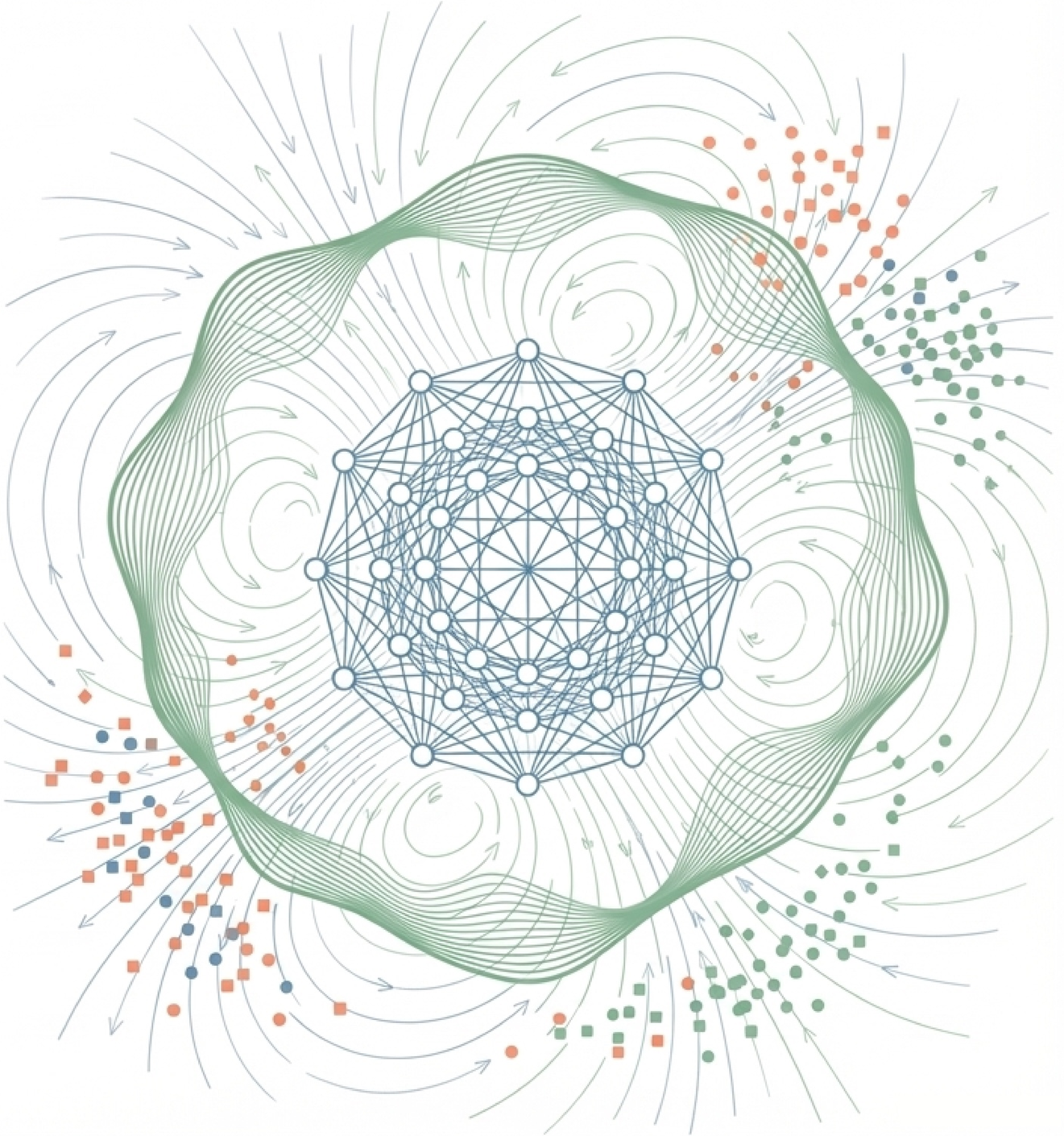
ScAdver: Adversarial Batch Correction for Single-Cell Data

Eliminate technical artifacts while preserving biological identity.

