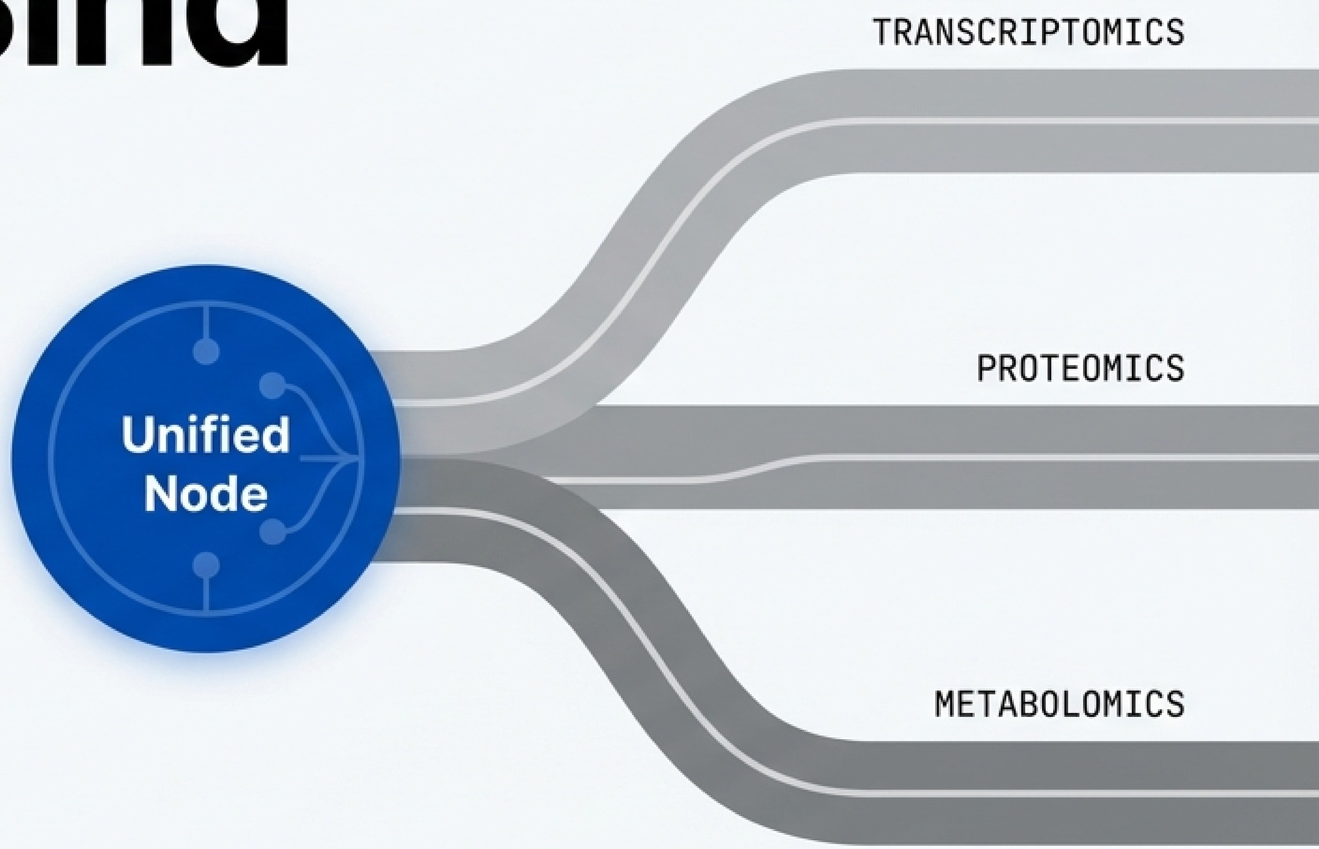
MultiOmicsBind

Bringing ImageBind's Modality to Biology

A Deep Learning Framework for Efficient Multi-Omics Integration.



