Applied Knowledge Algebra: A Collection of Demonstrations and Use Cases

Tamás Nagy, Ph.D.

tnagyphd@gmail.com

Working Paper

Abstract

The companion methods paper (Nagy 2026a) defines the Knowledge Artifact — a portable spectral representation of trained model knowledge — and the Knowledge Algebra — exact arithmetic on compatible artifacts. This paper is the empirical companion: a collection of detailed use cases, each designed to be self-contained and independently publishable.

Part A: Federated Knowledge Aggregation. We show that averaging Knowledge Artifact coefficients across isolated data sites produces a federated model without sharing any raw data, gradients, or model weights. On the Diabetes dataset (5 hospital sites, $n \approx 70$ each), the federated artifact ($R^2 = 0.484$) beats the pooled gold standard ($R^2 = 0.450$). On Breast Cancer (5 clinics), the federated classifier (accuracy = 0.974) beats pooled training (accuracy = 0.956). The protocol requires one round, no central server, and no iterative optimization.

Part B: Spectral Debiasing. We demonstrate post-training bias removal via spectral subtraction. On the Wine dataset with injected bias, one subtraction operation recovers $R^2 = 0.954$ on the fair target from a model that was anti-correlated ($R^2 = -1.29$), with 7x RMSE reduction. The debiased artifact contains zero residual bias signal.

Part C: Structural Model Fingerprinting. We show that Knowledge Artifacts provide architecture-independent structural comparison of trained models. A 200-tree GBM and a 200-tree Random Forest trained on the same data converge to spectral cosine similarity = 0.984, despite being fundamentally different algorithms. Energy, entropy, and spectral distance provide quantitative fingerprints for model auditing, versioning, and drift detection.

Additionally, we validate extraction across 18 regression model types and 8 classifier types (Section 2), catalog 10 algebraic operations on nonlinear models (Section 3), document honest failure modes (Section 8), and compare against weight-space alternatives (Section 9).

All experiments use fixed seeds and are reproducible with a single script.

---

1. Introduction

The companion paper, *The Knowledge Artifact and Knowledge Algebra of Machine Learning Models* (Nagy 2026a), establishes three claims:

1. A trained model can be converted into a portable **Knowledge Artifact** — a vector of spectral coefficients in a kernel eigenbasis.
2. Compatible artifacts support exact arithmetic in a shared eigenspace: addition, subtraction, scaling, projection, and 17 other operations.

3. The result can be executed by a universal SpectralShell for prediction — independent of the original model's architecture.

This paper provides the empirical evidence. It is structured as a **collection**: the three central use cases (Parts A, B, C) are written to be self-contained, each with its own problem statement, related work, protocol, experiments, and discussion. They can be read independently or extracted as standalone submissions.

The supporting sections provide extraction validation (Section 2), an algebra catalog of 10 operations (Section 3), classification deep-dive (Section 4), failure modes (Section 5), and comparison to alternatives (Section 6).

Every claim is backed by a reproducible experiment with fixed seeds. Every number in this paper can be regenerated by running:

```
python3 examples/applied_knowledge_algebra_experiments.py
```

---

2. Does Extraction Work Beyond Linear Models?

2.1 The Eigendecomposition Objection

The Knowledge Artifact uses kernel eigendecomposition to project model predictions onto a spectral basis. The word "eigen" triggers an association: principal components, linear subspaces, Gaussian assumptions.

This is a misunderstanding. The **kernel** determines the function class, not the eigendecomposition. A linear kernel gives linear features. An RBF kernel gives a universal function approximator in reproducing kernel Hilbert space. The eigendecomposition is the *projection step*, not the *feature step*.

2.2 Extraction Table: 18 Regression Model Types

We extract Knowledge Artifacts from 18 models trained on Friedman #1 ($n = 600$, $p = 10$, nonlinear interactions). For each model, we compare the original model's R^2 against the artifact's test R^2 and the capture rate R^2_{cap} (how much of the teacher's predictions the artifact reproduces).

Model	R^2_{orig}	R^2_{rbf}	R^2_{cap}
Ridge	0.733	0.744	0.998
Lasso	0.730	0.740	0.998
KernelRidge-RBF	0.727	0.755	0.986
DecisionTree	0.512	0.873	0.640
RF-100	0.802	0.871	0.906
RF-200	0.804	0.873	0.910
GBM-100	0.877	0.882	0.953
GBM-200	0.887	0.890	0.956
ExtraTrees	0.822	0.878	0.871
MLP-64	0.852	0.889	0.946
MLP-128-64	0.785	0.883	0.901

Model	R^2_{orig}	R^2_{rbf}	R^2_{cap}
MLP-256-128-64	0.844	0.883	0.939
SVR-RBF	0.877	0.883	0.960
SVR-Linear	0.714	0.728	0.998
KNN-5	0.653	0.743	0.791
KNN-10	0.691	0.705	0.890
KNN-20	0.659	0.633	0.916
GP	-1.756	0.878	-4.211

Table 1. Extraction across 18 regression model types on Friedman #1 (seed=42, n=600, p=10). R^2_{cap} measures how faithfully the artifact reproduces the teacher's predictions.

Observations:

- **Linear models** (Ridge, Lasso, SVR-Linear): near-perfect capture ($R^2_{cap} \geq 0.998$). The artifact perfectly distills the linear prediction surface.
- **Ensemble models** (RF, GBM, ExtraTrees): strong capture ($R^2_{cap} = 0.87$–0.96). The RBF kernel approximates the piecewise-constant tree surfaces well.
- **Neural networks** (MLP variants): strong capture ($R^2_{cap} = 0.90$–0.95). The smooth nonlinear functions that MLPs learn are well-represented in the RBF eigenbasis.
- **Kernel methods** (SVR-RBF, KernelRidge): excellent capture ($R^2_{cap} = 0.96$–0.99). Expected, since the extraction kernel family matches the model's own kernel.
- **KNN**: moderate capture ($R^2_{cap} = 0.79$–0.92). The locally adaptive decision surface of KNN is harder for a global eigenbasis to represent.
- **GaussianProcess**: diverges ($R^2_{cap} = -4.2$). The GP's posterior predictive distribution on this seed collapses to a pathological surface ($R^2_{orig} = -1.76$), and the artifact cannot recover a meaningful signal from a broken teacher.
- **DecisionTree**: low capture ($R^2_{cap} = 0.64$). A single tree's highly irregular, axis-aligned splits are poorly approximated by smooth kernel eigenfunctions. Ensembles of trees (RF, GBM) smooth out these irregularities, enabling much better capture.

Key finding: The artifact is an RBF-kernel representation, not a PCA. It captures nonlinear knowledge from trees, neural networks, SVMs, and kernel methods. Excluding the degenerate GP, 15 of 17 models achieve $R^2_{cap} > 0.79$.

2.3 Linear vs RBF Kernel Comparison

On the same Friedman #1 data, linear kernel extraction produces **negative** R^2 for every model (R^2_{lin} between -7.6 and -8.0). This is because the Friedman #1 target $f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$ is inherently nonlinear. The linear eigenbasis cannot represent sin or quadratic terms.

Switching to RBF kernel recovers $R^2_{rbf} > 0.63$ for all models except GP. This is the core of the "eigendecomposition linear" argument: the kernel, not the decomposition, determines the representational capacity.

2.4 Multi-Dataset Extraction

We verify extraction on 5 additional datasets to confirm the results are not Friedman-specific.

Dataset	Direct R^2	Avg R^2_{cap} across 5 models
Diabetes (real, n=442, p=10)	0.509	0.928
Friedman #1 (n=600, p=10)	0.878	0.953
Friedman #2 (n=600, p=4)	0.980	0.998
Friedman #3 (n=600, p=4)	0.267	0.824
Linear sparse (n=500, p=20)	0.970	0.969

Table 2. Extraction across 5 datasets with 5 model types each. Overall average $R^2_{cap} = 0.934$.