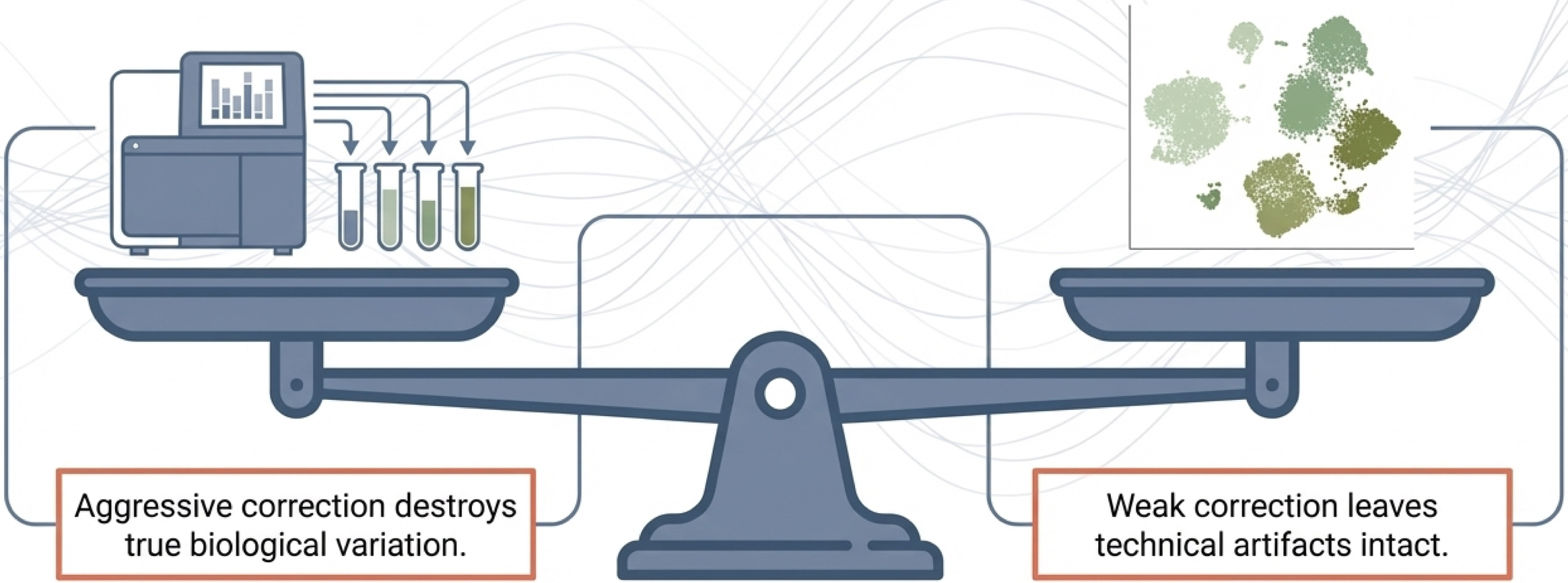


Created by Shivaprasad Patil | 2025

The persistent tension between technical noise and biological truth

Erase Technical Batch Effects

Preserve Cell Type Identity

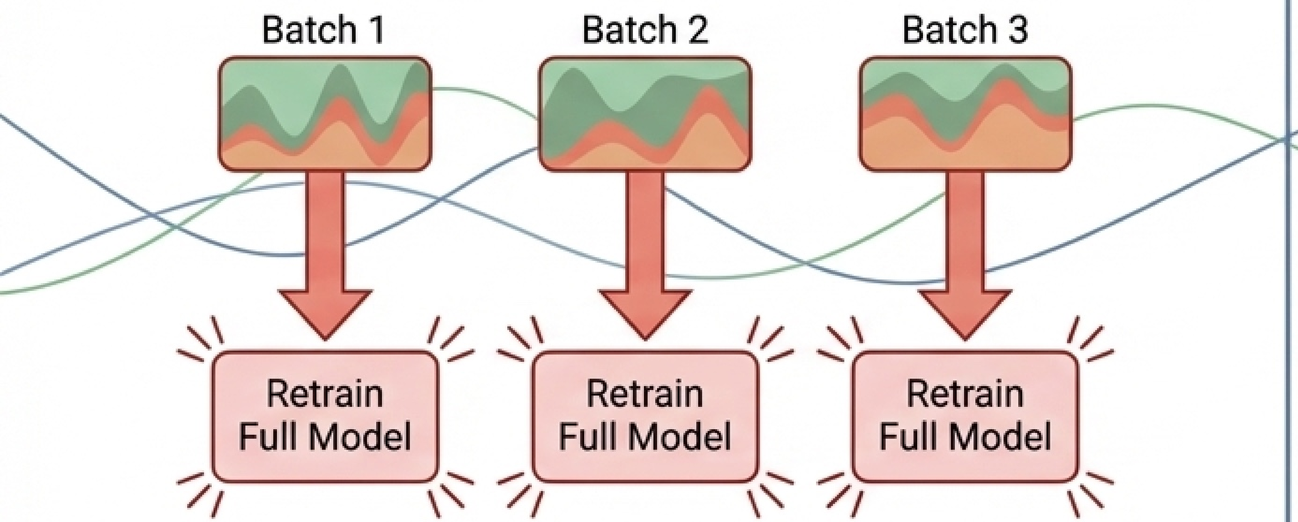


Aggressive correction destroys true biological variation.

Weak correction leaves technical artifacts intact.

