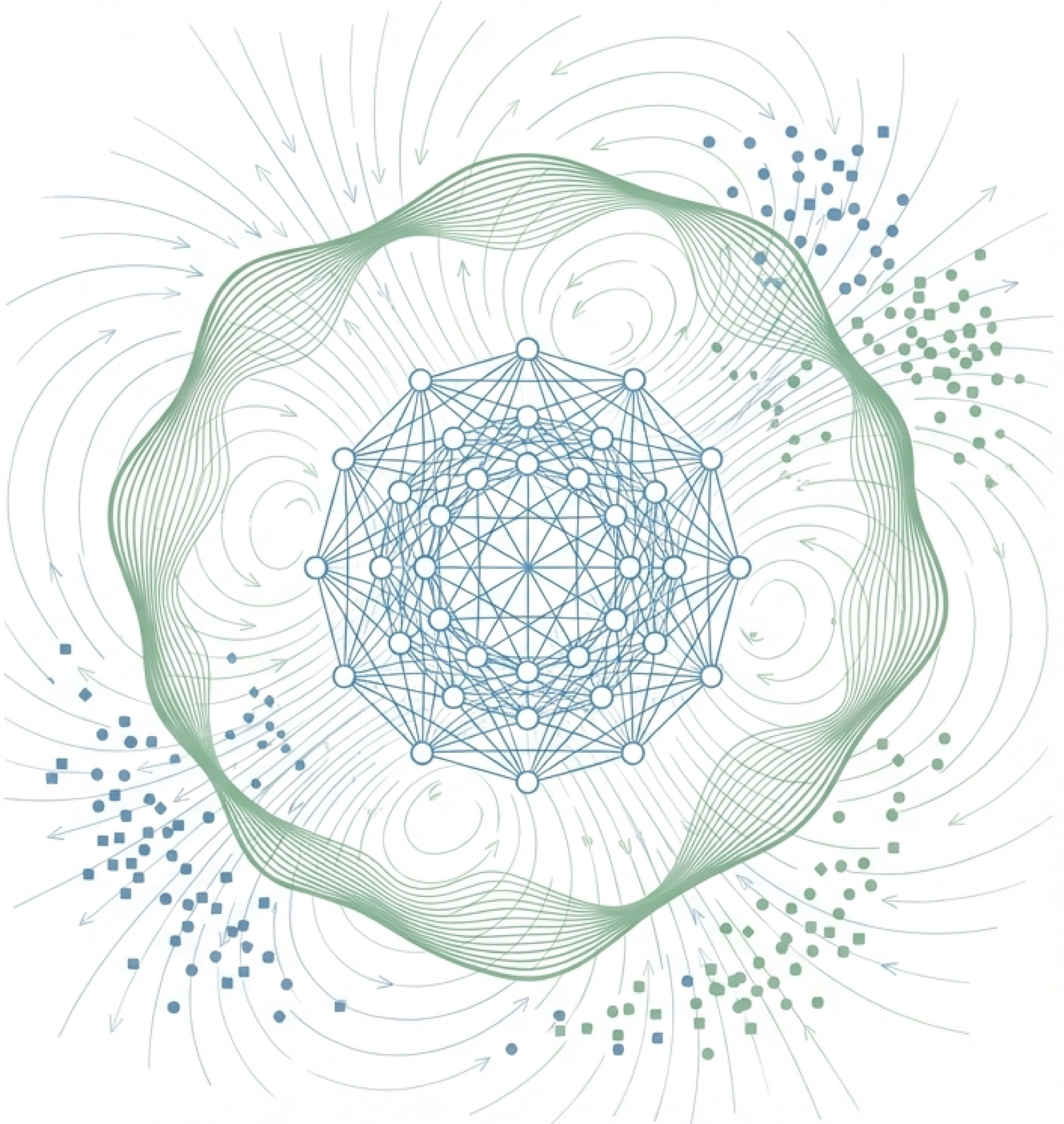
BIOINFORMATICS TOOLKIT

ScAdver: Adversarial Batch Correction for Single-Cell Data

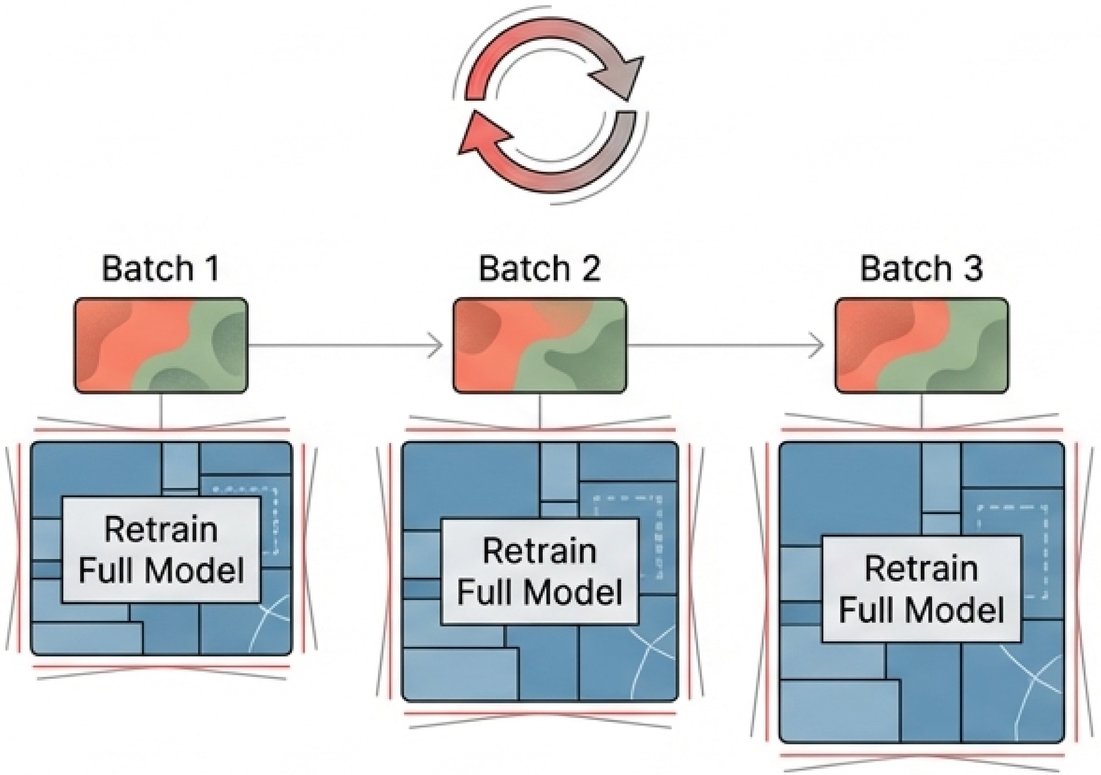
Eliminate technical artifacts while preserving biological identity.