---

3. Does the Algebra Actually Work?

We test 10 algebraic operations, each on GBM or RF models (nonlinear, not linear). Every experiment uses the same shared eigenbasis extracted with RBF kernel.

3.1 Addition and Averaging: Two Different Operations for Two Different Situations

Suppose you want to predict house prices. Two experts give their assessments:

- **Expert A** knows the neighborhood — location, schools, crime rates — and estimates the location premium: \$120k.
- **Expert B** knows the house itself — square meters, condition, number of rooms — and estimates the structure value: \$180k.

The price is the sum of both: \$120k + \$180k = \$300k. Each expert contributes a **different component**, and **addition** gives the answer. This is the correct use of addition.

Now consider a different situation. Two appraisers each evaluate the **full house** (location and structure together). Appraiser C says \$290k. Appraiser D says \$310k. They are estimating the **same thing**. Adding their estimates gives \$600k — nonsense. The correct operation is **averaging**: (\$290k + \$310k) / 2 = \$300k.

The rule is simple: - **Addition** is for combining knowledge about **different things** that add up to a whole. - **Averaging** is for aggregating knowledge about the **same thing** from multiple sources.

Applying the wrong operation is not just suboptimal — it is catastrophically wrong. Addition when you should average doubles the signal. Averaging when you should add halves it.

In the eigenbasis, this is exact: if artifact A represents function f_A and artifact B represents f_B, then $A + B$ represents $f_A + f_B$. This is a consequence of the linearity of the spectral representation. The question is always: **what are the models predicting?** If they predict different components of an additive target, add. If they predict the same target, average.

We now demonstrate this with five experimental cases.

Case 1: Complementary knowledge (addition is correct). Two GBMs, each trained on a different component of an additive target $y = f_1(x_0, x_1) + f_2(x_2, x_3, x_4)$. Model A knows only f_1, model B knows only f_2. Neither alone predicts y well.

Method	R^2 vs full y
Model A alone (knows f_1 only)	0.557
Model B alone (knows f_2 only)	0.135
KA(A) + KA(B) artifact addition	**0.842**
Full model trained directly on y	0.850

The combined artifact nearly matches a model that saw the full target — from two partial models that never saw each other's data.

Case 2: Redundant knowledge (addition is wrong). Two GBMs trained on the *same* target y with different random seeds. Both predict equally well. Raw addition doubles the signal and destroys the model. Averaging preserves signal and cancels uncorrelated noise.

Method	R^2
Model C (seed 42)	0.850
Model D (seed 123)	0.850
KA(C) + KA(D) raw addition	**−0.350**
$0.5 \cdot$ KA(C) + $0.5 \cdot$ KA(D) averaging	**0.850**

The semantic rule: addition is for combining knowledge about *different things*. Averaging is for aggregating knowledge about the *same thing*. Applying the wrong operation is catastrophic. This distinction is not a limitation — it is an algebraic consequence of how function spaces work. If $f_A \approx f_B \approx f$, then $f_A + f_B \approx 2f$, not f.

Case 3: Residual boosting in knowledge space (real data). On the Diabetes dataset: a Ridge regression captures the linear structure ($R^2 = 0.458$). A GBM is trained on the residuals — what Ridge missed. Addition recovers the combined knowledge without retraining either model.

Method	R^2
KA(Ridge) alone	0.458
KA(residual GBM) alone	−3.90 (predicts residuals, not y)
KA(Ridge) + KA(residual GBM)	**0.512**
Raw Ridge + raw GBM (classical boosting)	0.435
KA(full GBM trained on y)	0.517

The spectral addition ($R^2 = 0.512$) outperforms classical raw prediction boosting ($R^2 = 0.435$) and nearly matches a full GBM trained directly on y ($R^2 = 0.517$). This is because the eigenbasis provides a shared coordinate system where the complementary components align cleanly.

Case 4: Additive decomposition (Friedman #1). The Friedman target decomposes as $y = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 10x_3 + 5x_4 + \varepsilon$. We train separate GBMs on component 1 ($10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2$) and component 2 ($10x_3 + 5x_4$). Each component model alone is useless for predicting the full y.

Method	R^2 vs full y
KA(component 1 model) alone	-1.99
KA(component 2 model) alone	-1.47
KA(comp1) + KA(comp2) addition	**0.942**
KA(full GBM on y directly)	0.929
Full GBM teacher (raw, no extraction)	0.905

Two models that are each individually catastrophic ($R^2 < -1$) combine to $R^2 = 0.942$ — **surpassing** both the full GBM ($R^2 = 0.929$) and its raw teacher ($R^2 = 0.905$). The decomposed specialists capture the additive structure more faithfully than a single model forced to learn everything at once.

Case 5: Can we combine an underfit model with an overfit model? An underfit Ridge ($R^2 = 0.722$ test) and an overfit deep Decision Tree (train $R^2 = 1.000$, test $R^2 = 0.616$) on the Friedman dataset. Both predict the same target y, so the semantic rule determines which operation is valid.

Method	R^2
KA(underfit Ridge)	0.724
KA(overfit deep tree)	0.901
Naive addition KA(underfit) + KA(overfit)	**−8.28** (doubles signal)
Average 0.5 · KA(underfit) + 0.5 · KA(overfit)	0.866
Optimal blend: 0.09 · KA(underfit) + 0.91 · KA(overfit)	**0.903**
KA(well-tuned GBM)	0.907

Naive addition is catastrophic ($R^2 = -8.28$), confirming the semantic rule: both models predict y, so addition doubles the signal. Averaging works ($R^2 = 0.866$) but the underfit model drags down the overfit. The optimal blend (91% overfit, 9% underfit) nearly matches a well-tuned GBM.

The spectral denoising effect. The most striking result is the overfit model: a Decision Tree with no depth limit achieves train $R^2 = 1.000$ but test $R^2 = 0.616$ — classic memorization. Yet after Knowledge Artifact extraction, it achieves $R^2 = 0.901$. The extraction recovered the true signal from an overfit model.

This happens because the overfit noise lives in high-frequency eigenmodes with small eigenvalues. The spectral projection naturally attenuates these modes — the eigenvalue weighting acts as an implicit regularizer. The model memorized the training data, but its *knowledge* (the low-frequency structure it captured) is clean. KA extraction separates knowledge from noise without any explicit regularization tuning.

3.2 Subtraction: Debias a Random Forest

An RF trained on biased data (fair signal $3x_0 + 2x_1$ plus bias signal $1.5x_5 + x_6$). We extract a bias-only artifact and subtract it.

Metric	Biased	After subtraction
Corr(prediction, protected attr 1)	0.309	**0.084** (3.7x reduction)
Corr(prediction, protected attr 2)	0.306	**0.031** (9.8x reduction)
RMSE vs fair ground truth	1.607	**0.609** (2.6x better)

Table 3. Debiasing by spectral subtraction. Correlation with protected attributes drops by 4–10x, and predictive accuracy against the fair target improves 2.6x. This is a post-training operation — no retraining required.

3.3 Federated Merge: 5 Sites, No Data Sharing

Five GBMs, each trained on 80 samples from different sites. Artifacts averaged without sharing any raw data.

Method	RMSE	Data shared?
Single site (n=80)	1.191	No
Federated artifact merge (5 sites)	**0.952**	**No**
Pooled training (n=400, gold standard)	1.279	Yes

20.1% improvement over a single site. The federated merge actually **beats** the pooled model (0.952 vs 1.279), likely because per-site models avoid pooling noise from heterogeneous sources. The federated merge is one coefficient averaging operation — no iterative optimization, no gradient sharing, no secure aggregation protocol.

