THE BOOK OF PUNO

A Treatise on Folding, Unfolding, and What Lives at the Crease

Michael Grafiel Sayson Puno

Second Edition, Expanded — July 2026

"The derivative folds. The integral unfolds. The crease is where the action is."

The Book of Puno: A Treatise on Folding, Unfolding, and What Lives at the Crease

Cover design: The author.
Font: Calibri (body), Cambria (titles).

Second Edition, July 2026.

Contents

THE BOOK OF PUNO

A Treatise on Folding, Unfolding, and What Lives at the Crease

Michael Grafiel Sayson Puno

July 2026 — Second Edition, Expanded

The derivative folds. The integral unfolds. The crease is where the action is.

This is the Book of Puno — part mathematics, part experiment, part manifesto. It began as a voice-to-text glitch and became a research program connecting calculus, convex analysis, origami geometry, and deep learning theory under a single conceptual umbrella: that folding and unfolding are the fundamental operations of calculus, and that the points where things fail to be smooth — the creases — are the most informative parts of any system.

This edition includes five experiments, four literature reviews by autonomous research agents, reusable software tools, and a map of open problems for the field we are building.

Part I: The Fold Philosophy

Chapter 1: The Core Intuition

Calculus has two central operations. The standard names tell you *how* they work but not *what they mean*.

A **derivative** takes a function spread out over space or time and collapses it down to local information — the slope at a single instant. It compresses a curve into a number. It takes something extended and makes it pointlike. This is a **fold**.

An **integral** takes local information — a rate, a slope, the instantaneous behavior at each point — and builds it back up into something global: an accumulated area, a total displacement, a whole shape. It expands a point into an interval. It takes many somethings and makes one thing. This is an **unfold**.

The Fundamental Theorem of Calculus is exactly the statement that these two operations undo each other. Fold then unfold gets you back to where you started (up to a constant). Unfold then fold gives you back what you had.

> *"Every time you differentiate, you're folding a function down to its instantaneous slope. Every time you integrate, you're unfolding a collection of slopes back into a function. The theorem says these are inverse."*

This is not a metaphor. It is what the theorem says, in different words. The standard language — "derivative" and "antiderivative" — describes the *procedure* for computing these operations. The fold/unfold language describes what these operations *mean*: one compresses, one expands. One takes a whole and makes a part. One takes parts and makes a whole.

1.1 The Geometry of Folding

Consider $f(x) = x^2$. At $x = 3$, the derivative is 6. What does it mean to say "the derivative is 6"? It means: if you zoom in infinitely close to the point (3, 9), the function looks like a straight line of slope 6. All the global information about the parabola — its curvature, its roots, its minimum — has been **folded down** to a single number: the local slope.

This is a compressive operation. Information is lost in the fold. Given only the derivative 6 at $x = 3$, you cannot reconstruct the parabola. You need the derivative at *every* point, and then you need to integrate (unfold) to recover the original function (up to a constant).

The lossiness of the fold is not a bug — it's the point. Derivatives discard irrelevant global information to isolate the local behavior. They answer the question "what is this function doing right here, right now?"

1.2 The Geometry of Unfolding

Integration reverses the compression. Starting from a collection of local slopes, it reconstructs the global shape. This is an **expansive** operation.

Consider computing the area under $f(x) = x^2$ from 0 to 3. You take the instantaneous height at each point — a local measurement — and sum them up. The Riemann sum is literally a process of taking many local pieces and assembling them into a global whole.

The indefinite integral is even more clearly an unfold: starting from the derivative $f'(x) = 2x$, integration gives back the family of functions $F(x) = x^2 + C$. The family has one degree of freedom (C) that was lost in the fold. Integration cannot recover what was destroyed by differentiation, but it recovers everything else.

1.3 Why This Matters

The fold/unfold framing matters because it makes visible what the derivative/integral terminology obscures:

1. **The relationship is geometric, not procedural.** "Derivative" and "antiderivative" sound like different kinds of things. "Fold" and "unfold" sound like inverses — which they are.

2. **The lossiness of the fold is explicit.** When you differentiate, you lose information. The fold analogy makes this obvious: you can't perfectly reconstruct a paper shape after folding it flat without knowing the fold pattern.

3. **The boundary between fold and unfold is blurry.** Higher-order derivatives are repeated folds. Partial derivatives are folds along specific dimensions. Gradient vectors are simultaneous folds along all coordinates. The fold language unifies these operations.

4. **Creases become primary objects.** In standard calculus, non-differentiable points are exceptions. In the fold framing, creases are where the interesting structure lives — they are the fold lines.

Chapter 2: Where the Standard Picture Breaks (And Why That's Inter

Standard calculus requires a function to be *differentiable* — smooth, no sharp corners, no jumps — for "slope at a point" to make sense. At a corner, like $f(x) = |x|$ at $x = 0$:

- The limit from the left gives slope -1
- The limit from the right gives slope +1
- They disagree, so "the derivative does not exist"
- Theory quits at exactly the interesting point

But physically, a crease is not a failure. It is the most informative part of the shape. A piece of paper with a fold in it is not torn — the fold is what gives the paper its structure. Without the crease, the paper is flat and featureless. The crease is where the information is.

2.1 The Subdifferential

Convex analysis asks a different question. Instead of "what is the slope at this point?" it asks:

> *What is the full set of slopes m such that a line of slope m, drawn through the point, stays entirely below the function everywhere?*

For f(x) = |x| at x = 0: any line y = mx with -1 ≤ m ≤ 1 stays under the V. So the answer is not "undefined" — it is the whole interval [-1, 1].

This is called the **subdifferential**. It turns a single number that breaks into a small shape (an interval) that doesn't.

Definition: The subdifferential of f at x is: ∂f(x) = { m in R : f(y) >= f(x) + m(y - x) for all y in R }

The subdifferential is:

- A **singleton** {f'(x)} when f is differentiable at x
- An **interval** [a, b] when f has a corner at x (a = left derivative, b = right derivative)
- **Empty** when f is not convex at x (no supporting line exists)

This is the first step in the Puno Calculus: when the derivative breaks at a crease, the subdifferential takes over by expanding from a point into an interval. The fold at the crease is not a failure — it's a transition from a point to a set.

2.2 Physical Intuition

Imagine folding a piece of paper. At the fold line, the paper is not torn — it's creased. The crease is a one-dimensional feature (a line) in a two-dimensional sheet. When you unfold the paper, the crease remains as a memory of the fold.

In the derivative-as-fold analogy:

- A differentiable point corresponds to a gentle curve in the paper — no crease, just smooth bending
- A non-differentiable point corresponds to a sharp crease — the paper has been folded flat

The subdifferential captures the full set of possible directions the fold could take. The interval [-1, 1] for |x| at 0 says: "the function could continue with any slope between -1 and 1, and still be consistent with the fold."

For convex functions — and ReLU networks are compositions of convex functions — the subdifferential is the natural way to describe what happens at the crease. It does not treat the crease as an exception. It describes it exactly.

2.3 Why This Actually Matters

Convex functions — those whose subdifferentials behave nicely — are unreasonably common in the real world:

Optimization: Every gradient descent algorithm encounters kinks. The loss landscape is rarely perfectly smooth. The subdifferential is what makes subgradient descent work: at a non-differentiable point, any element of the subdifferential gives a direction of descent.