Created by Shivaprasad Patil | 2025

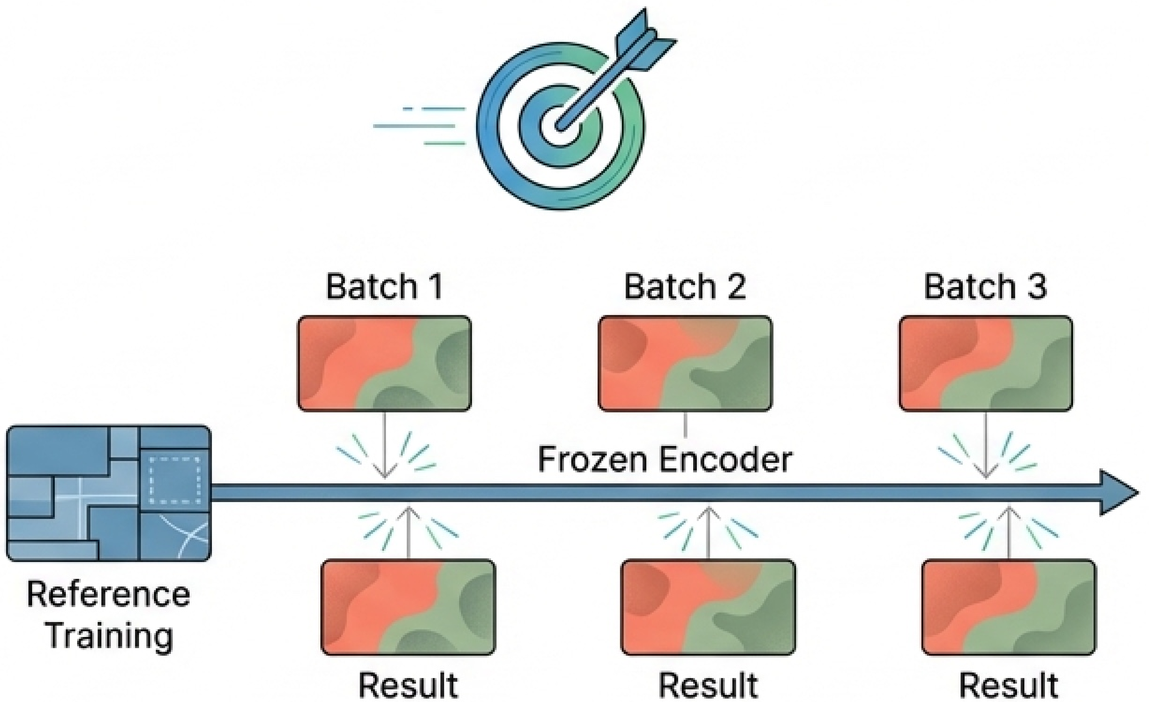
The Train-Once, Project-Forever Paradigm

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 & Feature Set

Train Once, Project Forever

Save trained models and process unlimited query batches.

Fast Inference