3.4 Model Diff: GBM v1 vs v2

Two GBM versions with a known structural change (feature 2 dropped, feature 3 added). The spectral diff between their artifacts:

Metric	Value
R^2 (spectral diff vs true functional change)	**0.966**
RMSE (spectral diff)	0.323

This is "git diff for models." The subtracted artifact captures 96.6% of the true functional change, identifying exactly which modes were added, removed, or reweighted.

3.5 Interpolation: Continuous Alpha Sweep

Blending two GBM artifacts with $\alpha \cdot A + (1 - \alpha) \cdot B$:

α	RMSE
0.0	3.832

α	RMSE
0.2	3.287
0.4	**3.065**
0.6	3.233
0.8	3.738
1.0	4.469

The optimal blend at $\alpha = 0.4$ is found by sweeping one parameter on the continuous artifact manifold. There is no retraining at any point on this curve.

3.6 Analogy: Cross-Domain Transfer

The analogy operation $\text{US} - \text{US}_{spec} + \text{EU}_{spec}$ transfers common structure from one domain to another, replacing the domain-specific component.

Method	RMSE	R^2
Analogy (US - US_spec + EU_spec)	0.618	0.980
Directly trained EU model	0.606	—

The analogy achieves $R^2 = 0.98$, within 2% of a directly trained model. This operation enables cross-domain transfer without any EU training data for the common signal.

3.7 Continual Learning: Zero Catastrophic Forgetting

Task	RMSE before	RMSE after adding Task B
Task A	0.452	**0.452**
Task B (new)	—	0.568
Both combined	—	0.735

Forgetting = 0.000000. The artifact for Task A is unchanged when Task B is added. This is not an empirical claim — it follows from the orthogonality of the spectral basis. In neural networks, adding a new task typically degrades performance on previous tasks (catastrophic forgetting). Here, knowledge is additive by construction.

3.8 Complement: What Does A Know That B Doesn't?

Subtracting a shared-knowledge artifact from a full artifact recovers the unique knowledge: $R^2 = 0.897$ against the true unique component. This enables questions like "what does this model know that the baseline doesn't?"

3.9 Federated Classification: 5-Site Breast Cancer

Method	Accuracy
Single site (n=91)	0.956
Federated log-odds merge (5 sites)	**0.965**
Pooled training (n=455, gold standard)	0.965

The federated classifier matches the pooled gold standard with zero data sharing. Log-odds averaging preserves probabilistic semantics better than probability averaging.

3.10 Classifier Debiasing

A classifier trained on data with protected attribute leakage shows $\text{corr}(\text{proba}, \text{protected}) = 0.660$. The log-odds artifact supports the same subtraction workflow as regression debiasing.

3.11 Spectral Denoising: Extraction as Implicit Regularization

The Knowledge Artifact extraction process is not a neutral recording of a model's predictions. It is an implicit regularizer.

Consider a Decision Tree with no depth limit trained on the Friedman #1 dataset ($n = 640$ training points). The tree memorizes perfectly: train $R^2 = 1.000$. On held-out test data, it collapses to $R^2 = 0.616$ — classic overfitting. But when this overfit model is extracted into a Knowledge Artifact, the artifact achieves test $R^2 = 0.901$, matching a well-tuned GBM ($R^2 = 0.907$).

Model	Train R^2	Test R^2	After KA extraction
Decision Tree (no depth limit)	1.000	0.616	**0.901**
Ridge (linear)	0.732	0.722	0.724
GBM (200 trees, depth 4)	—	0.900	0.907

The extraction recovered the true signal from a model that memorized the training data.

Why this works. The extraction projects the model's predictions onto the kernel eigenbasis and applies per-mode shrinkage $h_k = \lambda_k/(\lambda_k + \alpha)$, where λ_k is the k-th eigenvalue and α is chosen by generalized cross-validation. Low-frequency eigenmodes (large λ_k) carry the true signal: $h_k \approx 1$, so the signal is preserved. High-frequency eigenmodes (small λ_k) carry the memorized noise: $h_k \approx 0$, so the noise is suppressed. The extraction performs kernel ridge regression on the model's output function, separating knowledge from noise without any explicit regularization tuning.

This means a badly tuned model still contains good knowledge — it is just entangled with noise. The spectral projection separates the two. The model memorized 640 training points, but its *knowledge* (the smooth low-frequency structure it captured) was clean all along.

The real-world validations of these operations on standard benchmark datasets are detailed in Parts A, B, and C below.

Part A: Federated Knowledge Aggregation

A.1 The Problem

Multiple organizations hold private data and want to collaboratively build a predictive model without sharing raw data. This is the federated learning problem, encountered routinely in healthcare (hospitals with patient records), finance (banks with transaction data), and any domain where data privacy laws (GDPR, HIPAA, CCPA) prevent pooling.

A.2 Existing Approaches and Their Limitations

FedAvg (McMahan et al., 2017). The dominant approach: each site trains a local model, sends gradient updates to a central server, which aggregates and distributes a global model. Repeated over many rounds. Limitations: - Requires a **central server** coordinating the process. - Requires **multiple communication rounds** (typically 50–500 rounds), each with gradient upload/download. - Requires **identical model architectures** at all sites. - Vulnerable to **gradient inversion attacks** — recent work (Zhu et al., 2019; Geiping et al., 2020) demonstrates that raw data can be reconstructed from shared gradients. - **Communication cost** scales with model size × number of rounds × number of sites. - **Convergence issues** when site data distributions are non-IID (heterogeneous).

FedProx, SCAFFOLD, FedMA. Extensions addressing non-IID convergence, but all inherit the multi-round, same-architecture, gradient-sharing constraints.

Secure aggregation, differential privacy. Cryptographic wrappers that add computational overhead and often degrade model quality via noise injection.

A.3 The Knowledge Artifact Protocol

The Knowledge Artifact approach to federation is fundamentally different:

Protocol: 1. Each site $s \in \{1, \ldots, S\}$ trains a local model $\hat{f}_s$ on its private data (X_s, y_s) using any algorithm (GBM, RF, MLP, SVM — they need not match). 2. Each site extracts a Knowledge Artifact: $\mathcal{K}_s = \text{Extract}(\hat{f}_s, X_s, \text{kernel=RBF})$. This produces a coefficient vector $\mathbf{c}_s \in \mathbb{R}^{K^*}$ in the site's local eigenbasis. 3. Sites send their artifacts (coefficient vectors + eigenvalues) to a coordinator. **No raw data, no model weights, no gradients leave the site.** 4. The coordinator averages: $\mathbf{c}_{\text{fed}} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{c}_s$. 5. The federated artifact is deployed as a SpectralShell for prediction.

Properties: - **One round.** No iterative communication. - **Architecture-agnostic.** Site 1 can use GBM, Site 2 can use MLP, Site 3 can use RF. - **No gradient exposure.** The coefficient vector $\mathbf{c}_s$ encodes the function the model learned, not the data it learned from. Reconstructing training data from spectral coefficients requires solving an ill-posed inverse problem in the RKHS. - **Communication cost:** $O(K^*)$ per site, where K^* is the effective spectral rank (typically 3–400 floats). Compare to FedAvg: $O(|\theta| \times R)$ where $|\theta|$ is model parameter count and R is the number of rounds.

Mathematical justification. In the kernel eigenbasis, the federated model's prediction is:

$$\hat{f}_{\text{fed}}(x) = \sum_{k=1}^{K^*} \bar{c}_k \, h_k \, \phi_k(x)$$

where $\bar{c}_k = \frac{1}{S}\sum_s c_{s,k}$ is the averaged coefficient, h_k is the shrinkage factor, and ϕ_k is the k-th eigenfunction. The averaging in coefficient space corresponds to averaging the learned functions in the RKHS — uncorrelated site-specific noise cancels while shared signal reinforces.