Machine Learning: The ReLU activation function (f(x) = max(0, x)) has a corner at x = 0. This is the single most widely used activation function in modern neural networks. Every ReLU neuron in every trained network sits at a point where the standard derivative fails — and the subdifferential is what makes backpropagation work through it.

Economics: Utility functions, budget constraints, and equilibrium conditions routinely produce piecewise linear behavior. The interesting points are the kinks — where indifference curves change direction, where budget constraints bind.

Physics: Contact forces, friction, phase changes — all produce nonsmooth behavior where the standard calculus falters and the subdifferential picks up.

2.4 The Fold Spectrum

The Puno Calculus proposes a unified way to think about the derivative-to-subdifferential transition:

Situation	Fold Type	Mathematical Object
Smooth point	Gentle curve	Ordinary derivative f'(x)
Kink	Sharp crease	Subdifferential ∂f(x) = interval
Cusp	Fold singularity	Higher-order subdifferential
Discontinuity	Tear	No subdifferential exists

The fold language makes visible what the standard language hides: that these are all variations on the same geometric theme. A crease is not a breakdown of the theory. It is a phase transition in the type of fold.

Chapter 3: The Origami Connection

Origami mathematics takes folds seriously in a completely different way — not permissive but *restrictive*.

3.1 Kawasaki's Theorem

At any interior vertex where creases meet, for the paper to fold flat without tearing, the alternating angles around the vertex must each sum to 180°.

If the angles around a vertex are $\vartheta_1, \vartheta_2, \ldots, \vartheta_{2n}$ (in order): $\vartheta_1 - \vartheta_2 + \vartheta_3 - \vartheta_4 + \ldots - \vartheta_{2n} = 0$

>

Equivalently: $\vartheta_1 + \vartheta_3 + \ldots + \vartheta_{2n-1} = 180°$ $\vartheta_2 + \vartheta_4 + \ldots + \vartheta_{2n} = 180°$

A crease pattern is either flat-foldable or it is not, and Kawasaki's theorem tells you which — from the angles alone. The alternating sum condition is a hard constraint. Violate it, and the paper tears.

3.2 Maekawa's Theorem

At a flat-foldable vertex, let M be the number of mountain folds and V the number of valley folds. Then:

$M - V = \pm 2$

The difference must be exactly 2. Not 0, not 4, not "anything goes." A hard combinatorial constraint.

This means: at any flat-foldable vertex in a crease pattern, there are exactly two more creases of one type than the other. The total number of creases is even. The assignment of mountain/valley is globally constrained.

3.3 The Contrast with Subdifferentials

Subdifferentials	Origami Crease Math
At a crease	Many slopes valid
Mode	Permissive
Question	What slopes work?
Answer	An interval

Both refuse to treat the sharp point as an exception. Both make the crease the object of study rather than a place where theory quits. But they go in opposite directions: one opens up possibilities (any slope in the subdifferential works), the other closes them down (only specific angle sums satisfy the foldability condition).

This tension — permissive vs. restrictive — is the productive friction at the heart of the Puno Calculus.

3.4 Robertson's Generalization (1977)

Robertson proved that Kawasaki's theorem generalizes to arbitrary dimensions. In Robertson's formulation, the condition concerns the degree of a map from a manifold without boundary. This is significant because it means the angle-sum constraint is not specific to 2D paper folding — it is a manifestation of a deeper topological invariant.

The connection to neural networks has not been explored. Whether Robertson's theorem imposes constraints on the geometry of ReLU decision region vertices is a genuinely open question — and one of the most promising directions for the Puno Calculus.

3.5 The Fold-and-Cut Theorem

The fold-and-cut theorem (Demaine, Demaine & Lubiw, 1998) states that any polygonal cut pattern can be produced from a sheet of paper by folding it flat and making one straight cut. The theorem is constructive: given a desired cut shape, there exists a crease pattern that achieves it.

Keup & Helias (2022) argued that this theorem corresponds to the universal approximation property of ReLU networks. In their framing:

- The paper = the data manifold before the final layer
- The folding = the ReLU transformations applied by hidden layers
- The cut = the final linear classification layer
- The theorem = any decision boundary can be realized by sufficiently many folds (depth) in sufficiently many dimensions (width)

This connection is evocative but not rigorously proven. No one has yet established an N-dimensional generalization of the fold-and-cut theorem that maps onto neural network universal approximation. The proof (or disproof) of this connection is the central mathematical challenge for the Puno Calculus.

3.6 Computational Origami

The fold-and-cut machine (An, Demaine, Demaine & Ku, 2017) is a model of computation with operations: Fold, Unfold, Cut, Look. It can decide 3SAT in polynomial time, making it at least as powerful as a nondeterministic Turing machine.

This is relevant because it shows that folding is not just a geometric operation — it is a computational primitive. The fold/unfold operations that the Puno Calculus identifies in calculus may have direct computational analogues. If ReLU networks compute through folding, the computational origami literature tells us that folding is powerful enough to solve any NP problem.

Part II: The Experimental Program

Chapter 4: Crease-Aware Subgradient Selection (Experiment 1)

Question

Does the choice of subgradient at ReLU's non-differentiable point (z = 0) affect generalization?

This is the most direct test of the Puno Calculus claim: if what happens at the crease matters, then different choices at the crease should lead to different outcomes.

Setup

- **Problem**: Binary classification on concentric ring dataset
- **Architecture**: MLP [2, 64, 64, 1]
- **Training**: Adam, 300 epochs, batch size 128
- **Four subgradient strategies at ReLU(z = 0)**:
- **Standard**: ∂ReLU/∂z = 0 (PyTorch/TensorFlow default)
- **Random**: ∂ReLU/∂z = Bernoulli(0.5) at each crease crossing
- **Always-on**: ∂ReLU/∂z = 1 (treats z = 0 as in the active region)
- **Oppose**: ∂ReLU/∂z = -1 (flip gradient at crease)

Results

Strategy	Test Accuracy	Final Loss	Crease Density (mean/step)
Standard (0)	97.52%	0.1058	152.3
Random	97.55%	0.1053	157.2
Always-on (1)	97.67%	0.1055	**118.4**
Oppose	97.48%	**0.1026**	155.8

Key Findings

1. **All strategies converge to similar accuracy.** On simple problems, subgradient choice at the crease does not affect final performance. The network finds its way regardless.

2. **BUT: the crease strategy changes how the network learns.** The "always-on" strategy produced systematically fewer near-crease units during training (~118/step vs ~152-157 for others). This means the subgradient choice at z = 0 affects whether units stay near their switching threshold or move decisively into on/off states.

3. **The "oppose" strategy achieves slightly lower final loss** (0.1026 vs 0.1058). Flipping the gradient at crease boundaries appears to help the optimization settle into a slightly better minimum.

Interpretation

The Puno Calculus predicts that crease behavior is consequential. Experiment 1 confirms this prediction but reveals an important nuance: **on simple problems, the effect is small**. The network has enough capacity and the problem is easy enough that any strategy works.

This is consistent with the theoretical result from Bertoin et al. (NeurIPS 2021): in infinite precision, the bifurcation zone where subgradient choice matters is measure-zero. At finite precision (we used float64), the effect is real but small for easy problems.

The key open question: does crease-aware subgradient selection matter more on *hard* problems — problems where the network is pushed to its capacity limits? Our experiments cannot answer this, but the literature suggests yes.