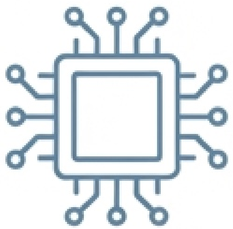
Real-time processing with no retraining loop.

Biology Preserved

Cell types and biological variation are strictly maintained.

Batch-Free

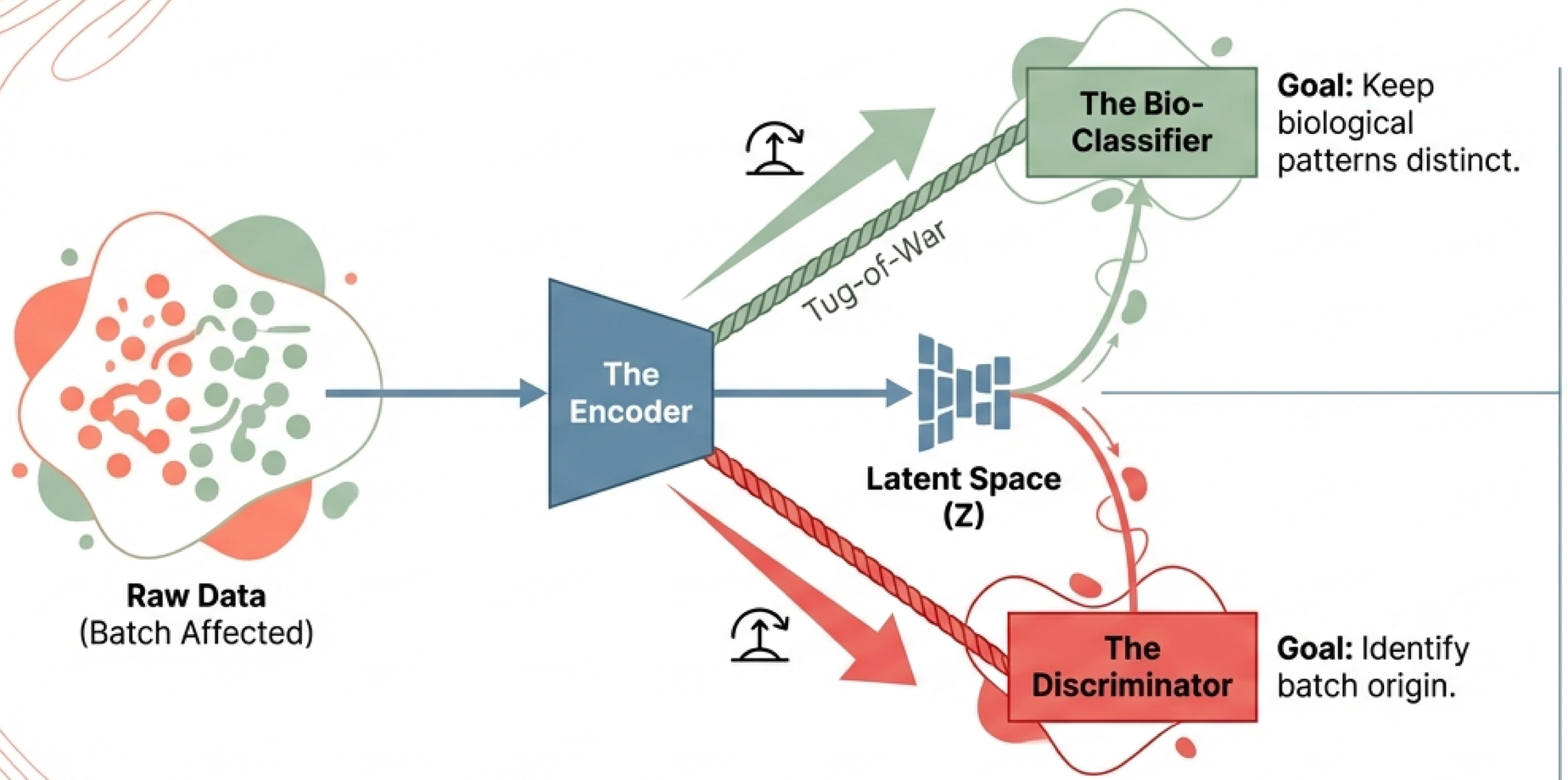
Technical variation and protocol effects are adversarially removed.



