SWIFT: Scalable Wasserstein Factorization for Sparse Nonnegative Tensors

Ardavan Afshar\textsuperscript{1}  Kejing Yin\textsuperscript{2}  Sherry Yan\textsuperscript{3}  Cheng Qian\textsuperscript{4}  Joyce C. Ho\textsuperscript{5}  
Haesun Park\textsuperscript{1}  Jimeng Sun\textsuperscript{6}

\textsuperscript{1}Georgia Institute of Technology  \textsuperscript{2}Hong Kong Baptist University  \textsuperscript{3}Sutter Health  \textsuperscript{4}IQVIA  
\textsuperscript{5}Emory University  \textsuperscript{6}University of Illinois at Urbana-Champaign
**Background: CP Tensor Factorization**

CP factorization\(^1\) approximates a tensor \( \mathcal{X} \) as the sum of \( R \) rank-one tensors:

\[
\mathcal{X} \approx \hat{\mathcal{X}} = [A^{(1)}, A^{(2)}, \ldots, A^{(N)}] = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \ldots \circ a_r^{(N)},
\]

- \( A^{(n)} \): Factor matrix for the \( n \)-th mode.
- \( a_r^{(n)} \): the \( r \)-th column of \( A^{(n)} \)

An example of CP factorization with input of a patient-diagnosis-medication tensor.

- It is widely-used in various applications, e.g. healthcare data analytics.
- It is highly interpretable: each rank-one tensor can be treated as a latent factor.

---

\(^1\) Tamara G Kolda and Brett W Bader. “Tensor decompositions and applications”. In: SIAM Review (2009).
Motivation

Existing tensor factorization models assume certain distributions of input, for example:

- **Gaussian distribution**: \( \min \hat{X} \| X - \hat{X} \|_F^2 \) ← **MSE loss**
- **Poisson distribution**: \( \min \hat{X} \hat{X} - X \log(\hat{X}) \) ← **KL divergence**
- **Bernoulli distribution**: \( \min \hat{X} \log(1 + e^{\hat{X}}) - X \hat{X} \) ← **logit loss**

Do we always know the distribution of a given input tensor?

- Real-world data often have very complex distributions.
- We usually do not know the underlying distribution of the input tensor.

---

Instead of assuming a specific distribution, the **Wasserstein distance** can be an alternative.

- *a.k.a.* Earth Mover Distance (EMD);
- is a potentially better measure of the difference between two distributions;
- does not assume any particular distributions of input data; and
- can leverage correlation relationship within each mode by defining the cost matrix.
### Preliminaries: Wasserstein Distance and Optimal Transport

**Definition (Wasserstein distance between vectors)**

Wasserstein distance between probability vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$ W(\mathbf{a}, \mathbf{b}) = \langle \mathbf{C}, \mathbf{T} \rangle $$

- $\mathbf{C}$ is the cost matrix, where $c_{ij}$ is the cost of moving $a_i$ to $b_j$.
- $\mathbf{T} \in U(\mathbf{a}, \mathbf{b})$ is an Optimal Transport (OT) solution between $\mathbf{a}$ and $\mathbf{b}$.
- $U(\mathbf{a}, \mathbf{b}) = \{ \mathbf{T} \in \mathbb{R}^{n \times m}_+ | \mathbf{T1}_m = \mathbf{a}, \mathbf{T}^T \mathbf{1}_n = \mathbf{b} \}$ is the feasible set of the OT problem.

Solving this OT problem is very expensive\(^5\): it has a complexity of $O(n^3)$.

---

An efficient alternative:

**Definition (Entropy regularized OT problem)\(^6\)**

The entropy regularized OT problem is defined as:

\[
W_V(a, b) = \min_{T \in U(a, b)} \langle C, T \rangle - \frac{1}{\rho} E(T),
\]

(2)

where \(E(T) = -\sum_{i,j=1}^{M,N} t_{ij} \log(t_{ij})\) is the entropy of \(T\).

- It is strictly convex with a unique solution.
- It can be tackled with \(u\) and \(v\) such that \(\text{diag}(u) \exp(-\rho C) \text{diag}(v) \in U(a, b)\).
- Optimal \(u\) and \(v\) can be computed via the Sinkhorn’s algorithm\(^7\).