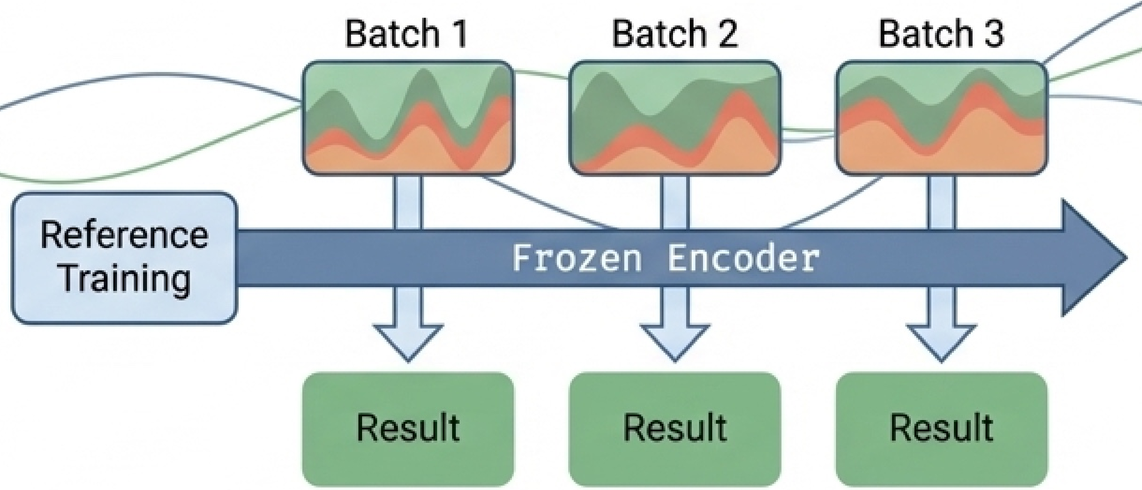
Escaping the retraining loop with a train-once, project-forever architecture

The Traditional Friction



Traditional batch correction requires retraining the entire **model whenever new data arrives.** This is computationally expensive and disrupts established biological embeddings.

The ScAdver Advantage



ScAdver uses adversarial learning to create a **frozen encoder** that **automatically strips batch effects** from new data.

Fast inference. Biology preserved. No retraining required.

Core capabilities driving the ScAdver engine

Biology Preserved

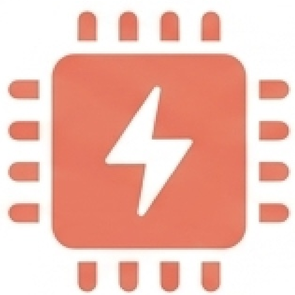
Adversarial discriminator removes batch effects without touching biological signals.

Fully Reproducible

`set_global_seed()` explicitly seeds every random operation for deterministic runs.

Distribution Alignment

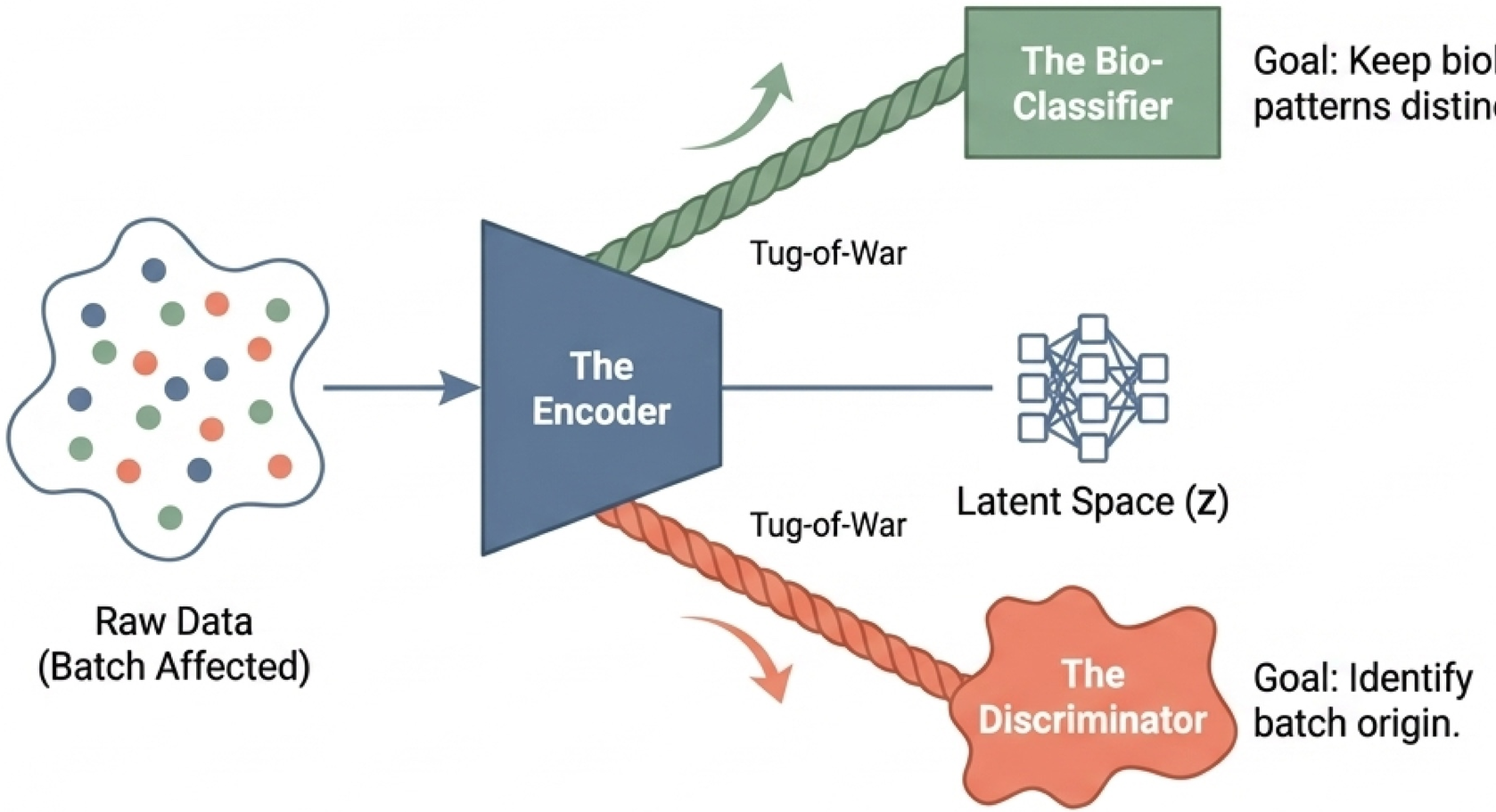
Uses MMD + Moment-Matching + CORAL losses for robust domain adaptation.