Multi-Device Architecture

Native support for CPU, NVIDIA CUDA, and Apple Silicon (MPS).

The Mechanism: Encoder vs. 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.

Choosing the Optimal Workflow

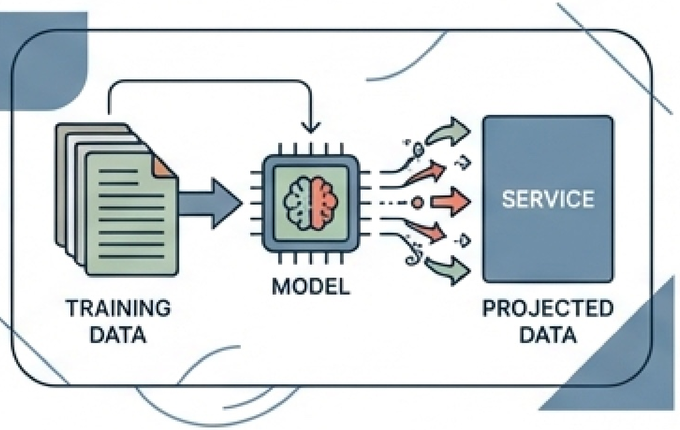
Data Availability

Workflow 1: All-in-One

Context: “I have all my data right now.”

Best For: One-time analysis, static datasets

Workflow 2: Train-Then-Project



Context: “My data arrives in streams” or “I am building a service.”