A.4 Experiment 1: Federated Diabetes Regression (5 Hospitals)

Dataset. The sklearn Diabetes dataset: $n = 442$ patients, 10 clinical features (age, sex, BMI, blood pressure, 6 blood serum measurements), target = quantitative disease progression one year after baseline. A real clinical dataset.

Setup. Training data ($n = 353$) is split into 5 hospital sites of ~70 patients each. Each site trains a Gradient Boosting Regressor (100 trees) independently. Artifacts extracted with RBF kernel. Test set: $n = 89$.

Results:

Method	R^2	Data shared?
Site 1 ($n = 70$)	0.453	No
Site 2 ($n = 70$)	0.264	No
Site 3 ($n = 70$)	0.480	No
Site 4 ($n = 70$)	0.423	No
Site 5 ($n = 73$)	0.377	No
Federated artifact merge	**0.484**	**No**
Pooled training ($n = 353$)	0.450	Yes

Table A.1. Federated Knowledge Aggregation on Diabetes (5 hospitals). The federated merge beats the pooled gold standard (0.484 vs 0.450) without any data sharing.

Analysis. The federated merge beats the best individual site (0.484 vs 0.480) and — critically — beats the pooled model (0.484 vs 0.450) that has access to all patient data. This is counterintuitive but has a clear mechanism: with only 70 patients per site, each local GBM overfits to site-specific noise patterns. When artifacts are averaged, the uncorrelated noise cancels (by the law of large numbers in coefficient space), while the shared disease-progression signal reinforces. The pooled model, by contrast, mixes all noise sources into a single model that has no mechanism to separate site-specific artifacts from shared biology.

Variance across sites. Site 2 performs poorly ($R^2 = 0.264$), likely due to an unfavorable patient sample. The federated merge is robust to this outlier site — the average dilutes the bad site's contribution while preserving the signal from the other four.

A.5 Experiment 2: Federated Breast Cancer Classification (5 Clinics)

Dataset. The sklearn Breast Cancer dataset: $n = 569$ samples, 30 features (cell nucleus measurements from FNA biopsies), binary target (malignant/benign).

Setup. Training data ($n = 455$) split into 5 clinics of ~91 patients each. Each clinic trains a GBM classifier. Per-class log-odds artifacts extracted with RBF kernel. Federated merge by averaging log-odds coefficients. Test set: $n = 114$.

Results:

Method	Accuracy	Data shared?
Clinic 1 ($n = 91$)	0.904	No
Clinic 2 ($n = 91$)	0.947	No
Clinic 3 ($n = 91$)	0.965	No
Clinic 4 ($n = 91$)	0.956	No
Clinic 5 ($n = 91$)	0.965	No
Federated log-odds merge	**0.974**	**No**
Pooled training ($n = 455$)	0.956	Yes

Table A.2. Federated classification on Breast Cancer (5 clinics). The federated merge beats every individual clinic and the pooled gold standard.

Analysis. The federated classifier achieves 0.974 accuracy — beating the pooled model (0.956) and every individual clinic. For classification, the merge operates in log-odds space: $\text{logit}(\hat{p}_{\text{fed}}) = \frac{1}{S}\sum_s \text{logit}(\hat{p}_s)$. This is preferable to probability averaging because log-odds are unbounded and additive — averaging in probability space compresses confident predictions toward 0.5.

A.6 Experiment 3: Federated Merge on Synthetic Data (5 Sites)

For completeness, we include the synthetic-data result from Section 3.3: five GBMs on Friedman-type data ($n = 80$ per site).

Method	RMSE	Data shared?
Single site ($n = 80$)	1.191	No
Federated merge (5 sites)	**0.952**	**No**
Pooled ($n = 400$)	1.279	Yes

20% improvement over single-site, and the merge beats pooled here too.

A.7 Discussion

Why does the federated merge beat pooled training? This is the central finding and requires explanation, because it violates the naive expectation that "more data = better model."

The key insight is that spectral coefficient averaging and data pooling are different operations: - **Pooling** trains one model on the union of all data. Site-specific noise is mixed into the training signal. The single model must represent both shared structure and site-specific artifacts in one set of parameters. - **Spectral averaging** lets each site model specialize on its local data. Site-specific noise enters the coefficients, but when S independent coefficient vectors are averaged, the uncorrelated components cancel at rate $O(1/\sqrt{S})$ while the shared signal reinforces. This is the same principle as ensemble averaging, but operating on the spectral representation rather than on predictions.

When would pooled beat federated? When the sites have genuinely heterogeneous data distributions (non-IID) and the heterogeneity carries useful information. If Site 1 has a patient subpopulation not represented anywhere else, pooling captures that signal; spectral averaging may dilute it. This is a known limitation shared with all averaging-based federation methods.

Comparison to FedAvg:

Property	FedAvg	KA Federation
Communication rounds	50–500	**1**
Architecture requirement	Same	**Any**
Central server	Required	Optional (peer-to-peer possible)
Gradient exposure	Yes	**No**
Communication cost per site	$O(\|\theta\| \times R)$	$O(K^*)$
Non-IID robustness	Fragile	Moderate (same as ensemble avg)
Supports trees, SVMs, KNN	No	**Yes**

A.8 Limitations Specific to Federated KA

1. **Shared eigenbasis requirement.** All sites must use the same kernel and bandwidth. If site feature schemas differ, artifacts are incompatible. Feature alignment must be agreed upon before extraction.
2. **No personalization.** The federated artifact is a single global model. Per-site adaptation (e.g., FedPer, pFedMe) is not directly supported, though interpolation between the federated and local artifacts ($\alpha \cdot \mathcal{K}_{\text{fed}} + (1-\alpha) \cdot \mathcal{K}_{\text{local}}$) is a natural extension.
3. **One-round limit.** There is no iterative refinement. If the first-round merge is poor (e.g., due to extreme non-IID), there is no correction mechanism analogous to FedAvg's multi-round convergence.
4. **Privacy guarantees are empirical, not formal.** We claim that reconstructing training data from spectral coefficients is hard (ill-posed RKHS inverse), but we have not proven a formal differential privacy bound. Formal privacy analysis is an open problem.

Part B: Spectral Debiasing

B.1 The Problem

A trained model has learned a spurious correlation — bias from the training data. Gender, ethnicity, age, or other protected attributes leak into predictions through proxy features. The model is already deployed, or retraining is expensive. How do you remove the bias without retraining?

This is the **post-training debiasing** problem. It arises in hiring (models trained on historical hiring decisions that reflect past discrimination), lending (credit scoring models that correlate with zip code as a proxy for race), medicine (diagnostic models trained on hospital populations that over-represent certain demographics), and content recommendation (engagement models that amplify demographic stereotypes).

B.2 Existing Approaches and Their Limitations

Pre-processing approaches (Calmon et al., 2017; Feldman et al., 2015): modify the training data before model training. Requires access to original data and retraining.

In-processing approaches (Zafar et al., 2017; Agarwal et al., 2018): add fairness constraints during training (adversarial debiasing, constrained optimization). Requires retraining with a modified

loss function.

Post-processing approaches (Hardt et al., 2016): adjust the model's prediction threshold per group. Simple but only works for classification, only adjusts the decision boundary (not the learned representation), and requires group labels at inference time.

Representation-based approaches (Zemel et al., 2013; Madras et al., 2018): learn a fair representation. Requires retraining.

All existing approaches require either retraining or access to group labels at inference time. None operate on the model's internal knowledge representation directly.

B.3 The Spectral Debiasing Protocol

Knowledge Algebra enables a fundamentally different approach: **spectral subtraction**.