Hardware Optimized

Native multi-device support across CPU, CUDA, and Apple Silicon (MPS).

The mechanism: A zero-sum game between encoder and discriminator



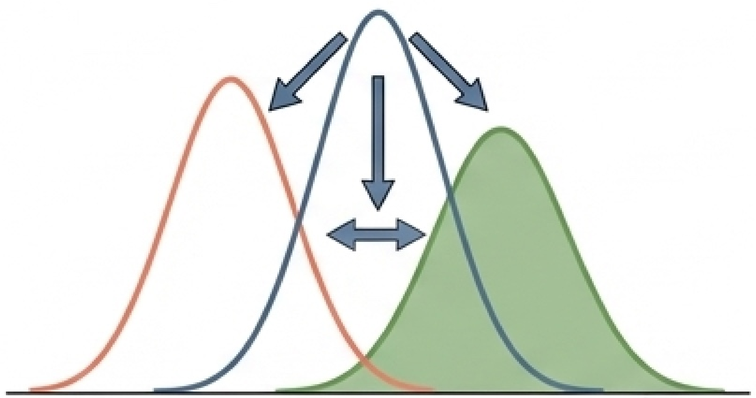
The system plays a **zero-sum game.**

The Encoder learns to produce embeddings where biological signals are strong, but batch signals are so weak the Discriminator cannot tell them apart.

Result: A frozen encoder that effectively ‘unlearns’ batch signals.

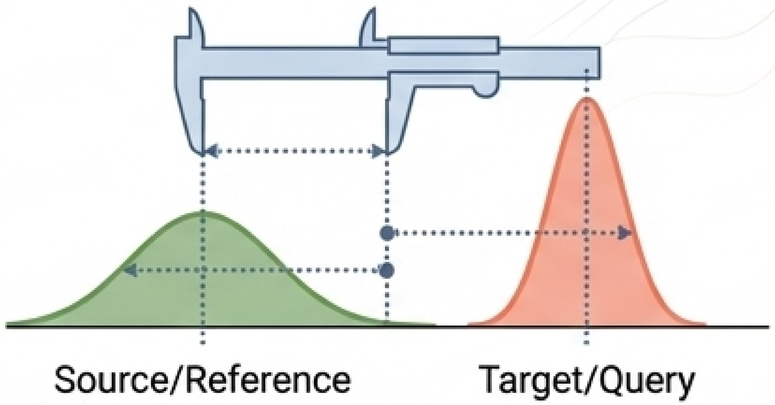
Enforcing mathematical alignment across distributions

MMD
(Maximum Mean Discrepancy)