github.com/shivaprasad-patil/MultiOmicsBind

The Fragmentation of Biological Data

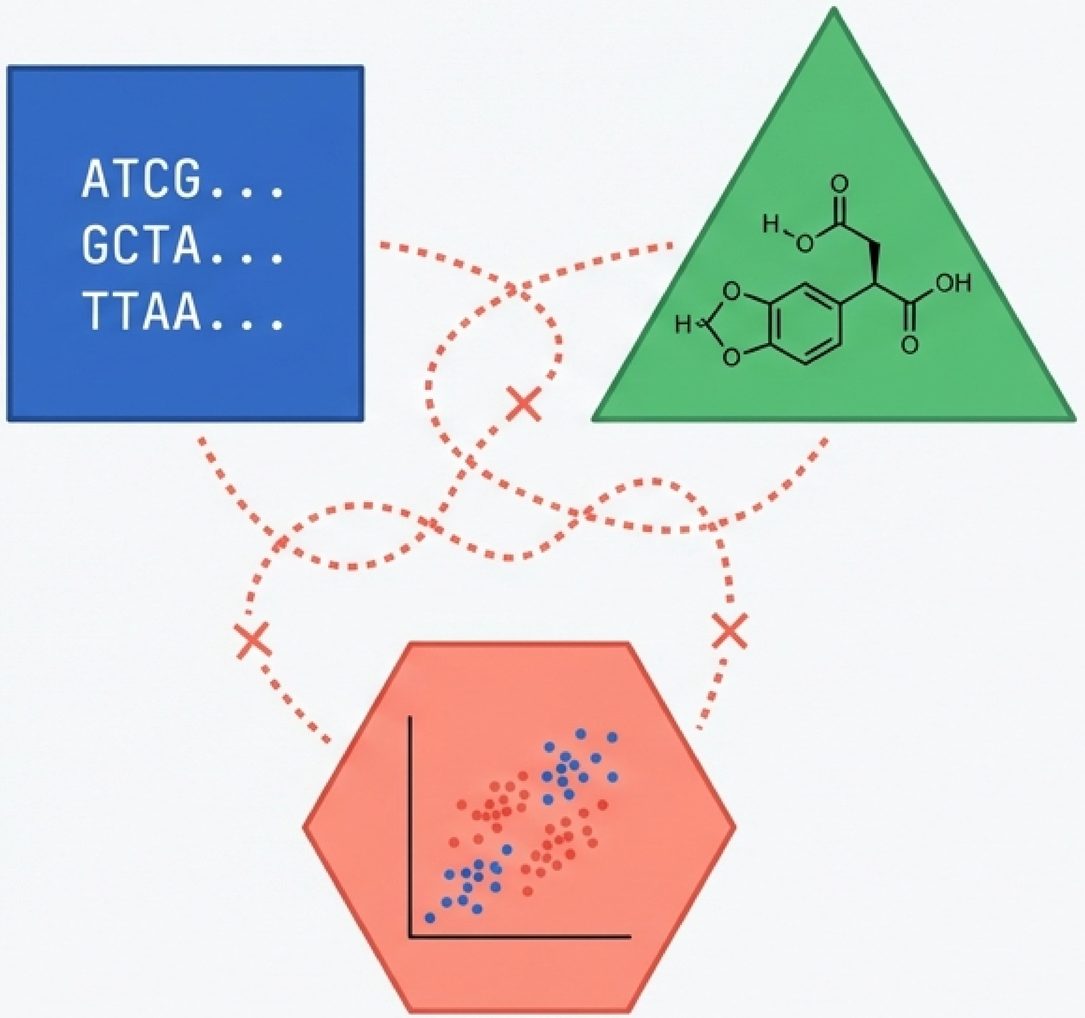
Modern computational biology relies on multi-modal data sources: Transcriptomics, Proteomics, Metabolomics, and Clinical Metadata.

The Problem:

Traditional integration methods are computationally **expensive**. They often require complex pairwise matching that scales poorly ($O(n^2)$ other Inter) as new modalities are added.