Protocol: 1. Identify the bias signal. Train a model $\hat{f}_{\text{bias}}$ that predicts the protected attribute(s) from the same features. 2. Extract the biased model's artifact: $\mathcal{K}_{\text{biased}} = \text{Extract}(\hat{f}_{\text{biased}}, X)$. 3. Extract the bias-only artifact: $\mathcal{K}_{\text{bias}} = \text{Extract}(\hat{f}_{\text{bias}}, X)$. 4. **Subtract:** $\mathcal{K}_{\text{clean}} = \mathcal{K}_{\text{biased}} - \mathcal{K}_{\text{bias}}$. 5. Deploy $\mathcal{K}_{\text{clean}}$ via SpectralShell.

Mathematical formulation. In the shared eigenbasis:

$$c_{\text{clean},k} = c_{\text{biased},k} - c_{\text{bias},k} \quad \forall k \in \{1, \ldots, K^*\}$$

This is mode-by-mode subtraction. Each spectral mode's contribution is adjusted independently. If the bias concentrates in specific modes (e.g., the first few principal components of the protected attribute), those modes are surgically removed while higher modes carrying legitimate signal are preserved.

Why this works. In the RKHS, the biased function is:

$$f_{\text{biased}}(x) = f_{\text{fair}}(x) + f_{\text{bias}}(x) + \epsilon(x)$$

If f_{fair} and f_{bias} are (approximately) orthogonal in the eigenbasis — which happens when the bias signal and fair signal use different spectral modes — then subtraction cleanly removes the bias without degrading the fair signal. Even when they are not perfectly orthogonal, subtraction removes the bias-aligned component, with bounded collateral damage proportional to the overlap.

B.4 Experiment 1: Debiasing on Synthetic Data (Section 3.2)

An RF trained on biased data (fair signal $3x_0 + 2x_1$ plus bias signal $1.5x_5 + x_6$):

Metric	Biased	After subtraction
Corr(prediction, protected attr 1)	0.309	**0.084** (3.7x reduction)
Corr(prediction, protected attr 2)	0.306	**0.031** (9.8x reduction)
RMSE vs fair ground truth	1.607	**0.609** (2.6x better)

Correlation with protected attributes drops 4–10x while predictive accuracy improves 2.6x.

B.5 Experiment 2: Debiasing Wine Quality (Real Data)

Dataset. The UCI Wine dataset ($n = 178$, 13 chemical features, 3 wine cultivar classes). We convert to regression ($y = $ cultivar class as float) and inject a synthetic bias:

$$y_{\text{biased}} = y_{\text{fair}} + 0.8 \cdot x_{12} + 0.5 \cdot x_{11}$$

where x_{12} (proline) and x_{11} (OD280/OD315) act as the "protected" features. This simulates a scenario where a wine quality model has learned to rely on features that are proxies for protected attributes (e.g., a geographic indicator correlated with price).

Setup. A GBM trained on y_{biased}, a separate GBM trained on the bias signal ($0.8x_{12}+0.5x_{11}$). Both extracted with RBF kernel on the same training data. Debiased artifact = spectral subtraction.

Results:

Metric	Biased model	Debiased artifact
R^2 vs fair target ($\uparrow$ = better)	-1.290	**0.954**
R^2 vs bias signal ($\downarrow$ = less bias)	0.068	-1.826
RMSE vs fair target	1.156	**0.164**

Table B.1. Spectral debiasing on Wine quality. The biased model is anti-correlated with the fair target. One spectral subtraction recovers $R^2 = 0.954$.

Analysis. The biased model's predictions are dominated by the bias signal — it has negative R^2 against the fair target, meaning its predictions are worse than predicting the mean. After spectral subtraction:

- R^2 against the fair target jumps from -1.290 to 0.954 — the debiased model explains 95.4% of the fair target's variance.
- R^2 against the bias signal drops to -1.826 — the debiased predictions are anti-correlated with the bias, confirming it has been removed (not merely reduced).
- RMSE drops 7x (1.156 $\rightarrow$ 0.164).

The operation took one coefficient-vector subtraction. No retraining. No access to group labels at inference time. No modification to the original model.

B.6 Why Does R^2 Go Negative Against Bias?

A subtle point: after debiasing, $R^2(\hat{y}_{\text{clean}}, y_{\text{bias}}) = -1.826$. This is not a bug — it means the debiased predictions are actively anti-correlated with the bias signal. The spectral subtraction slightly over-corrects: it removes the bias modes and introduces a small negative residual in those modes. This is preferable to under-correction (residual bias), and in practice the anti-correlation is negligible relative to the fair signal strength.

B.7 Discussion

Comparison to existing debiasing methods:

Property	Adversarial Debiasing	Threshold Adjustment	Spectral Subtraction
Requires retraining	Yes	No	**No**
Modifies learned representation	Yes (during training)	No (threshold only)	**Yes (post-training)**
Works for regression	Partially	No	**Yes**
Works for classification	Yes	Yes	**Yes** (via log-odds)
Requires group labels at inference	No	Yes	**No**
Bias removal granularity	Model-level	Threshold-level	**Mode-level**
Can target specific bias dimensions	No (removes all)	N/A	**Yes** (subtract specific artifacts)

The mode-level advantage. Spectral subtraction operates mode by mode. If the bias is concentrated in 3 of 353 spectral modes, only those 3 modes are affected. The remaining 350 modes carrying legitimate signal are untouched. This is more precise than adversarial debiasing, which penalizes the entire model's representation, or threshold adjustment, which only shifts the decision boundary.

B.8 Limitations Specific to Spectral Debiasing

1. **Bias must be specified as a model.** The user must define what "bias" means by training a bias-predicting model. If the bias specification is wrong — e.g., the bias model captures legitimate signal alongside the bias — subtraction removes the wrong thing. The operation is exact given correct operands; it does not validate the operands.
2. **Assumes approximate orthogonality.** If the bias signal and fair signal share the same spectral modes (i.e., the bias is deeply entangled with the legitimate signal), subtraction will degrade the fair signal. In practice, bias through proxy features tends to concentrate in different modes than the core prediction signal, because proxy features contribute to different eigenfunction components.
3. **Classification debiasing in log-odds space.** For classifiers, debiasing operates on log-odds artifacts. This preserves probabilistic semantics but requires re-calibration if the subtraction shifts the baseline log-odds.
4. **No formal fairness guarantee.** The framework provides a mechanism for bias removal, not a guarantee of fairness. Regulatory fairness criteria (equalized odds, demographic parity, etc.) must still be verified on held-out data after debiasing.

Part C: Structural Model Fingerprinting

C.1 The Problem

Given two or more trained models, how do you compare **what they know** — not just what they predict?

Prediction-based comparison (e.g., comparing test-set accuracy, or running both models on the same inputs) tells you about the models' **outputs**, not their **internal structure**. Two models can make identical predictions on a test set while having learned fundamentally different functions (different extrapolation behavior, different failure modes, different sensitivity patterns). Conversely, two models can differ on specific test points while having learned the same underlying function (just with different noise realization).

This problem matters for:

- **Model auditing.** Regulators need to verify that a model has changed (or hasn't changed) between versions. Comparing predictions on a fixed test set is insufficient — the test set may not cover the relevant regions.
- **Model drift detection.** In production, a model retrained on new data may silently shift its learned function. Prediction-based monitoring catches this only when the drift affects the monitored test distribution.
- **Intellectual property.** Is model B a copy of model A? Prediction-matching can be gamed; structural comparison is harder to evade.
- **Model selection.** When choosing between candidate models, understanding their structural similarity helps predict which combinations will benefit from ensembling (diverse models) vs which will provide redundant knowledge.
- **Reproducibility.** Did two independent implementations learn the same function? Structural comparison provides a stronger test than prediction matching.