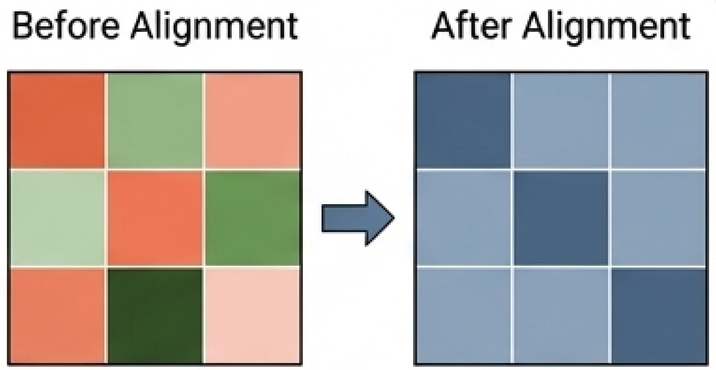
$$\mathrm{MMD}(P_x, P_y) \to 0$$

Moment-Matching



$$\mathbb{E}[X] = \mathbb{E}[Y]$$
$$\mathrm{Var}(X) = \mathrm{Var}(Y)$$

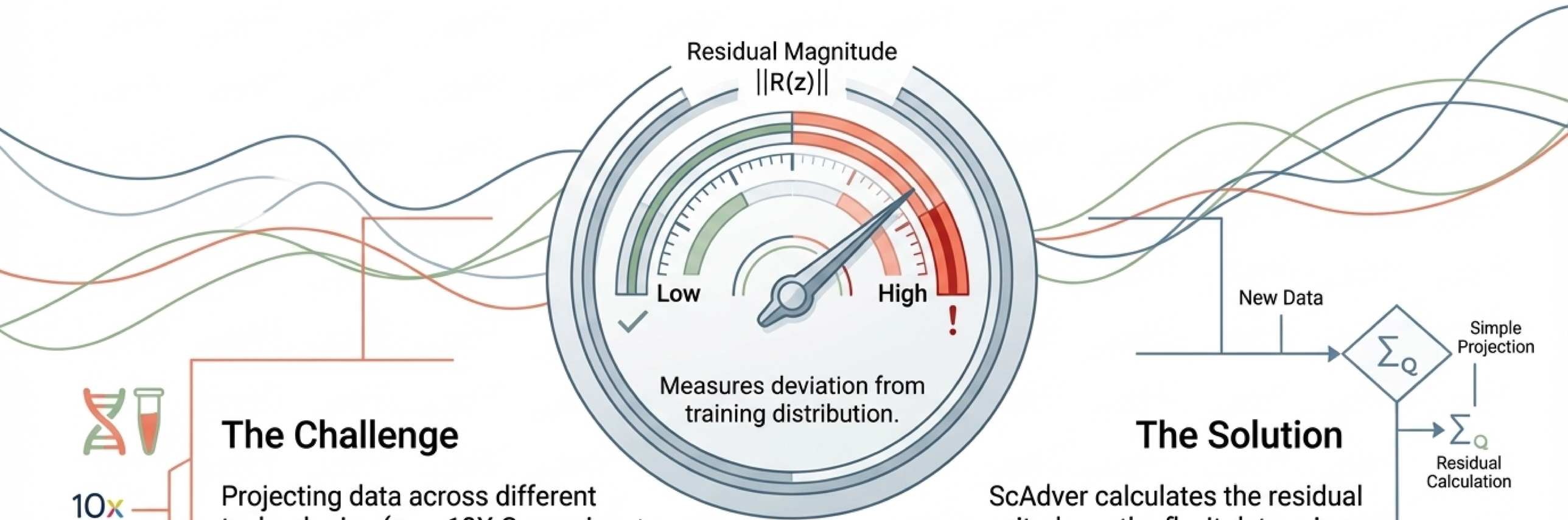
CORAL Loss



$$\|C_S - C_T\|_F^2 \to 0$$

ScAdver doesn't just rely on adversarial feedback. It enforces strict **statistical alignment** through an **Enhanced Residual Adapter** (3-layer, LayerNorm, GELU) featuring unbounded outputs with a learnable scale for ≤100 classes.