Best For: Reusable models, longitudinal studies, cross-protocol integration

Workflow 1: All-in-One Batch Correction

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

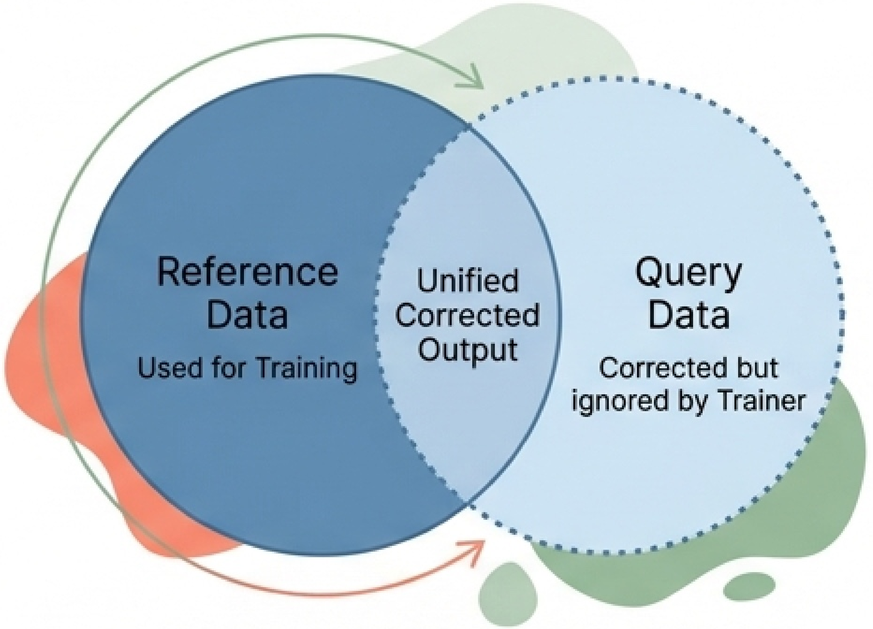
Single Function Call: Process all data, train the adversary, and return corrected embeddings in one step.

```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
)
```

Computational Biology Editorial

Advanced Config: Reference-Query Split

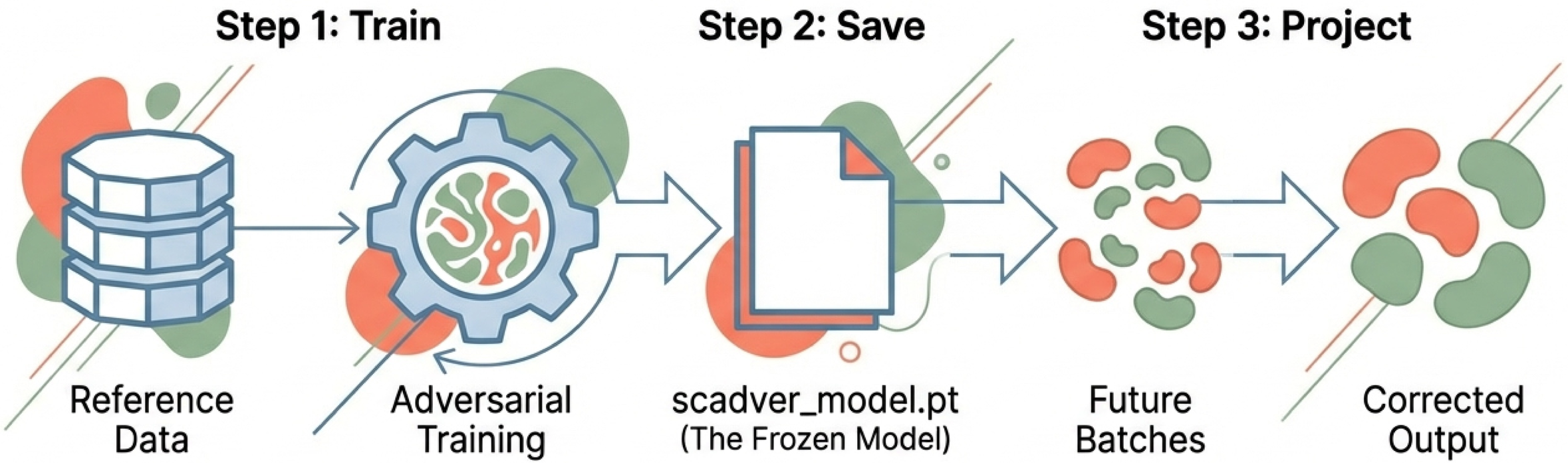
Train *only* on trusted reference samples, but correct *both* reference and query data.




```
# Mark samples as Reference vs Query
query_mask = np.array([tech in ["smartseq2", "celseq2"] for tech in adata.obs["tech"]])

# Train on Reference only
adata_corrected, model, metrics = adversarial_batch_correction(
    adata=adata,
    bio_label='celltype',
    batch_label='tech',
    reference_data='Reference', # Active Training
    query_data='Query'          # Passive Correction
)
```

Workflow 2: Train-Then-Project

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.