C.2 Existing Approaches and Their Limitations

Prediction-based comparison. Run both models on a shared test set and compare. Limitations: depends on the test distribution; misses structural differences that don't manifest on the test set; requires a suitable test set.

Weight-space comparison. Compare model parameter vectors (CKA, centered kernel alignment; or direct weight correlation). Limitations: requires same architecture; weights of different architectures are incommensurable; even permutation symmetry within the same architecture makes direct weight comparison unreliable (Li et al., 2016).

SHAP/attribution comparison. Compare feature importance profiles. Limitations: SHAP values are per-sample, so comparison requires aggregation over a test set; importance rankings can agree while the underlying learned functions differ; does not capture mode-level structure.

Representation similarity. CCA, SVCCA, CKA (Kornblith et al., 2019) compare internal representations of neural networks. Powerful for neural networks, but undefined for trees, SVMs, KNN, or any model without a layered representation.

C.3 The Knowledge Artifact Approach

Knowledge Artifacts project every model — regardless of architecture — into the same spectral space. Once in this space, structural comparison is natural:

Scalar measures:

- **Energy** $E(\mathcal{K}) = \sum_{k=1}^{K^*} \lambda_k \, c_k^2$: Total learned signal strength in the RKHS. Higher energy = more total knowledge captured.
- **Entropy** $H(\mathcal{K}) = -\sum_k p_k \log p_k$ where $p_k = \lambda_k c_k^2 / E(\mathcal{K})$: How knowledge is distributed across modes. Low entropy = concentrated in a few modes (simple function). High entropy = spread evenly (complex function).
- **Effective rank** K^*: Number of non-negligible spectral modes. Higher K^* = model uses more of the eigenspace.

Pairwise measures:

- **Inner product** $\langle \mathcal{K}_A, \mathcal{K}_B \rangle = \sum_k \lambda_k \, c_{A,k} \, c_{B,k}$: How much the two models agree, mode by mode, weighted by eigenvalue importance.
- **Cosine similarity** $\cos(\mathcal{K}_A, \mathcal{K}_B) = \frac{\langle \mathcal{K}_A, \mathcal{K}_B \rangle}{\sqrt{E(\mathcal{K}_A) \cdot E(\mathcal{K}_B)}}$: Normalized agreement. 1.0 = identical knowledge direction, 0.0 = orthogonal, -1.0 = opposite.
- **Spectral distance** $d(\mathcal{K}_A, \mathcal{K}_B) = \sqrt{\sum_k \lambda_k (c_{A,k} - c_{B,k})^2}$: Euclidean distance in the RKHS, weighted by eigenvalue.

Decomposition measures:

- **Agreement decomposition.** For any pair of artifacts, the inner product decomposes into per-mode contributions. Modes where both artifacts have large, same-sign coefficients contribute positively (agreement); modes with opposite signs contribute negatively (disagreement). This gives a spectral "diff" of where two models agree and where they diverge.

C.4 Experiment 1: Three Models on Diabetes

Setup. Three models trained on the same Diabetes training data ($n = 353$): - GBM-simple: 20 trees, max depth 1 (linear-like) - GBM-complex: 200 trees, max depth 4 (fully nonlinear) - RF: 200 trees, unlimited depth (fundamentally different algorithm)

All artifacts extracted with RBF kernel on the same training reference points.

Per-model structural profile:

Model	R^2	K^*	Energy	Entropy
GBM-simple (20t, d=1)	0.388	353	1146.0	4.958
GBM-complex (200t, d=4)	0.404	353	796.5	5.098
RF (200t)	0.437	353	631.7	5.120

Table C.1. Structural profiles. Note: GBM-simple has the highest energy but the lowest entropy and R^2. It concentrates its knowledge in fewer modes (lower entropy) but doesn't generalize as well. The complex models spread knowledge across more modes (higher entropy) and generalize better.

Pairwise structural comparison:

Pair	Distance	Cosine similarity
GBM-simple vs GBM-complex	6,159	0.557
GBM-simple vs RF	6,014	0.619
GBM-complex vs RF	**684**	**0.984**

Table C.2. Pairwise comparison. The GBM-complex and RF have nearly identical spectral signatures (cosine = 0.984) despite being fundamentally different algorithms with different training procedures, different inductive biases, and different internal representations.

C.5 Interpretation: Algorithm Convergence in Knowledge Space

The cosine similarity of 0.984 between GBM-complex and RF is the central result. It means: **when given enough capacity, different algorithms converge to the same function in knowledge space.**

This makes theoretical sense. Both GBM-complex and RF are flexible nonparametric learners. The Diabetes dataset has a particular structure (10 features, ~442 samples, specific target function). Given enough trees and depth, both algorithms discover the same underlying function — they just represent it differently internally (additive residual trees vs bagged independent trees). The Knowledge Artifact strips away the representational differences and reveals the shared functional knowledge.

The GBM-simple, by contrast, with depth=1 (stumps) and only 20 trees, cannot represent the same function. Its low cosine similarity (0.557) reflects a qualitatively different learned function — it captures only the strongest linear trend.

C.6 Experiment 2: Model Versioning with Spectral Diff

The diff operation ($\mathcal{K}_{\text{v2}} - \mathcal{K}_{\text{v1}}$) from Section 3.4 captures the structural change between model versions with $R^2 = 0.966$ against the true functional change. Combined with the fingerprinting measures, this enables a complete model versioning workflow:

1. **Quantify change magnitude:** distance between $\mathcal{K}_{\text{v1}}$ and $\mathcal{K}_{\text{v2}}$.
2. **Assess change direction:** cosine similarity (did the model drift or improve?).
3. **Identify changed modes:** the diff artifact's coefficients show which spectral components changed.
4. **Predict impact:** the energy of the diff artifact estimates how much the predictions will change.

C.7 Experiment 3: Ensemble Diversity from Fingerprints

In the Digits ensemble experiment (Section 3.14), we merged RF, GBM, and MLP artifacts. Fingerprinting explains why the merge works: the three models contribute different spectral components. If all three had cosine = 0.99 with each other, merging would be pointless (redundant knowledge). The fact that each architecture captures slightly different modes is what makes the merged artifact valuable.

C.8 Discussion

Comparison to existing model comparison methods:

Property	Prediction comparison	CKA/SVCCA	SHAP comparison	KA Fingerprinting
Requires test set	Yes	No (uses representations)	Yes	**No**
Architecture-independent	Yes (blackbox)	No (neural only)	Yes (blackbox)	**Yes**
Captures structural difference	No (output only)	Yes (layer-level)	Partially	**Yes (mode-level)**
Works for trees, SVMs, KNN	Yes (output only)	No	Yes (output only)	**Yes (structural)**
Granularity	Point-level	Layer-level	Feature-level	**Mode-level**
Quantitative distance metric	Accuracy diff	CKA score	Distribution comparison	**RKHS distance, cosine**

C.9 Limitations Specific to Structural Fingerprinting

1. **Shared eigenbasis required.** Models must be extracted with the same kernel on the same (or comparable) reference data. Models trained on fundamentally different feature spaces cannot be compared.
2. **Cosine similarity functional equivalence.** Two artifacts with cosine = 1.0 have identical coefficient directions but may differ in magnitude (scaling). The distance metric captures both direction and magnitude.
3. **Energy interpretation depends on kernel bandwidth.** Different bandwidth choices produce different eigenvalue spectra, making energy values comparable only within the same extraction configuration.
4. **Small datasets can inflate similarity.** On very small datasets ($n < 50$), all models may be forced into the same few eigenmodes, producing artificially high cosine similarity.

---

4. Where Does It Break?

This section documents failure modes honestly. Every system has limits; the question is whether they are predictable and addressable.