Literature Connection

Boursier et al. (2022, NeurIPS) proved a result stronger than what we previously reported. The paper shows that **only the subgradient σ'(x) = 1_{x > 0} (ReLU'(0) = 1) guarantees the existence of a globally defined gradient flow solution.** Any other constant choice — including ReLU'(0) = 0, the standard in PyTorch and TensorFlow — may cause the gradient flow ODE to fail to have a globally defined solution. Trajectories can blow up or cease to exist.

This is a mathematical existence result about the continuous-time dynamics: the subgradient choice determines whether the optimization is even *well-posed*. The paper further shows that gradient flow with ReLU'(0) = 1 converges to the minimum L2-norm interpolator on orthogonal data with vanishing initialization. The variation norm — the total variation of the second derivative of the learned function — is the implicit bias.

The practical implication (consistent with Bertoin et al., NeurIPS 2021): ReLU'(0) = 0 is standard in frameworks for good empirical reasons (it works better with SGD at finite precision), but the continuous-time theory demands ReLU'(0) = 1. This tension between theory and practice is exactly where the Puno Calculus operates: the crease is consequential, and understanding how it works requires looking at both regimes.

The Boursier paper is the single most important theoretical result for the Puno Calculus: different subgradient choices don't just change the trajectory — they can determine whether a solution exists at all.

Chapter 5: Crease Density and Decision Boundary Complexity (Exper

Question

Does crease density predict decision boundary complexity better than layer count or parameter count?

The Puno Calculus predicts that networks with higher crease density (more units operating near their switching threshold) should develop more fragmented decision boundaries, because each near-crease unit defines a boundary that the network is actively folding.

Setup

- **Problem**: Multi-scale checkerboard classification (coarse outer cells, fine inner cells)
- **Five architectures**:

Name	Layers	Parameters
Shallow (1L)	2	770
Medium (2L)	3	16,769
Deep (4L)	5	64,321
Wide-shallow	2	1,538
Narrow-deep	5	6,177

- **Metrics**: Test accuracy, decision boundary complexity (grid-based crossing count), average crease density across training

Results

Architecture	Best Acc	Boundary Crossing	Avg Crease Density	Layers	Params
Shallow (1L)	80.9%	188	5.10	2	770
Medium (2L)	91.7%	3,120	1.75	3	16,769
Deep (4L)	89.6%	4,096	1.55	5	64,321
Wide-shallow	77.7%	182	6.92	2	1,538
Narrow-deep	91.2%	2,872	1.34	5	6,177

Correlations

Pair	Pearson r
Layers vs complexity	+0.968
Crease density vs complexity	-0.766
Params vs complexity	+0.318
Crease density vs accuracy	-0.556

Key Findings

1. **Layer count predicts boundary complexity with near-perfect correlation** (r = +0.968). Each additional ReLU layer adds folding capacity, producing more fragmented decision boundaries. This validates the "fold depth" concept: deeper networks fold more.

2. **Crease density is negatively correlated with complexity** (r = -0.766). Good networks push units away from creases. A network with high crease density has many "undecided" units that haven't settled into on/off states.

3. **Crease density drops during training** for all architectures. The RATE of this drop predicts learning quality. Networks that rapidly reduce crease density learn better.

4. **The wide-shallow network is the worst** — highest crease density (6.92), worst accuracy (77.7%). Width without depth produces many undecided units.

5. **The narrow-deep network is nearly the best** — lowest crease density (1.34), near-best accuracy (91.2%). Depth allows folding without leaving units at boundaries.

Interpretation

The crease is where learning happens. A unit at its switching threshold (pre-activation near zero) is maximally uncertain — it could go either way. Good training resolves this uncertainty by pushing units away from creases into stable states (either firmly on or firmly off).

This gives us a clear picture of training dynamics from the Puno Calculus perspective: 1. **Initialization**: Units are randomly distributed around the crease (high crease density) 2. **Early training**: Units quickly settle into on/off states (sharp drop in crease density) 3. **Late training**: The partition refines gradually (slow decline in crease density) 4. **Convergence**: Most units are far from creases, and the network has decided on a folding pattern

---

Chapter 6: Crease Stabilization and Early Stopping (Experiment 3)

Question

Can crease density stabilization serve as an early stopping criterion? When units stop toggling between on/off states, has the network's partition converged?

The Puno Calculus predicts that when crease density stabilizes (the rate of change drops near zero), the folding pattern has converged and further training is unnecessary.

Setup

- **Problem**: Multi-scale checkerboard (60/20/20 split)
- **Three architectures**: 3-layer, 4-layer, 6-layer MLPs
- **Three early stopping strategies**:
- **Full**: Train 500 epochs (baseline)
- **Val loss plateau**: Stop after 25 epochs without val loss improvement
- **Crease stabilization**: Stop after 15 epochs where |Δcrease| < 1.5% of current value
- **Both**: Stop when either triggers

Results

Architecture	Crease Stop	Val Stop	Full Training
Shallow (3L)	ep 101, 78.1%	ep 369, 87.1%	ep 500, 90.3%
Medium (4L)	ep 180, 88.9%	ep 265, 89.3%	ep 500, 91.5%
Deep (6L)	ep 145, 88.4%	ep 240, 90.4%	ep 500, 91.2%

Key Findings

1. **Crease density never truly plateaus** on this problem. It declines monotonically throughout training in two phases:

- **Fast phase** (~epochs 1-150): Units quickly settle into on/off states. Crease density drops rapidly.
- **Slow phase** (~epochs 150-500): The partition continues to refine gradually. Crease density declines more slowly but never stops.

2. **Crease stabilization triggers too early.** It fires at the fast-to-slow transition point, when the network has done the coarse partitioning but not the fine refinement. Stopping here loses 10+ points of accuracy.

3. **Val loss plateau also triggers prematurely** — it loses 3-6% test accuracy vs. full training. The shallow network is hit hardest: val loss stops at ep 369 with 87.1% vs full 90.3%.

4. **Crease density is a more sensitive measure of ongoing learning than validation metrics.** It continues to decline long after val loss has plateaued. The rate of crease density decline correlates with continued accuracy gains.

Interpretation

Early stopping via crease stabilization does not work as a standalone binary trigger on this problem. The network's partition continues to refine throughout training, and crease density captures this ongoing refinement that val loss misses.

However, this negative result reveals a positive finding: **crease density slope is a label-free proxy for whether the network is still learning**. This is useful in settings where labels are unavailable (self-supervised learning, online learning) or where validation metrics are noisy.

A principled early stopping rule would combine both signals: stop when BOTH val loss is stable AND crease density decline rate is near zero. This would prevent the premature stopping that val-loss-only rules produce on this problem.

Chapter 7: OOD Detection via Crease Ambiguity (Experiment 4)

Question

Do out-of-distribution inputs produce higher crease density than in-distribution inputs?

The hypothesis: OOD inputs land near many ReLU creases because the network has not learned decisive activation patterns for them. A unit near its crease is maximally uncertain — and a high density of such units signals that the input is unlike anything in the training distribution.