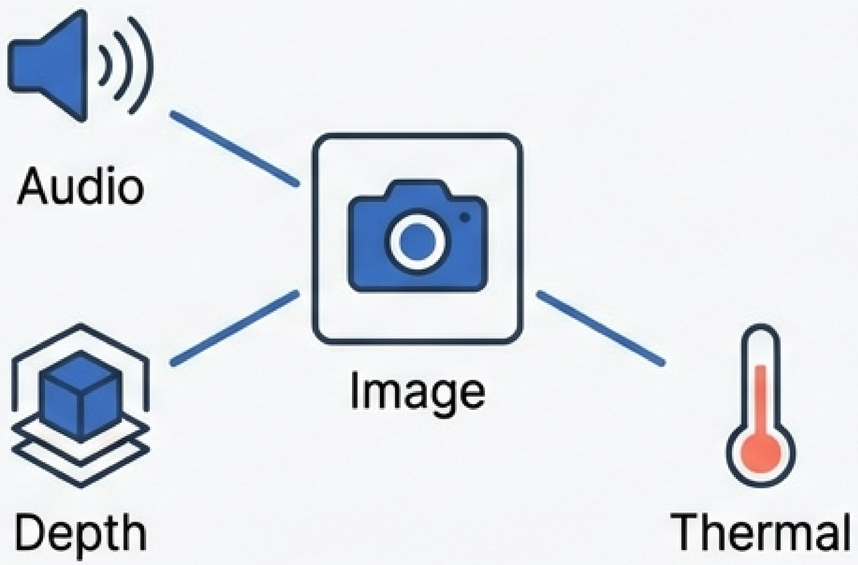
The Goal:

We need "Unified Embeddings"—a shared representation space where all biological layers speak the same mathematical language.



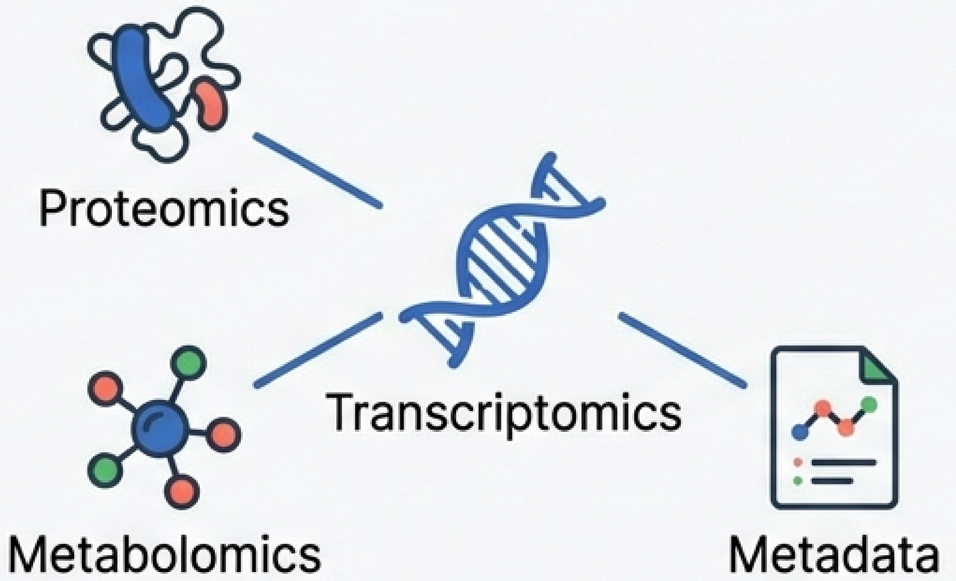
Leveraging the ImageBind Paradigm

INSPIRATION: COMPUTER VISION



ImageBind links modalities to a central image representation.

APPLICATION: MULTIOMICSBIND

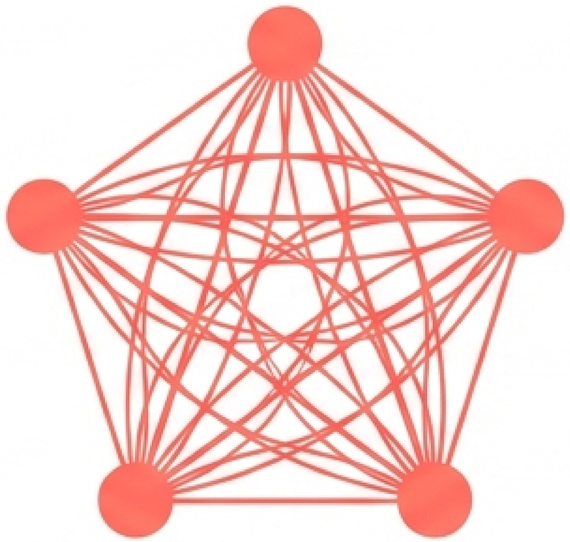


MultiOmicsBind uses a “Binding Modality” as the anchor for biological data.