---


Challenges

However, applying Wasserstein distance to tensor factorization is challenging:

1. **Wasserstein distance is not well-defined for tensors**: It is well-defined for vectors, yet vectorizing tensor yields extremely large vectors, making it infeasible to solve.

2. **Wasserstein distance is difficult to scale**: It requires to solve the OT problems many times in each iteration, which is extremely time consuming.

3. **Real-world input are often large, sparse and non-negative**: Efficient algorithms are possible only when the sparsity structure are fully utilized.
Our Contributions

Contribution 1: Defining Wasserstein Tensor Distance

- SWIFT is the first work that defines Wasserstein distance for tensors.
- It does not assume any particular distribution.
- Therefore, it can handle non-negative inputs, including binaries, counts, and real-values.

Contribution 2: Formulating Wasserstein Tensor Factorization

- We propose SWIFT model to minimize the Wasserstein distance between the input and its CP reconstructions.

Contribution 3: Efficiently Solving Wasserstein Tensor Factorization

- SWIFT effectively explores the sparsity structure of the input and reduces the number of times required to compute OT.
- It reduces the computational time by efficient rearrangement of its sub-problems.
- As a result, it achieves 921x speed up over a naive implementation.
We first define the Wasserstein distance for matrices by **summing that over their columns**:

**Definition (Wasserstein Matrix Distance)**

Given a cost matrix \( \mathbf{C} \in \mathbb{R}^{M \times M}_+ \), the Wasserstein distance between two matrices \( \mathbf{A} = [a_1, ..., a_P] \in \mathbb{R}^{M \times P}_+ \) and \( \mathbf{B} = [b_1, ..., b_P] \in \mathbb{R}^{M \times P}_+ \) is denoted by \( W_M(\mathbf{A}, \mathbf{B}) \), and given by:

\[
W_M(\mathbf{A}, \mathbf{B}) = \sum_{p=1}^{P} W_V(a_p, b_p) = \minimize_{\mathbf{T} \in U(\mathbf{A}, \mathbf{B})} \langle \mathbf{C}, \mathbf{T} \rangle - \frac{1}{\rho} E(\mathbf{T}),
\]

where \( \mathbf{C} = [\mathbf{C}, ..., \mathbf{C}] \), \( \mathbf{T} = [\mathbf{T}_1, ..., \mathbf{T}_p, ..., \mathbf{T}_P] \), and the feasible set \( U(\mathbf{A}, \mathbf{B}) \) is given by:

\[
U(\mathbf{A}, \mathbf{B}) = \left\{ \mathbf{T} \in \mathbb{R}^{M \times MP}_+ \mid \mathbf{T}_p \mathbf{1}_M = a_p, \mathbf{T}_p^T \mathbf{1}_M = b_p \quad \forall p \right\} = \left\{ \mathbf{T} \in \mathbb{R}^{M \times MP}_+ \mid \Delta(\mathbf{T}) = \mathbf{A}, \Psi(\mathbf{T}) = \mathbf{B} \right\},
\]

where \( \Delta(\mathbf{T}) = [\mathbf{T}_1 \mathbf{1}_M, ..., \mathbf{T}_P \mathbf{1}_M] = \mathbf{T}(\mathbf{I}_P \otimes \mathbf{1}_M), \Psi(\mathbf{T}) = [\mathbf{T}_1^T \mathbf{1}_M, ..., \mathbf{T}_P^T \mathbf{1}_M] \) and \( \mathbf{1}_M \) is a vector of all ones with the size of \( M \).
Defining Wasserstein Tensor Distance

Then we can define the Wasserstein distance for tensors by summing that over the matricization along each mode of the tensor:

**Definition (Wasserstein Tensor Distance)**

The Wasserstein distance between \( N \)-th order tensor \( \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N} \) and its reconstruction \( \hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N} \) is denoted by \( W_T(\hat{\mathcal{X}}, \mathcal{X}) \):

\[
W_T(\hat{\mathcal{X}}, \mathcal{X}) = \sum_{n=1}^{N} W_M(\hat{\mathcal{X}}(n), \mathcal{X}(n)) \equiv \sum_{n=1}^{N} \left\{ \text{minimize} \right. \\
\left. \bar{T}_n \in U(\hat{\mathcal{X}}(n), \mathcal{X}(n)) \right\} \left( \langle \bar{C}_n, \bar{T}_n \rangle - \frac{1}{\rho} E(\bar{T}_n) \right), \tag{5}
\]

where \( \mathcal{X}(n) \in \mathbb{R}^{I_n \times l(-n)} \) is the \( n \)-th mode matricization of \( \mathcal{X} \), \( \bar{C}_n = [C_n, C_n, \ldots, C_n] \in \mathbb{R}^{I_n \times l(-n)} \), and \( \bar{T}_n = [T_{n1}, \ldots, T_{nj}, \ldots, T_{n(l_n-n)}] \in \mathbb{R}^{I_n \times l(-n)} \). \( T_{nj} \in \mathbb{R}^{I_n \times l_n} \) is the transport matrix between the columns \( \hat{\mathcal{X}}(n)(::, j) \in \mathbb{R}^{I_n} \) and \( \mathcal{X}(n)(::, j) \in \mathbb{R}^{I_n} \).

The Wasserstein distance \( W_T(\mathcal{X}, \mathcal{Y}) \) defined above is a valid distance and satisfies the metric axioms of positivity, symmetry, and triangle inequality.
Defining Wasserstein Tensor Distance

\[ W_M(A, B) = W_V(a_1, b_1) + \cdots + W_V(a_p, b_p) \]

Illustration of the Wasserstein distances. Left: Wasserstein matrix distance; right: Wasserstein tensor distance.
SWIFT minimizes the Wasserstein tensor distance between input and its CP reconstruction:

**Optimization problem**

\[
\begin{align*}
\text{minimize} & \quad \sum_{n=1}^{N} \left( \langle C_n, \bar{T}_n \rangle - \frac{1}{\rho} E(\bar{T}_n) \right) \\
\text{subject to} & \quad \hat{X} = [A_1, \ldots, A_N] \\
& \quad \bar{T}_n \in U(\hat{X}_n, X_n), \quad n = 1, \ldots, N
\end{align*}
\]

**Constraint Relaxation using the generalized KL-divergence**

\[
\begin{align*}
\text{minimize} & \quad \sum_{n=1}^{N} \left( \langle C_n, \bar{T}_n \rangle - \frac{1}{\rho} E(\bar{T}_n) + \lambda \left( KL(\Delta(\bar{T}_n)||A_n(A_n^{(-n)})^T) + KL(\Psi(\bar{T}_n)||X_n) \right) \right) \\
& \quad \text{Part } P_1 \\
& \quad \text{Part } P_2 \\
& \quad \text{Part } P_3
\end{align*}
\] (6)

We alternate between \(A_n\) and \(\bar{T}_n\) to solve Eq. (6).
Note that: $\overline{T}_n = [T_{n1}, \ldots, T_{nj}, \ldots, T_{nl(-n)}] \in \mathbb{R}_{+}^{I_n \times I_n l(-n)}$.

The number of optimal transport problems to solve is: $l(-n) = l_1 \times \cdots \times l_{n-1} \times l_{n+1} \times \cdots \times l_N$.

Instead, we use the property of the OT solution $T_{nj}^* 1 = \text{diag}(u_j) K_n v_j = u_j \ast (K_n v_j)$; therefore:

**Proposition 2**

$$\Delta(\overline{T}_n) = [T_{n1} 1, \ldots, T_{nj} 1, \ldots, T_{nl(-n)} 1] = U_n \ast (K_n V_n)$$

minimizes (6), where $K_n = e^{(-\rho c_n - 1)} \in \mathbb{R}^{I_n \times I_n}$, $U_n = (\hat{X}(n))^{\Phi} \odot (K_n (X(n) \odot (K_n^T U_n))^{\Phi})^{\Phi}$, $V_n = (X(n) \odot (K_n^T U_n))^{\Phi}$, $\Phi = \frac{\lambda \rho}{\lambda \rho + 1}$, and $\odot$ indicates element-wise division.
Efficient Algorithms: 1. Solving for OT Problems ($\overline{T}_n$)

Exploring the sparsity structure for efficiently computing $\Delta(\overline{T}_n)$

- Many columns of the $X_{(n)}$ are all zeros; thus they can be ignored when computing $V_n$.
- Besides, each column of $V_n$ can be computed in parallel.