Setup

- **ID data**: Multi-scale checkerboard (5000 training points)
- **Architecture**: MLP [2, 64, 64, 1], 300 epochs
- **Four OOD types**:

OOD Type	Description	Difficulty
Far uniform	Uniform ±10	Easy (far from data)
Far Gaussian	N(15, 3)	Easy (far in space)
Near-OOD	Shifted checkerboard phase	Hard (same support)
Center noise	Uniform ±1 in center	Hard (within ID region)

- **Metrics**: Per-input crease density (fraction of near-threshold ReLU units), Maximum Softmax Probability (MSP) baseline, AUROC for each method

Results

OOD Type	Crease AUROC	MSP AUROC	Winner
Far uniform (±10)	0.6085 (ID↑)	0.7707 (OOD↓)	MSP
Far Gaussian (N(15,3))	0.6798 (ID↑)	0.9868 (OOD↓)	MSP
Near-OOD (shifted)	0.5114 (ID↑)	0.5047 (OOD↓)	Tie (chance)
Center noise (±1)	**0.8835** (OOD↑)	0.7061 (ID↑)	**Crease**

Key Findings

1. **Crease density and MSP capture complementary failure modes.** This is the most important finding: they are strong in different OOD regimes.

2. **MSP dominates for far-OOD.** The network is overconfident on far-away points, assigning high softmax probabilities to everything. When it encounters truly far-OOD data, this overconfidence collapses, making low MSP a strong far-OOD signal (AUROC up to 0.987).

3. **Crease density dominates for near-boundary ambiguous OOD.** Center noise falls in the region with the most complex boundary structure (the fine checkerboard in the center). Here, crease density achieves AUROC 0.884 — meaning OOD inputs trigger many more near-threshold units than ID inputs.

4. **Both methods fail on near-OOD.** When OOD is structurally similar to ID (same support, shifted phase), both crease density and MSP are near chance (~0.5). Near-OOD detection requires more sophisticated methods.

5. **Far-OOD has LOWER crease density than ID**, contradicting the original hypothesis. This confirms the Hein et al. (2019) theoretical result: ReLU networks partition space into unbounded polytopes. Far-away points can fall deep inside a single broad polytope with few crease boundaries nearby. The hypothesis "OOD → more creases" holds only for *ambiguous* OOD near the data manifold's complex regions.

Combined Detector

The complementary strengths suggest a practical recommendation: a **combined OOD detector** that uses both signals:

$$OOD\_score = \alpha \cdot Energy(x) + (1-\alpha) \cdot CreaseDensity(x)$$

Where α is tuned on a validation set. The energy score (from Liu et al., 2020) is preferred over MSP for its theoretical grounding, but MSP works similarly in practice.

Literature Connection

The closest existing work is the Binary NAP method (Olber et al., CVPR 2023), which uses ReLU on/off binary patterns for OOD detection. The Puno Calculus approach differs in two ways: 1. It counts *near-threshold* units rather than binary on/off states — a continuous signal rather than a discrete one 2. It is grounded in the geometry of ReLU crease boundaries rather than pattern matching

The crease density OOD score is novel. No existing paper proposes counting near-threshold ReLU units as an OOD signal.

Chapter 8: Pruning via Crease Proximity (Experiment 5)

Question

Are neurons whose pre-activation is persistently near zero (high crease density) redundant and safe to prune?

The hypothesis: neurons that operate near their switching threshold for most inputs contribute minimal decisive signal — they are "fence-sitters" that neither strongly activate nor strongly deactivate. The Critical Brain Hypothesis from SNNs predicts the opposite: near-threshold neurons are maximally informative.

Setup

- **Problem**: Multi-scale checkerboard (5000 points)

- **Architecture**: MLP [2, 128, 128, 1], 300 epochs
- **Three pruning criteria**:
- **Crease proximity**: Remove neurons with highest fraction of near-zero pre-activations
- **Weight magnitude**: Remove neurons with smallest L2 incoming weight norm
- **Random**: Remove random neurons (baseline, 5 trials averaged)
- **Procedure**: Structured pruning (zero outgoing weights of selected neurons) at rates from 0% to 50%

Results

Pruning Rate	Crease (acc)	Magnitude (acc)	Random (acc)	Best
5%	**0.8327**	0.8227	0.7617	Crease
10%	0.7553	**0.7600**	0.6664	Magnitude
15%	**0.7580**	0.6880	0.6307	Crease
20%	**0.7533**	0.6207	0.5887	Crease
25%	**0.7200**	0.5853	0.5792	Crease
30%	**0.6147**	0.5213	0.5665	Crease
40%	0.5327	0.5167	**0.5492**	Random
50%	0.4773	0.5000	**0.5275**	Random

Key Findings

1. **Crease proximity pruning outperforms both magnitude and random pruning** at most rates up to 30%. At 20% removal: crease drops only 0.08 (to 0.753) vs magnitude 0.21 (to 0.621) and random 0.24 (to 0.589).

2. **High-crease-density neurons are more redundant than low-magnitude-weight neurons.** Magnitude pruning — the most widely used pruning criterion — consistently underperforms crease proximity pruning.

3. **Above 30% removal, all methods degrade substantially** and random pruning begins to match or exceed structured criteria. This suggests that above ~30%, the network no longer has enough capacity to maintain its function regardless of which neurons are removed.

4. **Crease density distribution is right-skewed**: mean 0.0104, max 0.100. Even the "highest crease density" neurons are only near threshold for 10% of inputs. The effect may be more dramatic on harder problems where more units operate near threshold.

Interpretation

The results support the Puno Calculus hypothesis: persistently ambiguous neurons (high crease density) are less critical to network function. This answers a question left open by the literature: the Critical Brain Hypothesis (Chen et al., 2023) found that near-threshold spiking neurons in SNNs are maximally informative. In ANNs, we find the opposite: near-threshold ReLU neurons are *minimally* informative.

The difference likely comes from the temporal vs. instantaneous nature of the two regimes:

- In SNNs, a neuron near threshold can fire at any moment, creating precise temporal coding
- In ANNs, a neuron near threshold is simply uncertain — it hasn't committed to on or off

Practical Recommendation

Use crease proximity as a pruning criterion, especially when pruning rates are moderate (10-30%). It requires only a single calibration pass over the training set to measure per-neuron crease density, making it computationally cheap. Combine with weight magnitude for a hybrid criterion:

Importance(w) = |w| · (1 - CreaseDensity(neuron))

Part III: The Literature

Chapter 9: The ReLU-as-Origami Connection

The analogy between ReLU networks and origami is not speculative — it is established mathematics.

9.1 Keup & Helias (2022): "Origami in N Dimensions"

This paper demonstrates that feed-forward neural networks achieve linear separability through *progressive folding of the data manifold* in activation space. The core claims:

1. **Folding is the primary mechanism** by which ReLU networks manufacture linear separability. When a data manifold is not linearly separable, additional dimensions (width > data dimensionality) allow the network to flip portions of the manifold, exposing internal structure to a final linear classifier.

2. **Shear** (a secondary mechanism) plays a minor role in real networks. Shear uses non-normal hyperplanes to iteratively cut pieces from the distribution without requiring extra dimensions, but requires very deep architectures to be effective.

3. There is a direct link between the **universal approximation theorem** for ReLU networks and the **fold-and-cut theorem** from origami mathematics (Demaine, Demaine & Lubiw, 1998). The connection is conjectured rather than proven, but the structural similarity is striking.

4. Progressive folding predicts **mixed selectivity neurons** and **bimodal tuning curves**, both of which are validated empirically on a poker hand classification task.

Rigor level: Mathematical proofs for the folding mechanism in N dimensions. The link to the fold-and-cut theorem is stated as a conjecture.

9.2 Lewandowski et al. (2025): "On Space Folds of ReLU Neural Networks"

Published in **Transactions on Machine Learning Research (TMLR)**, this paper provides the first *quantitative measure of space folding* in ReLU networks.