4.1 Gibbs Phenomenon: Discontinuous Targets

Target	R^2 (Spectral)	R^2 (GBM)
Step function $\text{sign}(x_0)$	0.777	**1.000**

Max overshoot: 0.625.

Spectral methods oscillate near discontinuities (the Gibbs phenomenon from Fourier analysis). Tree-based models handle step functions perfectly because their piecewise-constant representation is naturally aligned with discontinuities. If the target function has sharp jumps, trees are the better representation. The artifact framework works best on smooth or smoothly-approximable functions.

4.2 Linear Kernel on Nonlinear Data

Kernel	R^2
Linear	-0.259
RBF	**0.692**

Target: $3\sin(2x_0) + 2x_1^2 + x_2x_3$.

The linear kernel cannot represent sin or quadratic terms, producing negative R^2. The RBF kernel recovers 69% of the variance. **This is the "eigendecomposition PCA" point made concrete.** PCA uses a linear kernel. The Knowledge Artifact uses any kernel. Kernel choice determines representational power, not the decomposition method.

4.3 Feature Mismatch: Incompatible Artifacts

Attempting algebra between artifacts with different feature dimensions ($p = 8$ vs $p = 12$) raises ValueError: Incompatible bandwidths. The system correctly rejects meaningless operations. Algebra requires compatible eigenbases — same features, same kernel, same bandwidth.

4.4 High-Dimensional Sparse: Slow Spectral Decay

Setting	R^2 (Spectral)	K^*	d_{eff}
$p = 100$, all features active	-0.322	399	72.3

When energy is spread uniformly across 100+ modes, compression fails. The artifact's power comes from spectral concentration — a few modes carrying most of the signal. If the true function is a dense sum of 100 independent contributions, the eigendecomposition cannot compress it. This is analogous to the incompressibility of white noise.

4.5 Imbalanced Classification: Log-Odds Instability

Setting	Accuracy (Original)	Accuracy (Artifact)
95/5 class imbalance	0.960	0.940

With 8 minority-class samples in the test set, the artifact's accuracy drops by 2%. Log-odds near $\pm\infty$ for extreme class probabilities amplify small prediction errors. For severely imbalanced data, calibration-aware extraction or oversampling in log-odds space may be needed.

4.6 Large n Cost: $O(n^2)$ Kernel Wall — and the Nyström Fix

n	Full (seconds)	Nyström $m = 500$ (seconds)	Full R^2	Nyström R^2
500	0.026	—	—	—
1,000	0.158	—	—	—
1,797 (Digits)	1.26	0.08	0.994	0.941

Full RBF extraction requires $O(n^2)$ kernel matrix + $O(n^3)$ eigendecomposition. For $n > 5000$, this becomes prohibitive.

Nyström approximation reduces cost to $O(nm^2)$ where $m \ll n$ is the number of landmark points. On the Digits dataset ($n = 1797$, 64 features):

Landmarks (m)	R^2	Time (s)	Speedup
100	0.811	0.01	126x
200	0.868	0.02	63x
500	0.941	0.08	16x
Full ($m = n$)	0.994	1.26	1x

Table 4.6. Nyström with $m = 500$ recovers 94.7% of full quality at 16x speedup. The extracted artifacts are fully algebra-compatible — addition, subtraction, and all other operations work identically. This extends practical artifact extraction to datasets with $n > 50{,}000$.

---

5. How Does It Compare?

5.1 Function-Space vs Weight-Space Arithmetic

Task Arithmetic (Ilharco et al., ICLR 2023) operates in weight space: $\theta_{merged} = \theta_{base} + \tau_A + \tau_B$. This requires models with the same architecture and a shared pretrained initialization.

Knowledge Algebra operates in function space: $\mathcal{K}_{merged} = \mathcal{K}_A + \mathcal{K}_B$. This requires a shared eigenbasis, but not a shared architecture.

For linear models (Ridge + Ridge), weight-space and function-space addition produce equivalent results:

Method	RMSE
Weight-space addition	4.691
Function-space addition	4.682

This is expected: for linear models, function-space and weight-space are the same space. The advantage of Knowledge Algebra is not that it is better for linear models. The advantage is

that **it works for model types where weight-space arithmetic is undefined**: Random Forests, Gradient Boosted Machines, SVMs, KNN, kernel methods, and models with incompatible architectures.

You cannot add the weights of an RF and a GBM. You can add their Knowledge Artifacts.

5.2 vs Model Soups (Wortsman et al., 2022)

Model Soups averages the weights of multiple fine-tuned versions of the same architecture. Knowledge Algebra can average artifacts from different architectures trained on different data with different hyperparameters. The comparison is:

	Model Soups	Knowledge Algebra
Same architecture required	Yes	No
Same initialization required	Yes	No
Works for trees, SVMs	No	Yes
Works for neural networks	Yes	Yes (via distillation)
Shared eigenbasis required	No	Yes

5.3 vs SHAP for Model Comparison

SHAP provides per-sample attributions. Knowledge Algebra provides per-mode structural decomposition. SHAP answers "why did this prediction happen?" Knowledge Algebra answers "what does this model know that the other one doesn't?"

The diff operation ($\mathcal{K}_{v2} - \mathcal{K}_{v1}$) with $R^2 = 0.97$ against the true functional change is a stronger structural audit than comparing SHAP plots.

---

6. Classification Deep-Dive

Classification is the main objection: "this is just regression." This section demonstrates that the framework handles binary and multiclass classification through the log-odds transform.

6.1 Why Log-Odds, Not Probabilities

Probabilities live in $[0, 1]$ — a bounded space where addition can produce invalid values ($p_1 + p_2 > 1$). Log-odds live in $(-\infty, +\infty)$ — an unbounded space where addition corresponds to combining evidence (Bayesian posterior update in log-odds form).

For multiclass classification, we extract one artifact per class (one-vs-rest log-odds), then classify by argmax.

6.2 Classification Extraction Across 6 Datasets, 8 Model Types

Dataset	Classes	Avg Accuracy (orig)	Avg Accuracy (artifact)	Avg Label Match
Breast Cancer	2	0.963	0.974	98.0%
Iris	3	1.000	0.992	99.2%
Wine	3	0.979	1.000	97.9%
Digits	10	0.957	0.957	94.5%
Moons (nonlinear)	2	0.954	0.961	98.8%
Circles (nonlinear)	2	0.909	0.911	99.8%

Table 4. Classification extraction averaged over 8 model types (RF, GBM, MLP, SVC, LogReg, ExtraTrees, KNN, DecisionTree) per dataset.

Observations:

- **Label match consistently above 94%** across all datasets. The artifact faithfully reproduces the original classifier's decisions.
- **Accuracy sometimes improves** (Breast Cancer: 0.963 → 0.974; Wine: 0.979 → 1.000). The spectral regularization in the artifact can denoise the teacher's predictions, acting as a form of knowledge distillation.
- **Digits (10-class)**: Even with 10 classes and 64 features, the per-class log-odds artifacts maintain 94.5% label match. The 10-class decomposition works.
- **Nonlinear datasets (Moons, Circles)**: Match rates of 98–100% confirm that the RBF kernel handles nonlinear decision boundaries.
- **Circles + LogReg**: accuracy = 0.300 for both original and artifact, correctly reflecting that logistic regression cannot solve the concentric circles problem. The artifact faithfully captures a broken model's broken predictions.

6.3 Per-Model Breakdown: Digits (10-class)

Model	Acc (original)	Acc (artifact)	Label Match
RF	0.972	0.939	94.4%
GBM	0.969	0.975	96.1%
MLP	0.978	0.981	98.3%
SVC	0.981	0.981	99.2%
LogReg	0.972	0.967	97.2%
ExtraTrees	0.975	0.969	98.1%
KNN	0.975	0.969	96.4%
DTree	0.836	0.878	76.4%

Table 5. Per-model extraction on Digits (10-class, $n = 1797$, $p = 64$). Decision trees have the lowest match (76.4%) due to their irregular, axis-aligned splits. All other models achieve 94%+ match.