Executing the Reusable Model

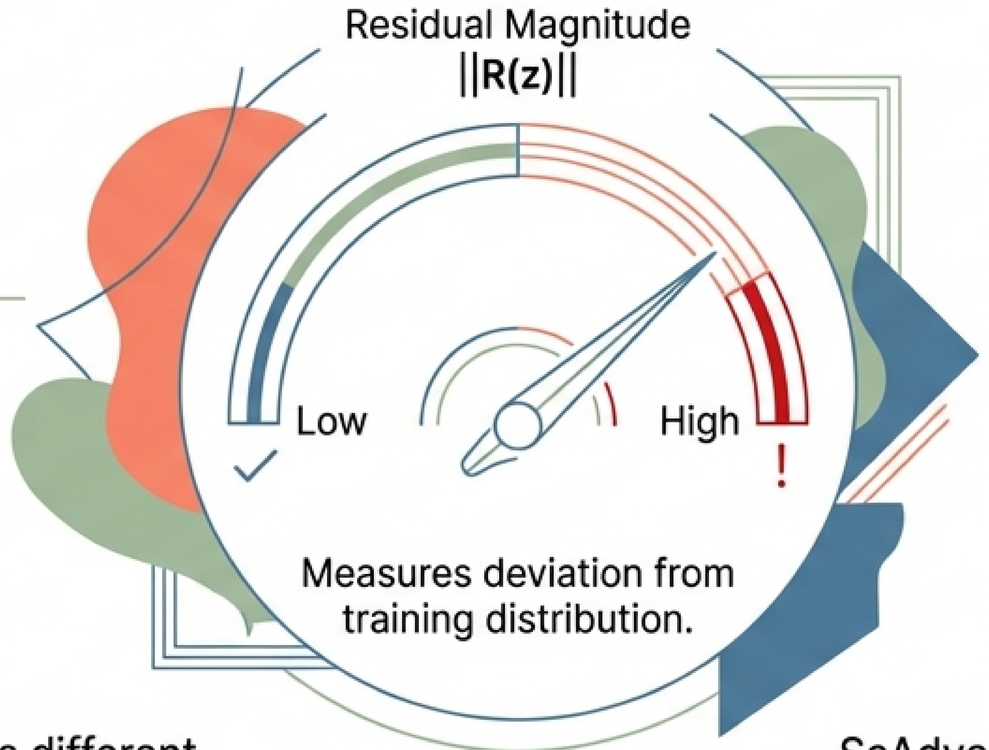
```
import torch
from scadver import adversarial_batch_correction,
    transform_query_adaptive

# Save model for reuse
torch.save(model.state_dict(),
'scadver_model.pt')

# Project query data (fully automatic)
adata_query = transform_query_adaptive(
    model=model,
    adata_query=query_data,
    adata_reference=adata_reference[:500],
    bio_label='celltype'
)
```

Fully Automatic Projection

- Save the trained weights with `torch.save`.
- Load and apply to any new data using `transform_query_adaptive`.

This function handles domain adaptation **automatically** without user intervention.