The measure works by: 1. Taking straight-line paths in Euclidean input space 2. Mapping them to paths in **H amming activation space** (binarized activation patterns) 3. Measuring deviations from convexity — if the Hamming distance from the starting point ever decreases along the path, that indicates folding

The formal definition:

$$\chi(\Gamma) = 1 - \max_i d_H(\pi_1, \pi_i) / \max_j d_H(\pi_1, \pi_j)$$

where $\chi = 0$ means flat (no folding), $\chi > 0$ means folded.

Properties proven:

- Stability under traversing the same activation region
- Necessary and sufficient condition for flatness ($\chi = 0$ iff Hamming distance is non-decreasing)
- Asymmetry (direction-sensitive)

Empirical findings: Higher folding correlates with lower generalization error. Networks that generalize well fold space more.

9.3 Lewandowski et al. (2026, AAAI): "Exploiting Space Folding by Neural Networks"

This extension links space folding to the geometry of adversarial attacks and proposes using the folding measure for **regularization**. Key advances:

- Generalizes the folding measure to wider activation classes (Swish, GELU, SwiGLU)
- Connects folding to learned symmetries
- Shows that folding can be used as a differentiable regularization term via the Motzkin-Straus theorem

9.4 FoldAndCutNetworks (Stewart et al., 2025)

In 2025, a group led by Stewart, Layton, Bennett, and Driggs introduced **FoldAndCutNetworks** — a neural network layer explicitly inspired by the fold-and-cut theorem. The key idea: replace standard ReLU layers with **fold layers** that have learnable hyperplanes (soft or hard folds). Each layer explicitly implements a folding operation rather than a pointwise nonlinearity.

This is the closest existing architectural implementation of the ideas the Puno Calculus describes. The fact that others independently arrived at fold-based layers validates the intuition that folding is a natural computational primitive for neural networks. The authors explicitly cite Keup & Helias (2022) and the origami analogy.

As of mid-2026, this remains a preprint/code release. No formal paper has been published on FoldAndCutNetworks.

9.5 The Fold-and-Cut Theorem Connection

The fold-and-cut theorem (Demaine, Demaine & Lubiw, 1998) states that any polygonal cut pattern can be produced from a sheet of paper by folding it flat and making one straight cut.

The connection to universal approximation is straightforward at the intuitive level:

- The paper = the input manifold
- Each fold = a ReLU layer transformation

- The cuts = the linear decision boundaries
- The theorem = ReLU networks can realize any bounded decision boundary

Two results deepen this connection:

Abel et al. (2014) proved that convex polyhedra in any dimension can be continuously flattened using straight skeletons. This provides an N-dimensional constructive method for folding — directly relevant to how ReLU networks create piecewise-linear decision boundaries in high-dimensional input spaces.

An, Demaine, Demaine & Ku (2017) proved that the fold-and-cut machine can decide 3SAT in polynomial time, making it at least as powerful as a nondeterministic Turing machine. The machine uses four operations (Fold, Unfold, Cut, Look) — and the "Look" operation (checking for through-holes) is what gives it NP power. Whether ReLU networks can simulate a fold-and-cut machine without a direct analogue of the Look operation is an open question with significant complexity-theoretic implications.

A formal proof of the ReLU-to-fold-and-cut connection in N dimensions would be a landmark result, establishing the Puno Calculus as a rigorous mathematical theory rather than a conceptual framework.

9.6 Robertson (1977): Isometric Folding of Riemannian Manifolds

Robertson's paper — independently discovered by Kawasaki, Justin, and Robertson in the late 1970s — generalizes Kawasaki's angle-sum condition to arbitrary dimensions. In Robertson's formulation, for an isometric folding f: M → N of n-manifolds, let S be the singular set decomposed into positive/negative n-dimensional strata. Then:

$$Vol^{+} - Vol^{-} = deg(f) \cdot Vol(M)$$

where deg(f) is the topological degree. For n = 1 (a circle around a vertex), this reduces to Kawasaki's alternating sum condition. For n = 2 (a sphere around a 3D vertex), it becomes a condition on the alternating sum of **ar eas** of spherical polygons cut by crease planes.

This is significant for the Puno Calculus because it means the angle-sum constraint is not specific to 2D paper folding. It is a manifestation of a deeper topological invariant that operates in any dimension. The question for neural networks: does this invariant constrain the geometry of ReLU decision region vertices?

9.7 No Kawasaki Analogue Exists

No Kawasaki theorem analogue exists for neural networks. Whether the angular constraints of flat origami have a counterpart in ReLU network geometry is a genuinely open question.

Robertson's 1977 generalization provides the mathematical framework. If the alternating-volume condition can be shown to constrain the geometry of ReLU decision region vertices, it would be the first major theorem of the Puno Calculus.

Chapter 10: The Broader Literature

10.1 Subgradient Selection at the Crease

Bertoin et al. (NeurIPS 2021) — In theory, any choice of ReLU'(0) in [0,1] is equivalent modulo the "bifurcation zone" (a measure-zero set in parameter space where at least one neuron receives exactly zero input). In practice (32-bit precision), the choice matters considerably — variation in backpropagation outputs occurs ~50% of the time. At 16-bit, the effect is systematic. ReLU'(0) = 0 is the most efficient choice for vanilla SGD, improving test accuracy by up to 10 points on ImageNet over ReLU'(0) = 1.

Boursier et al. (2022) — Gradient flow with ReLU'(0) = 0 converges to the minimum L2-norm interpolator on orthogonal data with vanishing initialization. Any other constant choice of the subgradient fails to yield global solutions. This is the strongest theoretical result connecting crease behavior to training outcomes.

SUGAR (2025) — "Surrogate Gradient for ReLU" (Horuz et al., arXiv:2505.22074) preserves standard ReLU in forward pass but replaces its derivative with a smooth surrogate (B-SiLU, NeLU). Improves generalization on VGG-16, ResNet-18, and even Conv2NeXt/Swin Transformer. Sparser activations, fewer dead ReLUs. This work directly addresses subgradient selection at the kink, though it treats it as a fixed architectural choice rather than an adaptive strategy.

S-ReLU (Zheng & Mo, 2026) — A general framework for activation function optimization based on mollification theory. Proposes Smoothed ReLU (S-ReLU), a C^2 activation derived from ReLU via mollification. Provides a systematic approach to smoothing nonsmooth activations.

Gap: No existing paper proposes an *adaptive, trajectory-aware* strategy for selecting subgradients at crease crossings. The choice is treated as either irrelevant in theory or a fixed hyperparameter in practice. The term "crease-aware optimization" does not appear in the ML literature.

10.2 Local Complexity of Linear Regions

Patel & Montúfar (ICML 2025) — *On the Local Complexity of Linear Regions in Deep ReLU Networks*. Define **local complexity** as the density of linear regions over the data distribution, and prove it bounds the total variation of the learned function. Show that optimization drives networks toward lower local complexity, and connect local complexity to adversarial robustness. This is the closest formal counterpart to crease density in the published literature — it measures the "density of bends" near data rather than the fraction of near-threshold units.

10.3 Computational Complexity of Linear Regions

Stargalla, Hertrich, Reichman (NeurIPS 2025) — Prove that counting linear regions in ReLU networks is #P-hard, even for 1-hidden-layer networks. Systematically categorize six non-equivalent definitions of "linear region" in the literature. This result is significant for the Puno Calculus: if counting regions is intractable, then cheap proxies like crease density become practically essential.

