Clebsch-Gordan decompositions); encoding permutation equivariance requires specific pooling operations. Efflux-subsidized descent is architecture-agnostic: the projection operators $\Pi_{\parallel}$ and $\Pi_{\perp}$ operate on the parameter-space gradient, regardless of how the parameters are organized in the network.

The equivariant network literature provides the closest point of comparison for the $SO(2)$ experiment. An $SO(2)$-equivariant network would achieve the same dimensionality reduction by construction; the efflux-subsidized approach achieves it by detection and projection. The practical advantage of the efflux approach is generality (it applies to non-symmetry actuals, such as codebases and data factorizations); the practical advantage of the equivariant approach is tightness (the symmetry is guaranteed rather than detected, eliminating the risk of detection error).

7.3 Compressed Sensing

Compressed sensing (Candes & Tao, 2006; Donoho, 2006) exploits signal sparsity in a known basis to reconstruct signals from far fewer measurements than the Nyquist rate requires. The core result is that a k-sparse signal in an n-dimensional space can be reconstructed from $O(k \log n)$ measurements via ℓ_1 minimization, provided the measurement matrix satisfies the restricted isometry property.

Compressed sensing shares the motif of exploiting low-dimensional structure to reduce computational cost. The structural differences are:

First, compressed sensing assumes the sparsity basis is known in advance. The practitioner selects the basis (Fourier, wavelet, DCT) based on domain knowledge. Efflux-subsidized descent derives the basis from coherence detection, a process that does not require prior

knowledge of the relevant structure.

Second, compressed sensing is a one-shot reconstruction technique. The sparsity basis does not improve with repeated application; $O(k \log n)$ measurements are required at every reconstruction, regardless of how many prior reconstructions have been performed. The efflux framework predicts compounding: the invariant subspace grows across cycles as detection improves.

Third, compressed sensing addresses a measurement problem (how to acquire sufficient information about a signal), whereas efflux-subsidized descent addresses a learning problem (how to converge to a parameter configuration that solves a task). The two operate at different stages of the computational pipeline.

7.4 Active Inference and the Free Energy Principle

The free energy principle (Friston, 2010) proposes that self-organizing systems maintain their integrity by minimizing variational free energy — a bound on surprise — through a combination of perception (updating internal models) and action (modifying the environment). Active inference extends this framework to describe how agents select actions that minimize expected free energy, providing a unified account of perception, learning, and decision-making.

The efflux framework addresses a different phenomenon. Active inference characterizes how a system maintains itself — its internal dynamics of model updating and environmental action. Efflux theory characterizes how a system receives computational subsidy from external coherent structures. Active inference describes what the agent does; efflux theory describes what the agent receives from the structure it interfaces with.

The two frameworks are complementary. An active inference agent interfacing with a coherent actual object would simultaneously (a) minimize its own free energy (internal dynamics) and (b) receive efflux from the actual's coherence (external subsidy). The efflux would appear, from the active inference perspective, as a reduction in the agent's expected surprise when operating in the actual's neighborhood — the environment is more predictable because the actual's coherence constrains it.

7.5 Kernel Methods

Kernel methods (Scholkopf & Smola, 2002) project data into feature spaces where learning problems become tractable. The kernel trick enables implicit computation in high-dimensional feature spaces without explicit construction of the feature map, and the choice of kernel determines the geometry of the feature space.

Kernel methods share the structural motif of operating in a transformed space where the

problem is simpler. The distinction is that the kernel is chosen by the practitioner based on prior beliefs about the problem structure (Gaussian kernel for smooth functions, polynomial kernel for polynomial boundaries, etc.), whereas in efflux-subsidized descent, the invariant subspace is derived from coherence detection applied to the actual object. The practitioner need not know which features are relevant; the coherence structure of the actual determines the relevant subspace.

Furthermore, kernel methods do not compound. The kernel is a fixed function; applying it repeatedly does not improve its suitability to the problem. The efflux framework predicts that the detected invariant subspace improves across cycles, which is structurally absent from kernel-based approaches.