Learns unified representations across different biological modalities using contrastive learning.

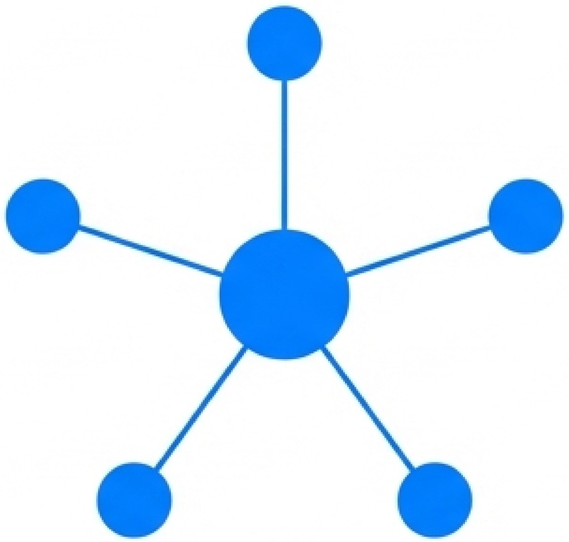
The Mathematics of Efficiency

Traditional All-Pairs Approach



Complexity: $O(n^2)$
Comparisons: $n \times (n-1)/2$
Speed: 1x (Baseline)

MultiOmicsBind Approach

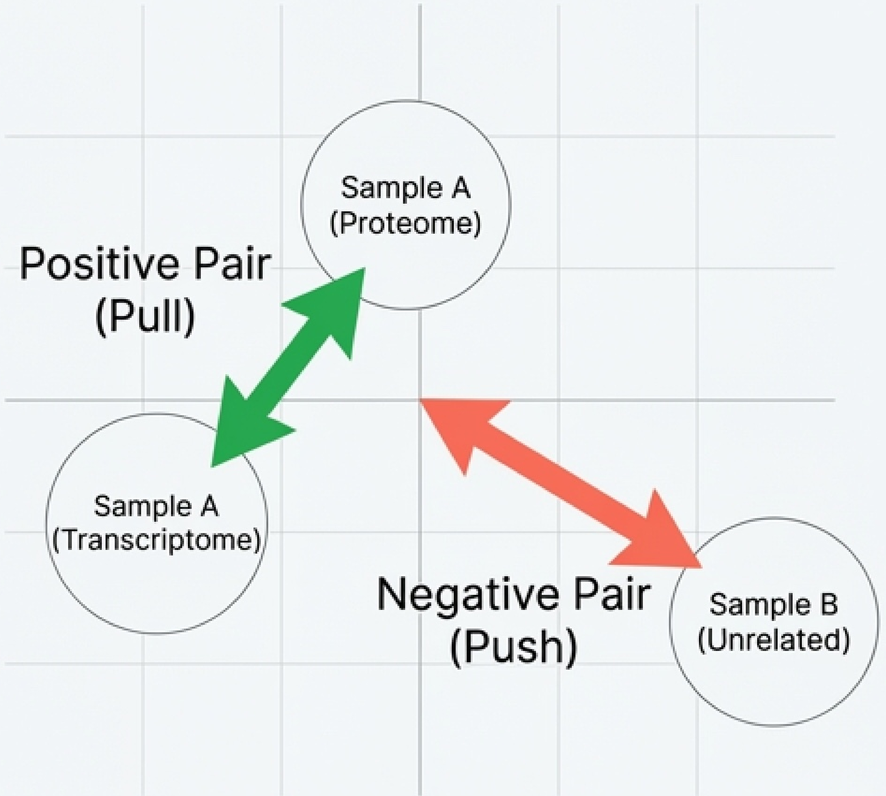


Complexity: $O(n)$
Comparisons: $n-1$
Speed: 5x+

By choosing a stable anchor (e.g., transcriptomics), computational complexity drops drastically while maintaining alignment.

Powered by Contrastive Learning

MultiOmicsBind employs self-supervised multi-modal alignment to generate high-quality embeddings without manual labeling.



Maximizes similarity between related samples while minimizing similarity between unrelated samples.