Intelligent domain shift detection for unpredictable query batches



The Challenge

10X
10X

Smart-seq2
Smart-seq2

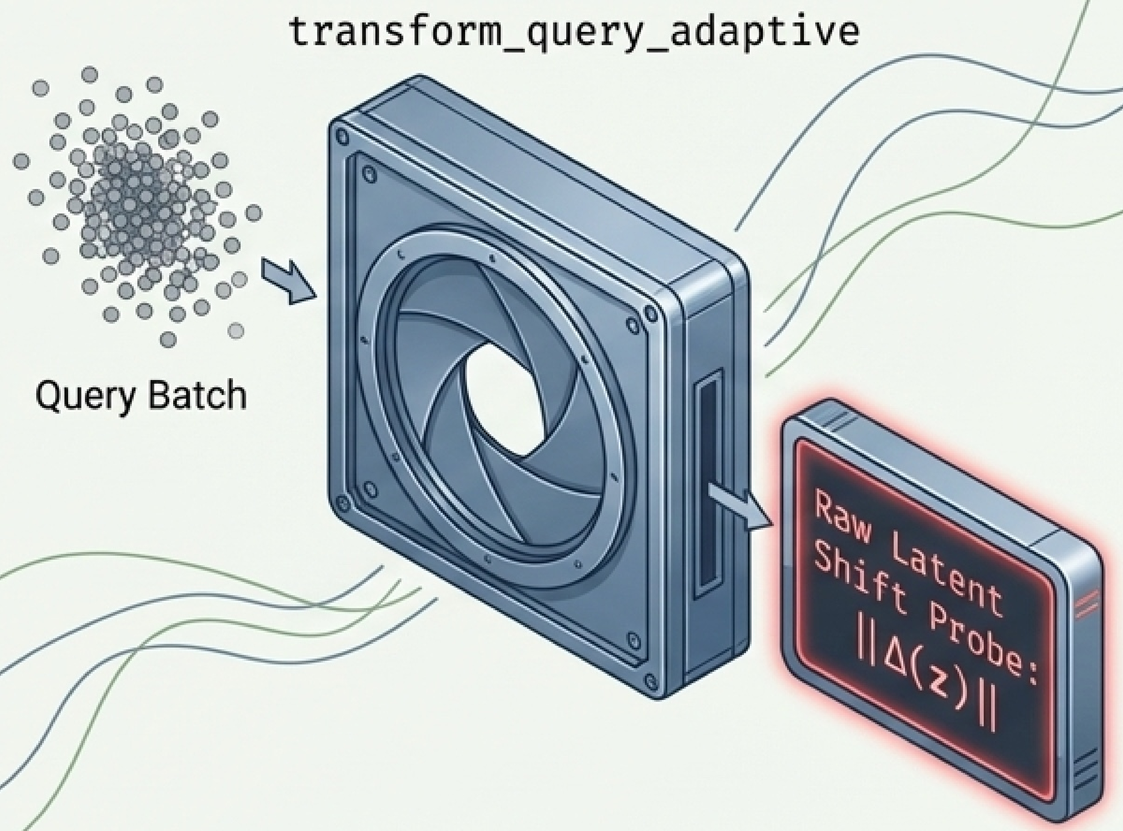
Projecting data across different technologies (e.g., 10X Genomics → Smart-seq2) introduces massive domain shifts that simple projection cannot handle.

The Solution

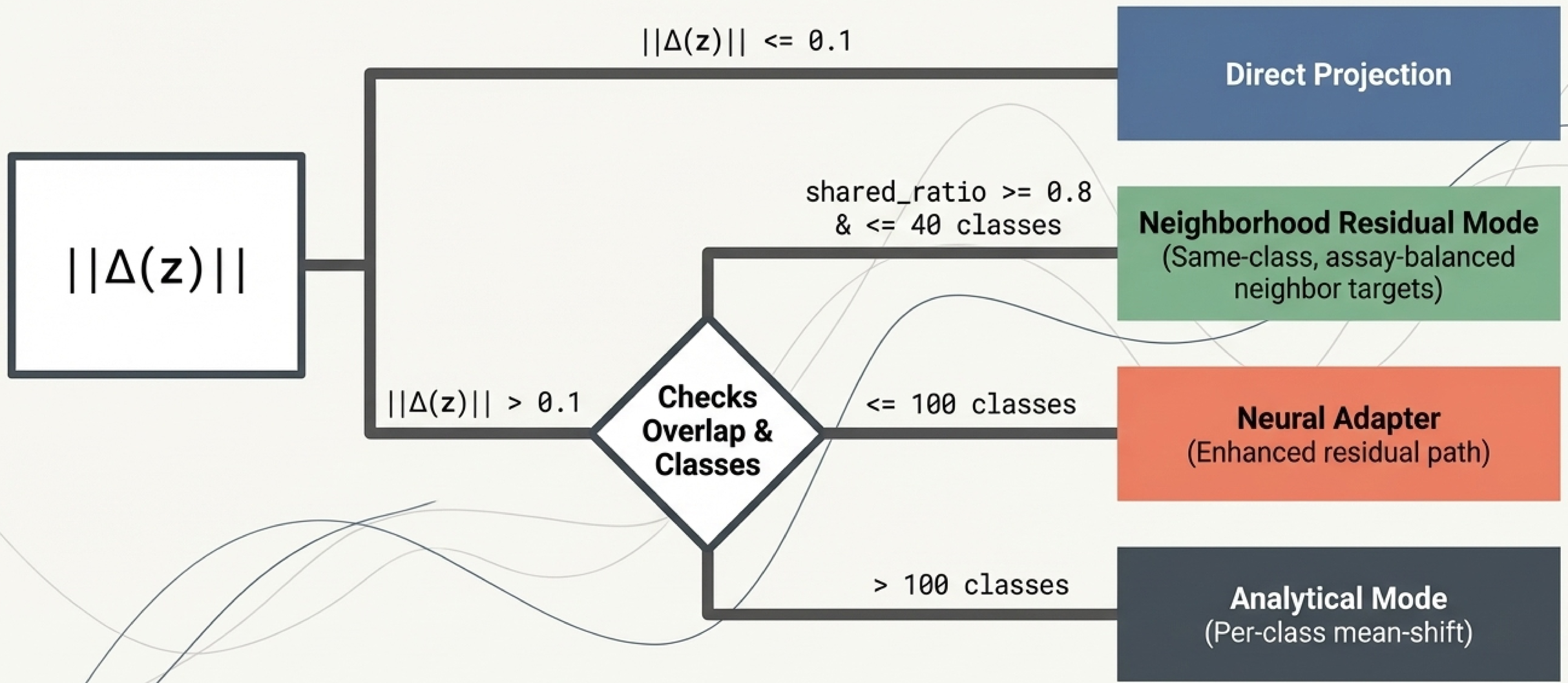
ScAdver calculates the residual magnitude on the fly. It determines whether the new data fits the model (Simple Projection) or requires mathematical adaptation (Residual Adapter).