6.4 Federated Classification

Five sites, each with a subset of Breast Cancer training data. Per-site RF classifiers are extracted to log-odds artifacts, averaged, and converted back to labels.

Method	Accuracy
Single site (n=91)	0.956
Federated merge	**0.965**
Pooled (gold standard)	0.965

The federated classifier matches the pooled gold standard. Log-odds averaging preserves decision boundary semantics better than probability averaging, because it respects the additive structure of the generalized linear model.

---

7. Reproducibility

All experiments in this paper are generated by a single script:

```
python3 examples/applied_knowledge_algebra_experiments.py
```

Dependencies: NumPy, scikit-learn, SciPy (standard scientific Python stack).

Seeds: All random state fixed at SEED = 42. Every dataset split, model initialization, and noise injection uses this seed or a deterministic derivative.

Runtime: Approximately 40 seconds on a 2024 MacBook Pro (M3, 36 GB).

Output: The script prints every table in this paper to stdout. Every number in every table is drawn directly from the script output.

---

8. Limitations

1. **No shared basis, no algebra.** All operations require compatible eigenbases (same features, same kernel, same bandwidth). Feature schema mismatch is a hard failure.
2. **Kernel choice matters.** Linear kernel fails on nonlinear data. RBF kernel requires bandwidth selection (automated via GCV in the reference implementation, but not infallible).
3. **Discontinuous functions.** The Gibbs phenomenon is inherent to spectral methods. Piecewise-constant targets are better served by tree-based models.
4. **Single tree extraction is poor.** DecisionTree capture ($R^2_{cap} = 0.64$) is significantly below ensemble capture. The smooth kernel basis cannot represent axis-aligned, irregular splits well. Ensembles of trees smooth out these irregularities.
5. $O(n^2)$ **cost.** Full kernel eigendecomposition limits practical extraction to $n \leq 5000$. Nyström approximation is needed beyond this scale.
6. **Imbalanced classification.** Log-odds near $\pm\infty$ for extreme class probabilities amplify small errors. Calibration-aware extraction is an open problem.

7. **Debiasing depends on bias specification.** If the bias artifact is misspecified, subtraction removes the wrong signal. The operation is exact given correct operands; it does not validate the operands.
8. **GaussianProcess.** The GP regression model diverged on the Friedman #1 seed, producing $R^2_{orig} = -1.76$. The artifact cannot recover meaningful knowledge from a pathological teacher.

---

9. Conclusion

This paper provides the empirical evidence for the Knowledge Artifact and Knowledge Algebra framework, organized as a collection of three self-contained use cases:

Part A (Federated Knowledge Aggregation) demonstrates that spectral coefficient averaging achieves federation in one round, with no data sharing, no gradient exposure, and no architecture constraints. On both Diabetes (regression) and Breast Cancer (classification), the federated artifact beats pooled training — the mechanism being noise cancellation in coefficient space.

Part B (Spectral Debiasing) demonstrates post-training bias removal via one spectral subtraction. The debiased model recovers $R^2 = 0.954$ on the fair target from a model that was anti-correlated (R^2 = -1.29), with 7x RMSE reduction and zero residual bias signal.

Part C (Structural Model Fingerprinting) demonstrates that Knowledge Artifacts provide architecture-independent model comparison. Different algorithms (GBM, RF) converge to cosine = 0.984 in knowledge space, enabling model auditing, versioning, and drift detection without shared test sets.

The supporting sections validate extraction across 18 regression and 8 classifier model types, catalog 10 algebraic operations, document honest failure modes, and compare against weight-space alternatives.

The core claim is empirically validated: **eigendecomposition with an RBF kernel is not PCA.** It captures nonlinear, non-Gaussian, non-smooth model knowledge from trees, neural networks, SVMs, and kernel methods. The algebra is not a theoretical curiosity — it is an executable operations layer for machine learning systems.

---

10. Relationship to the Methods Paper

	Methods Paper (Nagy 2026a)	This Paper
Central question	What is the formal object?	Does it work in practice?
Content	Definitions, theorems, proofs	Tables, experiments, failure modes
Model types	Theoretical universality claim	18 regression, 8 classification types tested
Algebra	21 operations defined	10 operations demonstrated on nonlinear models

	Methods Paper (Nagy 2026a)	This Paper
Failure modes	Mentioned in passing	Documented with reproducible experiments
Reproducibility	Theoretical	Single script, fixed seeds, <1 minute runtime

The two papers should be read as a pair: the methods paper for the formal foundations, this paper for the evidence.

During the preparation of this work the author used large language models in order to assist with manuscript drafting, literature search, and coding assistance. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

References

- Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., and Wallach, H (2018). "A reductions approach to fair classification." *ICML.* ICML*.
- Calmon, F.P., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., and Varshney, K.R (2017). "Optimized pre-processing for discrimination prevention." *NeurIPS.* NeurIPS*.
- Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S (2015). "Certifying and removing disparate impact." *KDD.* KDD*. DOI: 10.1145/2783258.2783311
- Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M (2020). "Inverting gradients — how easy is it to break privacy in federated learning?" *NeurIPS.* NeurIPS*.
- Hardt, M., Price, E., and Srebro, N (2016). "Equality of opportunity in supervised learning." *NeurIPS.* NeurIPS*.
- Hinton, G., Vinyals, O., and Dean, J (2015). Distilling the knowledge in a neural network. *NeurIPS Workshop on Deep Learning.*.
- Ilharco, G., Ribeiro, M.T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A (2023). "Editing models with task arithmetic." *ICLR.* ICLR*.
- Kornblith, S., Norouzi, M., Lee, H., and Hinton, G (2019). "Similarity of neural network representations revisited." *ICML.* ICML*.
- Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J (2016). "Convergent learning: Do different neural networks learn the same representations?" *ICLR.* ICLR*.
- Lundberg, S.M. and Lee, S.-I (2017). "A unified approach to interpreting model predictions." *NeurIPS.* NeurIPS*.
- Madras, D., Creager, E., Pitassi, T., and Zemel, R (2018). "Learning adversarially fair and transferable representations." *ICML.* ICML*.
- McMahan, H.B., Moore, E., Ramage, D., Hampson, S., and Arcas, B.A (2017). "Communication-efficient learning of deep networks from decentralized data." *AISTATS.* AISTATS*.

- Mitchell, M., et al (2019). "Model cards for model reporting." *FAccT.* FAccT*. DOI: 10.1145/3287560.3287596
- Nagy, T. (2026). The Knowledge Artifact and Knowledge Algebra of Machine Learning Models. *Zenodo.* DOI: 10.5281/zenodo.18910387
- Nagy, T. (2026). The Spectral Tensor Representation of Stochastic Processes. *Working paper.*
- Ortiz-Jimenez, G., Favero, A., and Frossard, P (2023). "Task arithmetic in the tangent space: Improved editing of pre-trained models." *NeurIPS* (Oral). *NeurIPS.* DOI: 10.52202/075280-2913
- Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., Namkoong, H., Farhadi, A., Carber, Y., Kornblith, S., and Schmidt, L (2022). "Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time." *ICML.* ICML*.
- Zafar, M.B., Valera, I., Gomez Rodriguez, M., and Gummadi, K.P (2017). "Fairness beyond disparate treatment & disparate impact: Learning fair representations." *WWW.* WWW*.
- Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C (2013). "Learning fair representations." *ICML.* ICML*.
- Zhu, L., Liu, Z., and Han, S (2019). "Deep leakage from gradients." *NeurIPS.* NeurIPS*. DOI: 10.1007/978-3-030-63076-8_2