High-Level API: Single Function Training

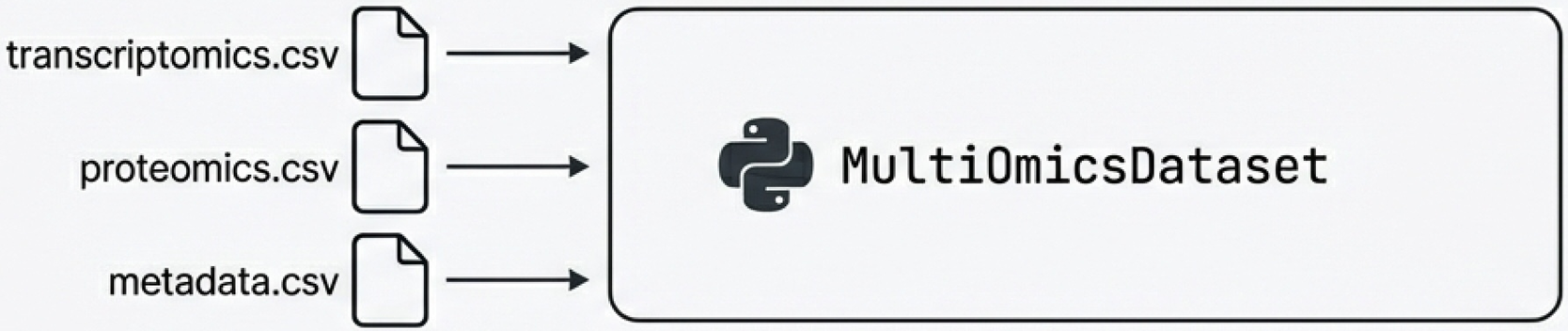
Designed for Developer Experience. The API allows researchers to bypass boilerplate code and focus on the experiment.

```python
import torch
from multiomicsbind import MultiOmicsDataset, train_multiomicsbind, set_seed

# Set seed for reproducibility
set_seed(42)

# Train model
model, history = train_multiomicsbind(
    dataset=dataset,
    device=device,
    epochs=20,
    batch_size=32,
    embed_dim=256
)
```

Abstracts the complex training loop.

Flexible Architecture & Data Loading

Supports any combination of omics types and metadata using a dictionary-based structure.



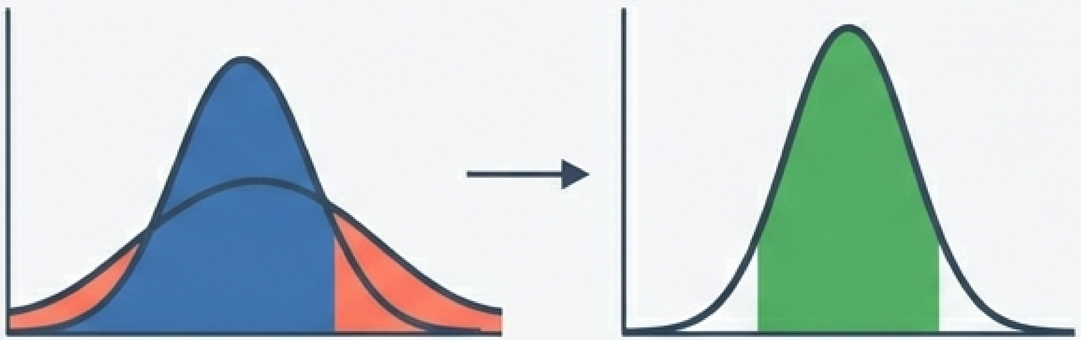

```python
dataset = MultiOmicsDataset(
    data_paths={
        'transcriptomics': 'transcriptomics.csv',
        'proteomics': 'proteomics.csv',
        'metabolomics': 'metabolomics.csv'
    },
    metadata_path='metadata.csv',
    label_col='response',
    binding_modality='transcriptomics' # Anchor modality
)
```

User defines the explicit Anchor modality.