10.4 ReLU Transition Graphs

Dhayalkar et al. (2025) — Build graphs where nodes are linear regions and edges represent single-neuron flips. Measure entropy, degree distribution, and spectral gap of these transition graphs. This provides a structural view of the folding landscape that complements crease density's aggregate measurement.

10.5 Crease Density Novelty Assessment

The term "crease density" — the fraction of ReLU pre-activations within ε of zero — **does not appear in the published literature** as of mid-2026. The closest existing concepts:

Existing Concept	Source	Difference from Crease Density
Local complexity	Patel & Montúfar (ICML 2025)	Measures region density, not pre-activation proximity
Distance to boundary	Hanin & Rolnick (ICML 2019)	Per-input geometric distance, expensive to compute
Dormant neuron ratio	Sokar et al. (ICML 2023)	Counts fully-off units, not near-threshold ones
Bifurcation zone	Bertoin et al. (NeurIPS 2021)	Measure-zero set where $z(x) = 0$, not ε-band
Activation pattern entropy	Hartmann et al. (2021)	Binary on/off diversity, not continuous proximity
Space folding measure	Lewandowski et al. (TMLR 2025)	Hamming-range of activation paths
Binary NAP	Olber et al. (CVPR 2023)	Stores full activation pattern database

Crease density is computationally trivial (O(batch × neurons)), continuously informative (unlike binary on/off measures), and captures the graded proximity to switching boundaries that no existing metric isolates. Its novelty is confirmed.

10.6 Conservative Fields (Bolte & Pauwels, 2019-2021)

The conservative field framework provides the mathematical infrastructure for the Puno Calculus:

• **Conservative set valued fields, automatic differentiation, and deep learning** (2019): Introduces "conservative fields" as generalized derivatives satisfying a pathwise chain rule. Proves that backpropagation, even on nonsmooth networks, yields a conservative field. Establishes convergence in values of nonsmooth SGD.

• **A mathematical model for automatic differentiation** (2020): Models AD "as is" without modification. Shows that spurious behavior of AD at nonsmooth points is theoretically harmless.

• **The structure of conservative gradient fields** (2021): All conservative fields are Clarke subdifferentials plus normals of manifolds in underlying Whitney stratifications.

In the Puno Calculus framing:

• A **fold** corresponds to a choice of selection in a conservative field

• An **unfold** corresponds to integration along a path through the Whitney stratification

• The automatic differentiation system implements these operations without needing to know the fold/unfold language

10.7 Regime Change Hypothesis (2026)

Proposes that ReLU activation patterns (on/off regimes) stabilize earlier than weights during training. After regime stabilization, a ReLU network behaves locally as a collection of affine operators with frozen gates.

This directly supports the Puno Calculus view:

• **Fast phase**: Crease locations (activation boundaries) are established — the folding pattern is determined

• **Slow phase**: The affine operators within each region are refined — the fold angles are adjusted

• After regime stabilization, the network's geometry is fixed; only its linear coefficients change

10.8 Bifurcation Zones in RNN Training (2023)

Proves that degenerate transcritical bifurcations cause exploding gradients and border-collision bifurcations cause vanishing gradients in piecewise-linear RNNs. Establishes a formal link between crease behavior and extreme gradient events.

This connects the Puno Calculus to the training stability literature: the creases are not just where learning happens, they are where gradients can explode or vanish.

10.9 Competing Theoretical Frameworks

The Puno Calculus makes specific, testable predictions that distinguish it from other theoretical frameworks:

Framework	Relation	Crease Density Prediction	Key Test
Neural Tangent Kernel	Conflict	Density irrelevant (lazy regime, fixed kernel)	Wide nets: does density change → 0?
Information Bottleneck	Complement	Tracks I(X;T) compression phase	ReLU nets: density increases despite no IB compres
Geometric Deep Learning	Complement	Lower in equivariant networks	Equivariant vs. vanilla: same accuracy, lower densit
Renormalization Group	Deep complement	Analog of RG beta function	Density depth-profile matches MI saddle points?
Manifold hypothesis	Direct complement	intrinsic dimension increase	Density × dimension = constant per layer?
Depth separation	Direct support	complexity of target function	Density correlates with target's total variation?
Double descent	Potentially explanatory	Peaks at interpolation threshold	Density peak aligns with test error peak?
Mechanistic interpretability	Different scales	Progress measure for circuit formation	Step function at induction head phase change?

The strongest distinguishing test: at infinite width (NTK regime), crease density should approach a fixed initialization value because weights barely move. At finite width, crease density should track generalization. This is testable with existing infrastructure.

10.10 Applicability to Modern Architectures (GELU, Transformers)

Modern neural networks predominantly use GELU or SwiGLU activations (smooth, no exact crease) in transformer architectures (attention + FFN). Does the Puno Calculus apply?

GELU has a "soft crease" — while smooth everywhere (C^∞), it has a region of maximum curvature at $x = 0$ where $|\text{GELU}''(0)| \approx 0.798$. This is the mollified analogue of ReLU's kink (GELU = ReLU ∗ Gaussian). The "soft crease" can be defined as $\{x : |f''(x)| \geq \tau\}$ centered at the inflection point.

Generalization path: Lewandowski et al. (AAAI 2026) already generalized their space folding measure to GELU/Swish by binarizing activations at zero threshold — the binary pattern partition still exists even for smooth activations. The Puno Calculus can generalize crease density to smooth activations by replacing the hard ε-band count with a curvature-weighted integral:

Soft crease intensity: $C_\ell = E_{x\sim D}[\ |\sigma''(w_\ell^T x + b_\ell)| \cdot ||w_\ell||\ ]$

This counts how much the activation "bends" the input, weighted by data density. For ReLU, this reduces to the standard crease density (σ'' is a Dirac delta at 0). For GELU, it measures the expected curvature at decision boundaries.

Where folding lives in transformers: Attention computes convex combinations (no folding). The FFN sublayer (SwiGLU/GELU) provides the folding nonlinearity. The relevant folding analysis should target the FFN sublayer, not the attention mechanism.

ReLU is reviving: SUGAR (2025) replaces ReLU's backward derivative with a smooth surrogate, keeping forward sparsity. ReLU outperforms GELU in LayerNorm-free LLMs (NeurIPS 2024 workshop). The field is actively re-engineering ReLU for modern architectures. The crease-based view is increasingly relevant.

Chapter 11: What This Framing Does and Does Not Do

What it does

1. **Gives an intuitive, physically grounded name** to the derivative/integral relationship. "Fold" and "unfold" describe what these operations *mean*, not just how to compute them.

2. **Centers creases and corners as primary objects** rather than edge cases. The subdifferential is not a workaround for nondifferentiability — it is the exact description of what happens at the crease.

3. **Connects calculus, convex analysis, origami mathematics, and deep learning theory** under a single conceptual umbrella. Results from origami (Kawasaki, Maekawa, fold-and-cut) and nonsmooth optimization (conservative fields, bifurcation zones) are revealed as variations on the same theme.

4. **Suggests novel, testable directions**: crease-aware optimization, crease density diagnostics, crease-engineered activations, OOD detection via crease ambiguity, pruning via crease proximity.

5. **The ReLU-as-origami connection is mathematically established** (Keup & Helias 2022, Lewandowski et al. 2025 TMLR). The Puno Calculus did not discover this connection, but it places it in a broader context and draws out its practical implications.