Intelligent Automatic Domain Shift Detection



The Challenge

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

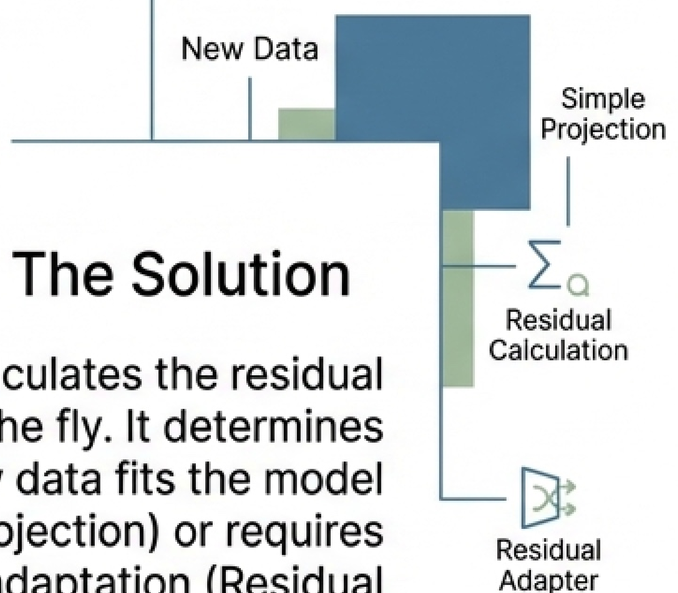
10X

10X

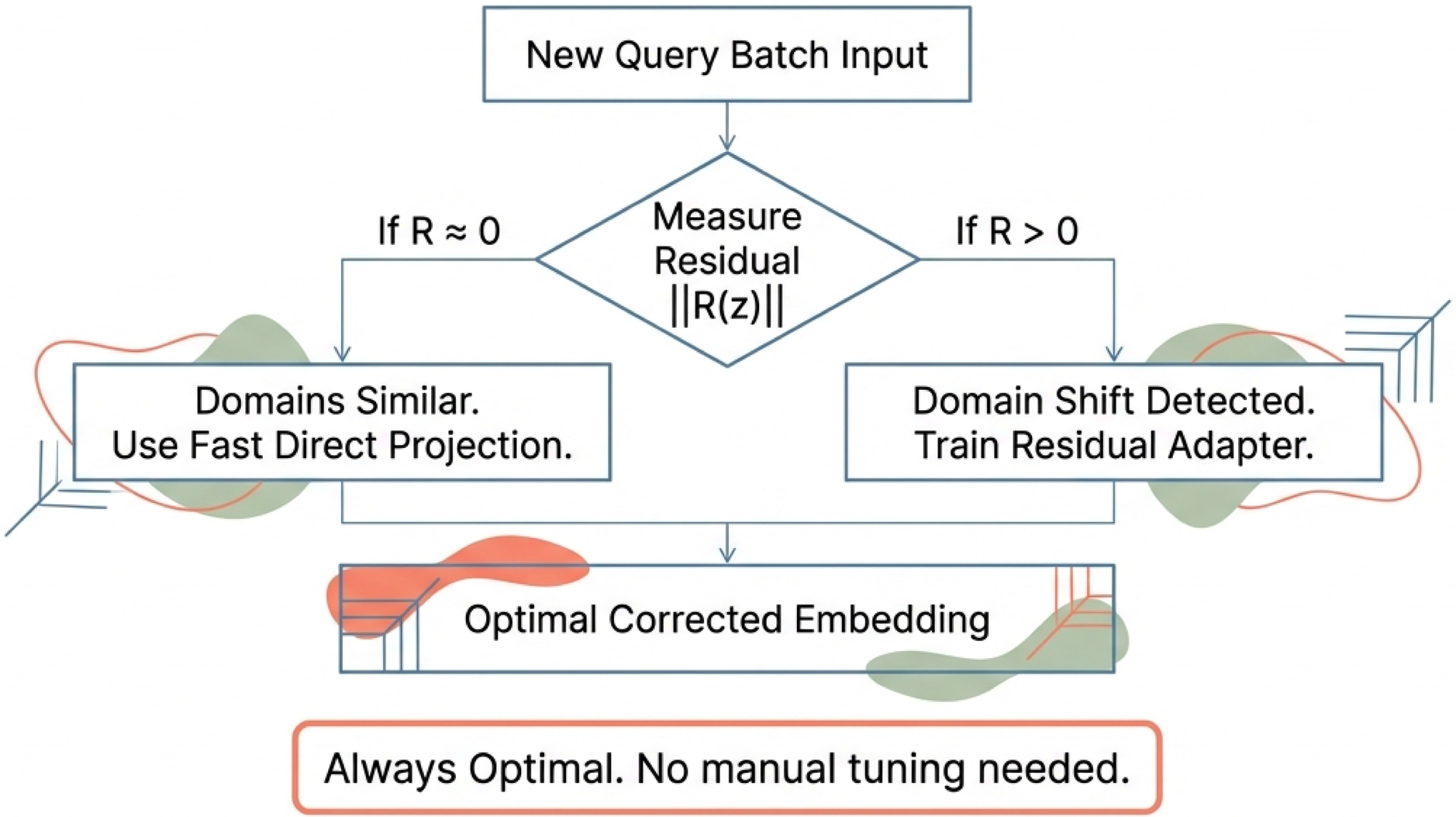
Smart-seq2

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).