Probe-gated query projection evaluates data upon arrival

- Not all query batches require the same level of correction.
- ScAdver automatically probes the raw latent shift and routes the data through the optimal projection path based on domain shift severity, class count, and label overlap.



Adaptive routing dynamically selects the optimal projection path



The Projection Path Matrix: Tailoring correction to the data

Path Name	Trigger Condition	Underlying Mechanism	Scale Capability
Direct Projection	`||Δ(z)|| <= 0.1`	Bypass adaptation entirely; data aligns naturally.	Infinite
Neighborhood Residual	High overlap, ≤ 40 classes	Deterministic residual step using same-class, assay-balanced targets.	Small/Targeted
Neural Adapter	Moderate overlap, ≤ 100 classes	`EnhancedResidualAdapter` utilizing adversarial + alignment losses, warmup, and early stopping.	Medium
Analytical Mean-Shift	> 100 reference classes	Fast, large-scale per-class mean-shift. Corrects 100k+ cells in seconds.	Massive Scale

Workflow 1: All-in-One Batch Correction

Best for one-time analysis when **all data is available upfront**.

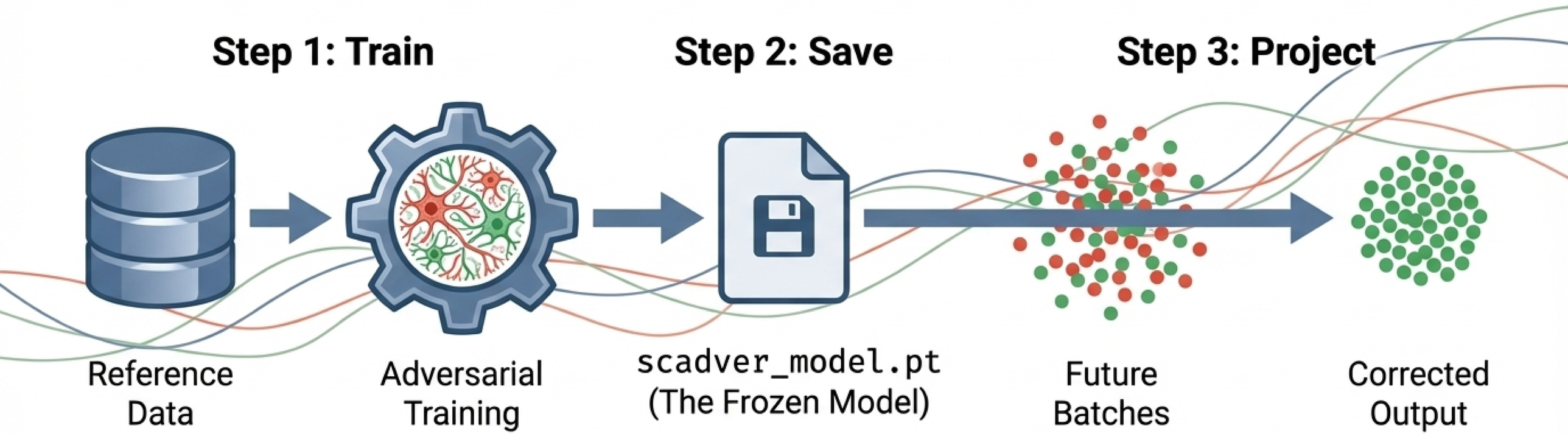
Single Function Call: Process all data, train the adversary, and return corrected embeddings in one step.

```python
import scanpy as sc
from scadver import adversarial_batch_correction

# Load and correct data
adata = sc.read('your_data.h5ad')
adata_corrected, model, metrics = adversarial_batch_correction(
    adata=adata,
    bio_label='celltype',
    batch_label='batch',
    epochs=500
)
```

Workflow 2: Train-Then-Project Architecture

The architecture for reusable models and scalable analysis.