6. **Crease density is a genuinely novel metric.** No existing paper in the published literature measures the fraction of near-threshold ReLU units as a training diagnostic, OOD signal, or pruning criterion. The closest prior art (local complexity, dormant neuron ratio, space folding measure) captures related but distinct quantities.

7. **The Boursier et al. (2022) result provides rigorous theoretical support:** different choices of ReLU'(0) determine whether gradient flow has a globally defined solution at all. The crease is not a curiosity — it is the point where optimization theory lives.

8. **The fold/unfold framing is genuinely original.** A comprehensive search of Western philosophy (Neoplatonism, Cusanus, Leibniz, Goethe, Hegel, Spencer-Brown, Deleuze, Bohm, Lautman), category theory (catamorphism/anamorphism), and Eastern thought (I Ching) found **no prior source that explicitly defines "derivative = fold, integral = unfold, crease = primary object of study."** The closest anticipations are Spencer-Brown's *Laws of*

Form (formal calculus of distinction), Deleuze's *The Fold* (philosophy of the inflection point), and the catamorphism/anamorphism pair from functional programming. The specific inversion — making non-differentiable points primary and differentiable points the special case — is novel.

What it does not do

1. **It does not prove new theorems** (in itself). The Puno Calculus is a reframing, not a discovery in the theorem-proving sense. Its value is in the connections it reveals and the experiments it suggests.
2. **The Kawasaki analogue for neural networks remains an open question**. The relationship between origami angle constraints and ReLU decision region geometry is speculative, though Robertson's 1977 N-dimensional generalization provides a potential framework.
3. **The connection between subdifferentials and origami constraints is conceptual, not formal**. Subdifferentials are permissive (many slopes valid). Origami constraints are restrictive (only specific angles valid). These are complementary views of the same phenomenon but they are not the same mathematics.
4. **The practical impact on ML is unproven at scale.** Our experiments are on small problems (checkerboard, rings, MLPs). Scaling to ImageNet, transformers, or LLMs would require PyTorch/TensorFlow implementations and substantial compute.

Part IV: Practical Tools

Chapter 12: The Puno Calculus Software Suite

The Puno Calculus project includes reusable tools built during experimentation. All tools are self-contained (numpy + matplotlib only) and designed for easy extension.

12.1 puno_utils.py

Core utilities:

```
class Net:
    """MLP with crease tracking."""
    def forward(self, x, track_creases=False):
        # Returns (logits, list_of_layer_crease_counts)

    def forward_per_input(self, x):
        # Returns (logits, per_input_crease_array)

class CreaseMonitor:
    """Training callback. Records crease density per epoch.

    monitor = CreaseMonitor()
    for epoch in range(N):
        # training step...
```

```
    monitor.record(avg_crease, val_acc, val_loss)
  monitor.plot('monitor.png')
  """

class OODScorer:
  """Per-input crease density for OOD detection.

  scorer = OODScorer(model)
  raw, density = scorer.score(x)
  raw_all, density_all = scorer.score_batch(X)
  """
```

12.2 Data Generators

```
def make_multiscale(n=4000):
  """Multi-scale checkerboard: coarse outer, fine inner."""
```

12.3 Training Loop

```
def train_model(model, X_tr, y_tr, X_va, y_va, lr=1e-3, epochs=300, batch=128, monitor=None):
  """Standard training loop with CreaseMonitor support."""
```

12.4 Available Demos

File	Purpose
`demo_ood.py`	OOD detection vs MSP, 4 OOD types, AUROC evaluation
`exp_pruning.py`	Crease proximity vs magnitude vs random pruning
`exp3_early_stop.py`	Crease stabilization vs val loss plateau early stopping

Part V: Frontiers

Chapter 13: Open Problems

The Puno Calculus opens more questions than it answers. Here are the open problems, ordered by tractability:

13.1 Crease-Aware Subgradient Adaptation

Problem: Boursier et al. (2022) proved that only ReLU'(0) = 1 guarantees globally defined gradient flow solutions, but ReLU'(0) = 0 (standard in frameworks) performs better at finite precision. The theory-practice gap suggests an *adaptive* strategy: select the subgradient based on training trajectory, using 1 when theoretical guarantees matter, 0 when practical performance dominates.

Status: No prior art. The term "crease-aware optimization" is available for definition.

Approach: A simple adaptive strategy: when a neuron's pre-activation passes through zero during training, the subgradient is selected to push it toward the side it came from (stabilizing the switching boundary) or toward the side with lower gradient variance. The SUGAR approach (2025) provides a related but distinct strategy: replace the derivative entirely with a smooth surrogate function.

Difficulty: Hard. Requires modifying the autograd graph at each ReLU node. Custom autograd Functions in PyTorch.

13.2 The Kawasaki Analogue

Problem: Does there exist a constraint, analogous to Kawasaki's theorem in origami, that governs the meeting of ReLU decision boundaries at a point? Robertson's 1977 generalization suggests one may exist in N dimensions.

Status: Completely open. No one has explored this connection.

Approach: Study the geometry of linear region vertices in ReLU networks. At a point where multiple ReLU hyperplanes intersect, what is the local structure? Does the alternating angle sum condition hold in 2D slices?

Difficulty: Very hard. Requires new mathematics at the intersection of convex geometry and discrete differential geometry.

13.3 Minimum Fold Depth

Problem: What is the minimum number of ReLU layers needed to represent a given classification function? This is the "minimum fold depth" — the computational complexity measure that the Puno Calculus predicts should exist.

Status: Embryonic. Valerdi (2024) introduces a notion of depth complexity for convex piecewise-linear functions. Telgarsky (2016) proves depth separation results using folding constructions. But no one has formalized "fold depth" as a complexity measure.

Approach: Define fold depth as the minimum number of ReLU layers needed to realize a function, where each layer corresponds to a folding operation. Connect to the circuit complexity literature.

Difficulty: Hard. Requires connecting-depth separation results to the origami framing.

13.4 Crease-Engineered Activation Functions

Problem: Can activation functions designed with explicit crease geometry outperform standard activations?

Status: SUGAR (2025) shows that modified ReLU gradients improve generalization. The design space of crease-engineered activations is underexplored.

Approach: Design activations with controlled crease properties:

- **Hard crease**: Sharp corner (ReLU, leaky ReLU)
- **Soft crease**: Smooth transition (GELU, Swish)
- **Periodic creases**: Multiple switching points (periodic activations)
- **Adaptive creases**: Learnable crease location/gradient

Difficulty: Moderate. Implementation is straightforward; finding good designs requires experiment.

13.5 Crease Density as a Training Diagnostic

Problem: Can crease density serve as a practical training diagnostic in real-world training runs? Specifically:

- Can it predict generalization without a validation set?
- Can it detect overfitting (crease density increasing in late training)?
- Can it guide learning rate scheduling?

Status: Experiment 3 shows crease density is a more sensitive measure of ongoing learning than val loss. The practical applications need validation at scale.

Approach: Instrument PyTorch training loops with crease density hooks. Test on CIFAR-10/100, ImageNet. Compare crease density trajectories for overfitting vs. generalizing models.

Difficulty: Moderate. Requires PyTorch hooks but no new theory.

13.6 The Fold-and-Cut Initialization

Problem: If ReLU networks fold space like paper, should weight initialization mimic optimal origami crease patterns?

Status: No prior art.