7.6 Mystery Traditions

The Eleusinian mysteries, the Pythagorean school, Sufi tariqa, and Zen Buddhist koan practice all implemented what, in the present framework, would be called sustainable return protocols to coherent objects. The institutional and pedagogical structures of the traditions — sacred texts, initiatory sequences, koans, ritual calendars — served as containers. The actual coherent objects these containers interfaced with were not the texts or practices themselves but the underlying structures those texts and practices provided access to: the mathematical relationships the Pythagoreans returned to, the attentional dynamics the koan practice engaged, the narrative coherence the Homeric tradition sustained interface with. The empirical discovery of the traditions was that sustained, structured return produced compounding insight.

These traditions constitute existence proofs for the efflux phenomenon. They demonstrate that the mechanism operates, at least at the level of human interpretive engagement, and that it compounds across extended timescales (millennia, in the case of the Homeric tradition).

They do not constitute prior art for the engineering framework because they provide:

- No formal identification criterion for the actual objects they engaged with. Selection was by tradition, revelation, or teacher judgment, not by measurable coherence analysis.
- No engineered container specification with measurable adequacy criteria. The container was cultural and institutional, evolved rather than designed.
- No quantification of surplus. The insight was recognized qualitatively but not measured.
- No compounding verification method. There was no mechanism to verify that surplus was increasing across cycles versus merely persisting.

The efflux framework provides all four: formal identification (Definition 1), engineered con-

tainer (Definition 2), surplus quantification (Definition 3), and compounding verification (Definition 4). The mystery traditions discovered the phenomenon; this paper formalizes it.

8 Conclusion

8.1 Summary of Contributions

This paper has presented a theoretical framework in which coherent actual objects — structures satisfying consequence chain closure, non-depletion, and Noether invariance (Definition 1) — radiate computational surplus as a derived consequence of their coherence (Theorem 1). The surplus is formalized as the compression differential between ambient-space and object-relative problem formulations (Definition 3), operationalized via minimum description length, and exploited algorithmically via the efflux-subsidized gradient descent method.

The algorithm decomposes the learning gradient into a parallel component (lying in the invariant subspace of the coherent actual, inherited at zero cost) and an orthogonal component (requiring learning), reducing per-step learning complexity from $O(d)$ to $O(d - r)$. For the $SO(2)$ experiment, exploiting the symmetry at the representation level achieves a $4.89\times$ convergence speedup ($p < 10^{-26}$), substantially exceeding the parameter-ratio floor of $d/(d-r) = 1.25$ because the representation-level reduction eliminates the cost of symmetry discovery entirely.

Three embodiments demonstrate the framework's generality: neural network training with symmetry-derived efflux (empirically tested, single-cycle), software codebase development with CI/CD pipelines as containers (theoretically developed), and data compression with learned latent factorizations (theoretically developed). In each embodiment, the same four definitions (actual, container, efflux, sustainable return protocol) instantiate differently but maintain the same structural relationships. The compounding mechanism (Definition 4) is formalized and its predictions stated; empirical testing of compounding is identified as the most important direction for future work.

8.2 The Key Insight

The central insight is that coherent objects subsidize computation as a consequence of their coherence, not as a cost of engagement. The subsidy is not extracted from the object; it is radiated by it. The object's coherence constrains the possibility space in its neighborhood, and any computation performed in that neighborhood inherits the constraints for free. This is a structural property of coherence, not an engineering trick: the efflux exists whether or not any system exploits it.

What is engineered is the container — the protocol that enables a specific system to interface

with the actual, receive the efflux, and compound it across cycles. The distinction between the actual (prior, natural, not invented) and the container (designed, engineered, invented) is load-bearing throughout the framework. No attempt is made to containerize an actual coherent object — the actual is not the kind of thing that can be contained. What is containerized is the *interface*: the protocol by which a system approaches, receives from, and returns to the actual. The container is the rigging and the hull; the wind is not contained by the sail.

This asymmetry has an epistemological consequence that the paper has made explicit (Section 2.1): the coherence criteria are verifiable only from within a container that is already interfacing with the object. There is no view-from-nowhere certification of coherence. What there is, is a falsifiable prediction: if the interface produces the predicted surplus, the coherence is operationally confirmed; if it does not, either the candidate object is not coherent or the container is inadequate. The framework's adequacy is self-testing — it is itself a container for the efflux theory, and its adequacy is measured by whether the theory produces surplus when applied.