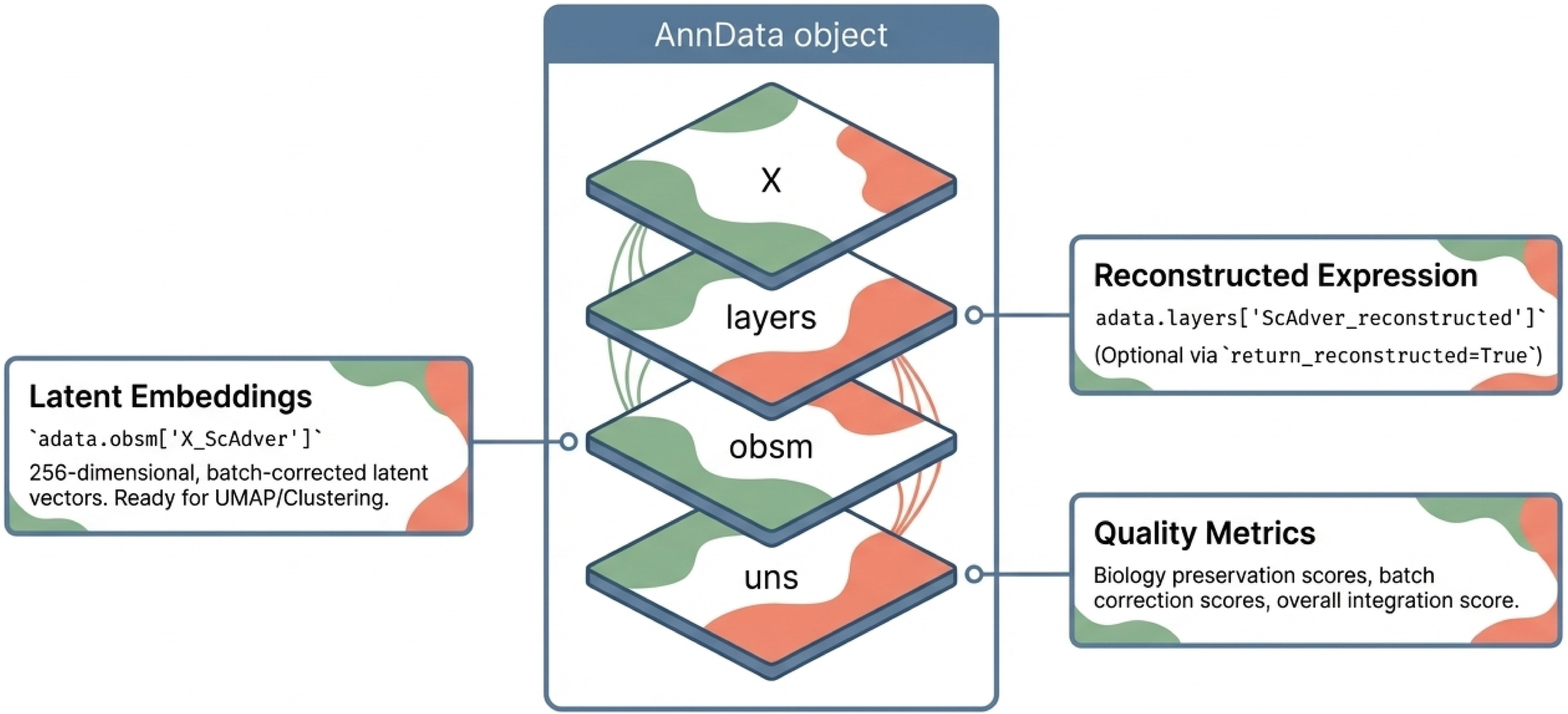


Use Cases: Deploying as a cloud service, processing patient data as it arrives, or handling massive datasets in chunks.

Decision Matrix: Selecting your implementation strategy

User Scenario	Recommended Workflow
All data available now, one-time analysis	**Workflow 1** (All-in-One)
Interactive analysis, have all data	**Workflow 1** (All-in-One)
Query batches arrive over time	**Workflow 2** (Train-Then-Project)
Deploying model as a cloud service	**Workflow 2** (Train-Then-Project)
Massive protocol shifts (e.g., 10X → Smart-seq2)	**Workflow 2** (Auto-Adaptive)

The anatomy of ScAdver data artifacts



ScAdver Ecosystem: Resources & Citation

Documentation

`ENCODER_MECHANISM_EXPLAINED.md`
(Deep dive on training dynamics)

`RESIDUAL_ADAPTER.md`
(Mathematics of domain adaptation)

License: Apache 2.0 (Open Source)

Citation

```bibtex
@software{scadver2025,
  title = {ScAdver: Adversarial Batch Correction for Single-Cell},
  author = {Shivaprasad Patil},
  year = {2025},
  url = {https://github.com/shivaprasad-patil/ScAdver}
}
```

Contributions welcome via Pull Request.