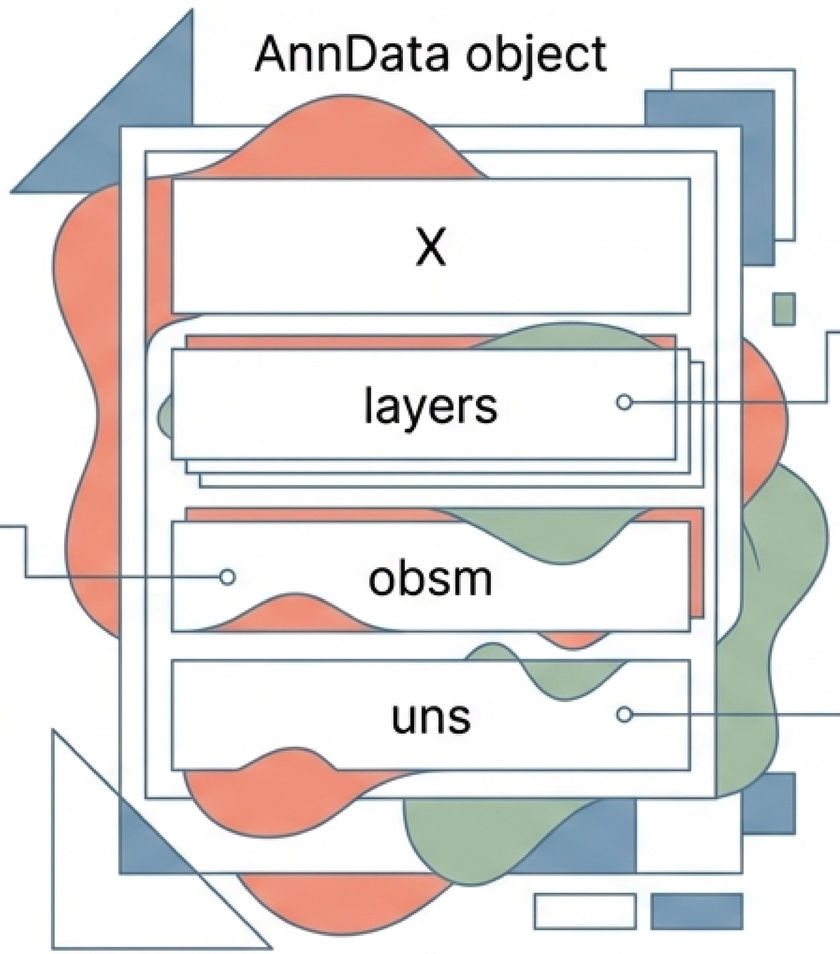
The Decision Logic Allgorithm



Decision Matrix: Select Your Approach

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 service	**Workflow 2** (Train-Then-Project)
Massive protocol shifts (10X → Smart-seq2)	**Workflow 2** (Auto-Adaptive)

Data Artifacts & Outputs



Latent Embeddings

`adata.obsm['X_ScAdver']`
256-dimensional, batch-corrected vectors. Ready for UMAP/Clustering.

Reconstructed Expression

`adata.layers['ScAdver_reconstructed']`
(Optional via `return_reconstructed=True`).

Quality Metrics

Biology preservation scores, batch correction scores, overall integration score.

Specifications & Installation

Installation

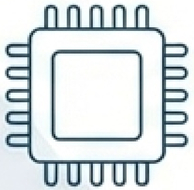
```
pip install
git+https://github.com/shivaprasad-
patil/ScAdver.git
```

Dependencies

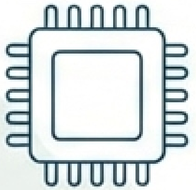
Requires Scanpy and PyTorch.

Hardware Acceleration

Apple Silicon
Native Metal (MPS) support (M1/M2/M3).



NVIDIA CUDA
High-performance training support.



Standard CPU
Universal fallback support.

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

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

Contributions welcome via Pull Request.