Intelligent Data Handling

Automated Normalization and Type Detection

Automatic Normalization



Original Data

Z-score Standardized
(mean=0, std=1)

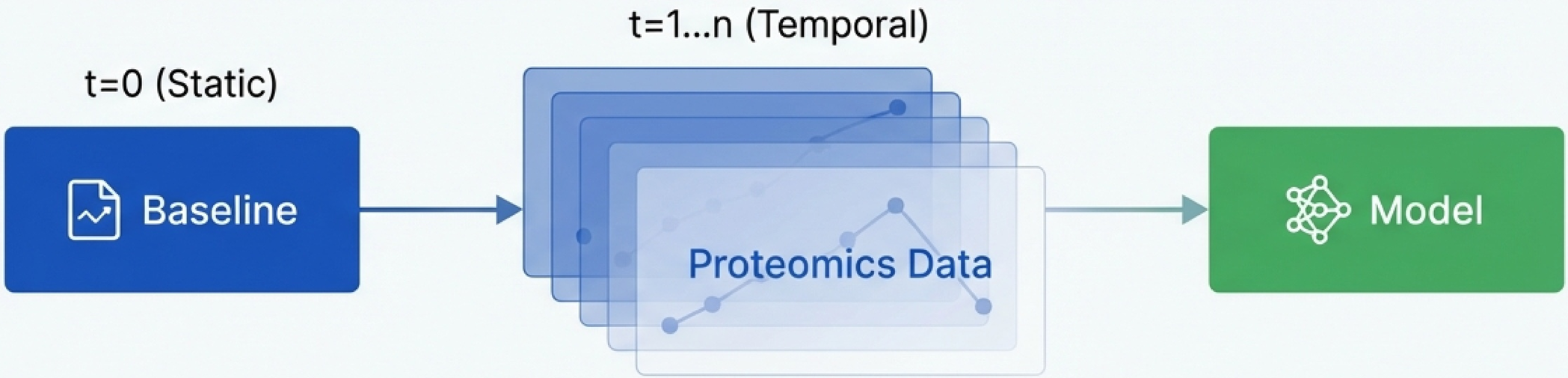
Default: `normalize=True`. Applies Z-score standardization automatically.

Smart Binary Detection

Detects binary data (e.g., mutations) and skips normalization to preserve discrete values.

Capturing Temporal Dynamics

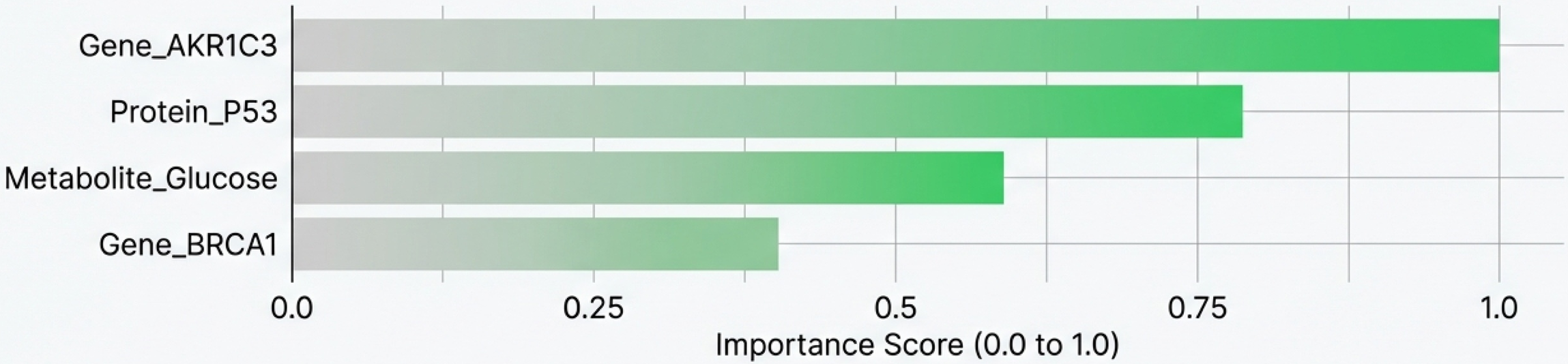
Biology happens over time. MultiOmicsBind supports time-series integration.




```
dataset = TemporalMultiOmicsDataset(
    static_data_paths={'transcriptomics': 'transcriptomics_t0.csv'},
    temporal_data_paths={'proteomics': 'proteomics_timeseries.csv'},
    ...
)
```

Feature Importance & Interpretability

Gradient-based attribution reveals which biological features drive the model's decisions.




```
importance_scores = compute_feature_importance(
    method='gradient'
)
```