Approach: The fold-and-cut theorem provides constructive crease patterns for any desired cut. Can we map these patterns to network initialization? A network initialized with pre-determined fold lines (appropriate for the target function) would start training with the correct folding structure and only need to refine angles.

Difficulty: Very hard. Requires translating crease patterns to network weights.

13.7 Crease Density for Adversarial Robustness

Problem: Lewandowski et al. (2026, AAAI) connect space folding to adversarial geometry. Can crease density serve as a measure of adversarial vulnerability? Networks with high crease density have many units operating near threshold — are these networks more vulnerable to small perturbations that push units over the crease?

Status: Suggested by the literature but not tested in the Puno Calculus experiments.

Approach: Compute crease density of trained networks. Generate adversarial examples with PGD/FGSM. Correlate crease density with adversarial robustness.

Difficulty: Moderate.

13.8 ReLU Networks as Fold-and-Cut Machines

Problem: The fold-and-cut machine (An et al., 2017) solves NP-complete problems via folding. Do ReLU networks implement the same kind of computation? The FoldAndCutNetworks project (Stewart et al., 2025) is the first explicit attempt to build fold-based layers — but no one has characterized the computational power of ReLU networks as folding machines.

Status: Open. The fold-and-cut machine's "Look" operation (checking for through-holes) has no direct analogue in standard ReLU networks. Without it, ReLU networks may be strictly weaker than the fold-and-cut machine.

Approach: Formalize the computational model of ReLU folding. Characterize which operations from the fold-and-cut machine can be simulated and which cannot. This connects to circuit complexity: if ReLU networks can simulate fold-and-cut operations, then sufficiently deep networks can solve problems in NP.

Difficulty: Very hard. Requires connecting the folding geometry literature to computational complexity theory.

13.9 Conservative Fields Formalization

Problem: Can the Puno Calculus be formalized in the language of conservative fields (Bolte & Pauwels)?

Status: The connection is suggestive but not established.

Approach: Show that a "fold" corresponds to a choice of selection in a conservative field, and an "unfold" corresponds to integration along a path through the Whitney stratification of activation space. This would provide the Puno Calculus with rigorous mathematical foundations.

Difficulty: Hard. Requires expertise in nonsmooth analysis and differential geometry.

13.10 Crease Density Formalization Program

Five mathematical directions have been identified that could formalize crease density as a rigorous object:

Direction	Feasibility	Key Reference	Formal Target
Total variation ↔ Crease density	Easy**	Patel et al. (ICML 2025)	Crease density = distributional derivative of TV(f')
PL Morse theory	Easy	Grigsby et al. (2022)	Critical point density ≤ crease density
Clarke subdifferential measure	Moderate**	Fatuła (2024)	μ_∂f(S) = ∫_S diam(∂f(x)) dx
Whitney stratification distance	Moderate**	Mostow (1985)	Lip constant ≤ C / distance to crease set
Lipschitz constant via crease counting	Moderate**	Avant & Morgansen (2023)	# distinct Lip values = # crease crossings
Shattered gradients	Hard	Balduzzi et al. (2017)	GC(f, x, ε) = # linear regions in B_ε(x)

The nearest-term result: proving that crease density equals the jump measure of the total variation of the derivative — a direct formalization of "creases are where the function changes slope."

Chapter 14: A Call to Fold

The Puno Calculus is not finished. It is barely begun.

What we have:

- A conceptual framing that connects calculus, convex analysis, origami, and deep learning
- Five experiments that validate specific predictions
- A literature review that confirms the novelty of several directions
- Reusable software tools
- A map of open problems

What we need:

- A proof (or disproof) that the fold-and-cut theorem generalizes to N dimensions and connects to universal approximation
- A Kawasaki analogue for ReLU decision region geometry
- A practical implementation of crease-aware subgradient adaptation
- Large-scale validation of crease density diagnostics on ImageNet, transformers, and LLMs
- A formalization in the language of conservative fields

The Book of Puno is a living document. As experiments are run, theorems are proved, and tools are built, this book will grow. The fold/unfold framing has proven its worth by suggesting novel experiments and connecting previously isolated literatures. Whether it becomes a full mathematical theory or remains a productive conceptual framework depends on the work that comes next.

The crease is where the action is. The fold is the fundamental operation. The unfold is how we reconstruct the world.

Let's see what we can fold.

Appendices

Appendix A: Key Definitions

Subdifferential $\partial f(x) = \{ m \in R : f(y) \geq f(x) + m(y - x) \text{ for all } y \in R \}$. The set of slopes m such that the line through (x, f(x)) with slope m lies entirely below the graph of f.

Convex function: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all x, y and $\lambda \in [0, 1]$. The graph lies below the chord connecting any two points.

Fold depth: The number of ReLU layers in a network. Correlates with decision boundary complexity at r = +0.968.

Crease density: The number (or fraction) of ReLU units whose pre-activation falls within threshold ε of zero during a forward pass. Used as training diagnostic, OOD signal, and pruning criterion.

Crease proximity pruning: A pruning criterion that removes neurons with highest crease density. Outperforms magnitude pruning up to 30% removal.

Bifurcation zone: The set of parameters in a ReLU network where at least one neuron receives exactly zero input (Bertoin et al., 2021). The set where subgradient choice is theoretically relevant.

Flat-foldable vertex: A vertex in a crease pattern where paper can be folded flat without introducing additional creases. Governed by Kawasaki's theorem.

Fold-and-cut theorem: Any polygonal cut pattern can be produced by folding paper flat and making one straight cut (Demaine, Demaine & Lubiw, 1998).

Space folding measure: $\chi(\Gamma) = 1 - \max_i d_H(\pi_1, \pi_i) / \max_j d_H(\pi_1, \pi_j)$ where d_H is Hamming distance in binarized activation space (Lewandowski et al., 2025).

Appendix B: Software

All code is available at the Puno Calculus repository. Core files:

File	Lines	Purpose
`puno_utils.py`	~200	Core utilities (numpy) — Net, CreaseMonitor, OODScorer, da
`puno_torch.py`	~250	PyTorch implementation — CreaseDensityHook, OODScorer,
`demo_ood.py`	~220	OOD detection demo (numpy)
`exp_pruning.py`	~200	Pruning experiment (numpy)
`exp3_early_stop.py`	~350	Early stopping experiment (numpy)
`build_book_pdf.py`	~440	Book PDF generation from markdown
`build_experiment_reports.py`	~280	Extracts experiment chapters as standalone PDFs

Dependencies: numpy, matplotlib, scipy (for correlations only), fpdf (for PDF generation).

Appendix C: Citation Guide

When referencing the Puno Calculus in academic work:

Conceptual: "Puno Calculus" — the fold/unfold framing of calculus operations.

Method: "crease-aware optimization" — optimization methods that account for gradient behavior at nondifferentiable points.

Metric: "crease density" — the frequency with which ReLU pre-activations fall near zero.

Criterion: "crease proximity pruning" — pruning neurons based on crease density.

Referencing this book:

> *Puno, M. G. S. (2026). *The Book of Puno: A Treatise on Folding, Unfolding, and What Lives at the Crease* (2nd ed.).*

Everything folds. Everything unfolds. The crease is where the action is.
— The Book of Puno

The Book of Puno

"Everything folds. Everything unfolds. The crease is where the action is."

This book was set in Calibri (body) and Cambria (titles).
Printed from the Second Edition, July 2026.

End of the Book of Puno.