*SWIFT explores sparsity structure in input data $X_{(n)}$ and drops zero values columns.*
Sub-problem for the CP factor matrices $A_n$

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} KL\left(\Delta(T_i) \parallel A_i(A_i^{(-i)})^T\right)\\
\text{subject to} & \quad A_n \geq 0
\end{align*}
\]

(8)

**Challenge:** $A_n$ is also involved in the Khatri-Rao product $A_i^{(-i)}$.

To tackle this, we define an rearranging operator $\Pi$, such that

**Efficient rearranging operation**

\[
\Pi(A_i(A_i^{(-i)})^T, n) = A_n(A_n^{(-n)})^T \in \mathbb{R}_{+}^{l_n \times l_{(-n)}} \quad \forall \ i \neq n.
\]

(9)

In this way, the Khatri-Rao product term no longer contain the factor matrix $A_n$. 

\[14/23\]
Efficient Algorithms: 2. Updating CP factors ($A_n$)

With this operator, the sub-problem is equivalent to:

**Rearranged sub-problem for $A_n$**

\[
\text{minimize } \begin{bmatrix} \Pi(\Delta(T_1), n) \\ \cdots \\ \Pi(\Delta(T_i), n) \\ \cdots \\ \Pi(\Delta(T_N), n) \end{bmatrix} \quad \text{KL} \left( \begin{bmatrix} A_n(A_{-n})^T \\ \cdots \\ A_n(A_{-n})^T \\ \cdots \\ A_n(A_{-n})^T \end{bmatrix} \right)
\]

With the rearranged objective function, the factor matrix $A_n$ can be efficiently updated via multiplicative update rules\(^8\).

Experiments: Datasets and Evaluation Metrics

**BBC News**
- a third-order counting tensor with size of 400 articles by 100 words by 100 words
- downstream task: article category classification; evaluated by accuracy.

**Sutter**
- a dataset collected from a large real-world health provider network
- a third-order binary tensor with size of 1000 patients by 100 diagnoses by 100 medications
- downstream task: heart failure onset; evaluated by PR-AUC.

We use the pair-wise cosine distance to compute the cost matrices for each mode of the two datasets.

---

We compare against the following tensor factorization models with different loss functions:

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Type</th>
<th>Underlying Distribution Assumption</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-ALS</td>
<td>MSE Loss</td>
<td>Gaussian</td>
<td>(Bader &amp; Kolda 2007)</td>
</tr>
<tr>
<td>CP-NMU</td>
<td>MSE Loss</td>
<td>Gaussian</td>
<td>(Bader &amp; Kolda 2007)</td>
</tr>
<tr>
<td>Supervised CP</td>
<td>MSE Loss</td>
<td>Gaussian</td>
<td>(Kim et al. 2017)</td>
</tr>
<tr>
<td>Similarity based CP</td>
<td>MSE Loss</td>
<td>Gaussian</td>
<td>(Kim et al. 2017)</td>
</tr>
<tr>
<td>CP-Continuous</td>
<td>Gamma Loss</td>
<td>Gamma</td>
<td>(Hong et al. 2020)</td>
</tr>
<tr>
<td>CP-Binary</td>
<td>Log Loss</td>
<td>Bernoulli</td>
<td>(Hong et al. 2020)</td>
</tr>
<tr>
<td>CP-APR</td>
<td>KL Loss</td>
<td>Poisson</td>
<td>(Chi &amp; Kolda 2012)</td>
</tr>
</tbody>
</table>
Experimental Results: Classification Performance