Engineered for Reproducibility

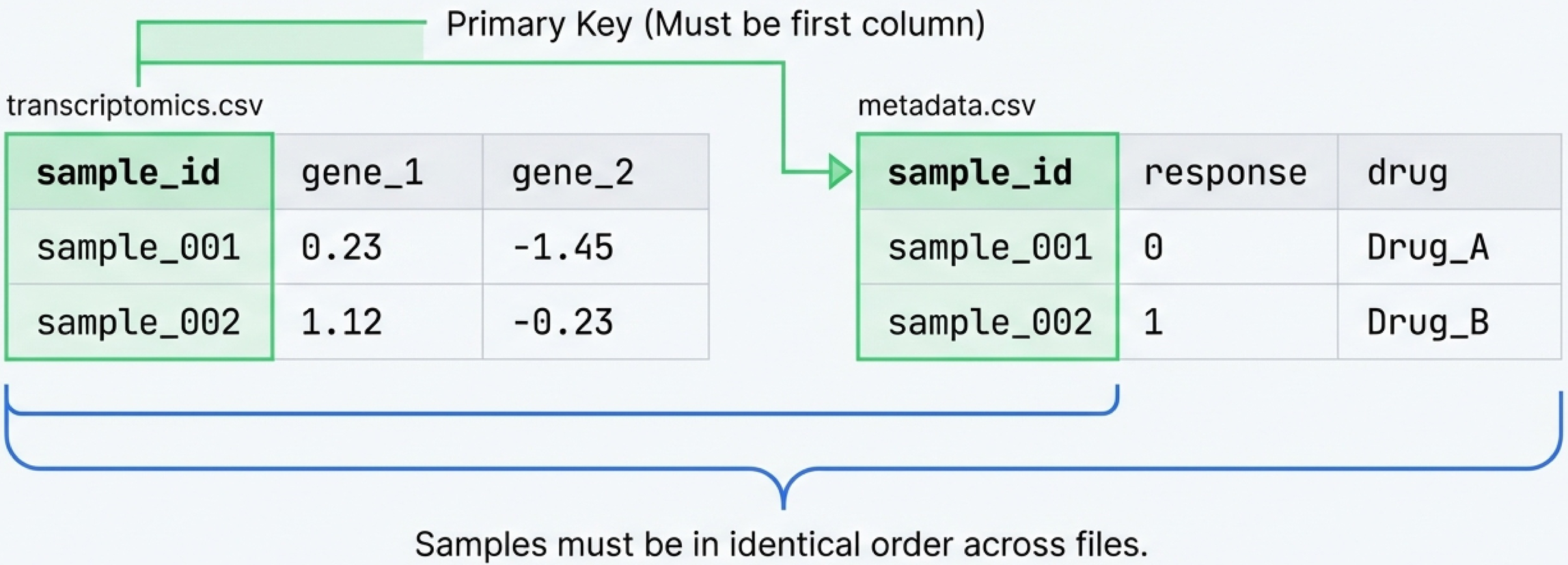
Consistent results across runs are enforced by default, solving a major crisis in AI biology.

- **Automatic Seeding**: Default `seed=42` ensures identical initialization.
- **Custom Control**: Users can pass specific seeds or `None` for random runs.

```
# Automatic seeding
model, history = train_multiomicsbind(..., seed=42)
```

Data Formatting Standards

Input requirements for seamless integration.



Installation & Requirements

- Python 3.8+
- PyTorch 1.9+
- NumPy, Pandas, scikit-learn
- CUDA (GPU) supported automatically

```
$ git clone
https://github.com/shivaprasad-patil/
MultiOmicsBind.git
$ cd MultiOmicsBind
$ pip install -e .
```

Comprehensive Examples

Explore the `examples/` directory to get started with specific use cases.

- `basic_example.py` - Standard multi-omics integration.
- `binding_modality_example.py` - Demonstration of **star-topology** concept.
- `temporal_example.py` - Handling time-series omics data.
- `flexible_modalities_example.py` - Variable modality configurations.
- `advanced_analysis.py` - Extracting embeddings and feature importance.

Ready for Research

```bibtex
@software{multiomicsbind2025,
  title = {MultiOmicsBind: Deep Learning Framework for Multi-Omics Integration},
  author = {Patil, Shivaprasad},
  year = {2025}
}
```


Star the repo and contribute:
github.com/shivaprasad-patil/MultiOmicsBind