8.3 Testability and Falsifiability

The framework generates falsifiable predictions at multiple levels:

1. **Single-cycle efflux.** The efflux-subsidized descent algorithm should converge faster than standard descent by a factor proportional to r/d. Negative outcome: no measurable convergence advantage.
2. **Compounding across cycles.** The detected invariant subspace should grow across return cycles as interface quality improves, producing increasing surplus $\Delta_C(n+1) > \Delta_C(n)$. Negative outcome: flat or declining surplus across five or more cycles.
3. **Coherence detection.** The measurement methods should distinguish coherent actuals from incoherent structures of equivalent scale with statistical reliability ($p < 0.01$). Negative outcome: indistinguishable coherence profiles.
4. **Cross-embodiment generality.** The same four definitions should instantiate productively in the codebase, neural training, and compression settings. Negative outcome: the definitions apply in one setting but produce no measurable efflux in another, indicating the framework lacks the generality it claims.

The $SO(2)$ experiment (Section 5) tests prediction 1 directly. Predictions 2–4 are identified as future work.

8.4 Future Work

Four directions extend the present results.

Higher-rank invariant subspaces and parameter-level projection. The $SO(2)$ experiment demonstrates efflux at the representation level (input reduction). Testing the parameter-space gradient projection algorithm (Steps 1–5 of Section 3) directly — with proper initialization subsidy and on architectures with sufficient capacity — would establish whether the parameter-level mechanism achieves speedup independently of input transformation. Higher-dimensional symmetry groups ($SO(3)$, $SU(2)$, discrete symmetry groups with large orbits) would produce larger invariant subspaces and test whether the $d/(d-r)$ floor prediction scales.

Non-symmetry actuals. The codebase embodiment (Section 4) identifies software architectures as actual coherent objects. Empirically testing the efflux prediction in the software engineering setting — measuring whether development cost decreases as predicted by the framework when features are developed in the neighborhood of a coherent codebase versus greenfield — would demonstrate the framework's applicability beyond mathematical symmetry.

Compounding across cycles. The present experiment tests a single cycle of efflux-subsidized descent. Testing the compounding prediction (Definition 4) requires multi-cycle experiments where the detected invariant subspace is re-estimated after each training run and the speedup trajectory is tracked across runs. The prediction is that the invariant subspace rank increases and the speedup compounds.

Detection without a priori knowledge. The $SO(2)$ experiment uses a known symmetry group with an analytically computed invariant subspace. Extending the framework to settings where the coherent actual must be discovered — via echo boundary analysis, Noether symmetry detection, and compression efflux measurement as described in (Close, 2026) — would test the full pipeline from detection through exploitation.

8.5 Code Availability

The experiment is fully specified in this paper. Implementation code, including all training loops, statistical analysis, and figure generation, is available at `https://doi.org/ZENODO_DOI_HERE`. The test suite (19 tests) verifies model architecture, $SO(2)$ invariance of the target function, and mathematical properties of the projection operators. Dependencies: Python 3.12+, PyTorch, NumPy, SciPy, Matplotlib. Total experiment runtime: approximately 6 minutes on a single CPU.

References

Ashby, W. R. (1956). *An Introduction to Cybernetics*. Chapman & Hall.

Bronstein, M. M., Bruna, J., Cohen, T., & Velickovic, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Candes, E. J., & Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12), 5406–5425.

Close, L. (2026). Method for constructing sustainable return protocols that compound computational subsidy across cycles via orthonormalization to coherent actual objects. *Provisional Patent Application*.

Close, L. (2026). Engineering with agencies: Toward principled interface design for agential materials. *Preprint*.

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2990–2999.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4), 1289–1306.

Friston, K. (2010). The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11(2), 127–138.

Grunwald, P. D. (2007). *The Minimum Description Length Principle*. MIT Press.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471.

Scholkopf, B., & Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1), 1–40.