<table>
<thead>
<tr>
<th></th>
<th>R=5</th>
<th>R=10</th>
<th>R=20</th>
<th>R=30</th>
<th>R=40</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BBC News Dataset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP-ALS</td>
<td>.521 ± .033</td>
<td>.571 ± .072</td>
<td>.675 ± .063</td>
<td>.671 ± .028</td>
<td>.671 ± .040</td>
</tr>
<tr>
<td>CP-NMU</td>
<td>.484 ± .039</td>
<td>.493 ± .048</td>
<td>.581 ± .064</td>
<td>.600 ± .050</td>
<td>.650 ± .031</td>
</tr>
<tr>
<td>Supervised CP</td>
<td>.506 ± .051</td>
<td>.625 ± .073</td>
<td>.631 ± .050</td>
<td>.665 ± .024</td>
<td>.662 ± .012</td>
</tr>
<tr>
<td>Similarity Based CP</td>
<td>.518 ± .032</td>
<td>.648 ± .043</td>
<td>.638 ± .021</td>
<td>.662 ± .034</td>
<td>.673 ± .043</td>
</tr>
<tr>
<td>CP-Continuous</td>
<td>.403 ± .051</td>
<td>.481 ± .056</td>
<td>.528 ± .022</td>
<td>.559 ± .024</td>
<td>.543 ± .043</td>
</tr>
<tr>
<td>CP-Binary</td>
<td>.746 ± .058</td>
<td>.743 ± .027</td>
<td>.737 ± .008</td>
<td>.756 ± .062</td>
<td>.743 ± .044</td>
</tr>
<tr>
<td>CP-APR</td>
<td>.675 ± .059</td>
<td>.768 ± .033</td>
<td>.753 ± .035</td>
<td>.743 ± .033</td>
<td>.746 ± .043</td>
</tr>
<tr>
<td><strong>SWIFT</strong></td>
<td><strong>.759 ± .013</strong></td>
<td><strong>.781 ± .013</strong></td>
<td><strong>.803 ± .010</strong></td>
<td><strong>.815 ± .005</strong></td>
<td><strong>.818 ± .022</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sutter Data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP-ALS</td>
<td>.327 ± .072</td>
<td>.333 ± .064</td>
<td>.311 ± .068</td>
<td>.306 ± .065</td>
<td>.332 ± .098</td>
</tr>
<tr>
<td>CP-NMU</td>
<td>.300 ± .054</td>
<td>.294 ± .064</td>
<td>.325 ± .085</td>
<td>.344 ± .068</td>
<td>.302 ± .071</td>
</tr>
<tr>
<td>Supervised CP</td>
<td>.301 ± .044</td>
<td>.305 ± .036</td>
<td>.309 ± .054</td>
<td>.291 ± .037</td>
<td>.293 ± .051</td>
</tr>
<tr>
<td>Similarity Based CP</td>
<td>.304 ± .042</td>
<td>.315 ± .041</td>
<td>.319 ± .063</td>
<td>.296 ± .041</td>
<td>.303 ± .032</td>
</tr>
<tr>
<td>CP-Continuous</td>
<td>.252 ± .059</td>
<td>.237 ± .043</td>
<td>.263 ± .065</td>
<td>.244 ± .053</td>
<td>.256 ± .077</td>
</tr>
<tr>
<td>CP-Binary</td>
<td>.301 ± .061</td>
<td>.325 ± .079</td>
<td>.328 ± .080</td>
<td>.267 ± .074</td>
<td>.296 ± .063</td>
</tr>
<tr>
<td>CP-APR</td>
<td>.305 ± .075</td>
<td>.301 ± .068</td>
<td>.290 ± .052</td>
<td>.313 ± .082</td>
<td>.304 ± .086</td>
</tr>
<tr>
<td><strong>SWIFT</strong></td>
<td><strong>.364 ± .063</strong></td>
<td><strong>.350 ± .031</strong></td>
<td><strong>.350 ± .040</strong></td>
<td><strong>.369 ± .066</strong></td>
<td><strong>.374 ± .044</strong></td>
</tr>
</tbody>
</table>

SWIFT outperforms all models consistently by a large margin.
### Experimental Results: Classification Performance

#### Comparison against widely-adopted classifiers:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy on BBC</th>
<th>PR-AUC on Sutter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso Logistic Regression</td>
<td>.728 ± .013</td>
<td>.308 ± .033</td>
</tr>
<tr>
<td>Random Forest</td>
<td>.628 ± .049</td>
<td>.318 ± .083</td>
</tr>
<tr>
<td>Multi-Layer Perceptron</td>
<td>.690 ± .052</td>
<td>.305 ± .054</td>
</tr>
<tr>
<td>K-Nearest Neighbor</td>
<td>.596 ± .067</td>
<td>.259 ± .067</td>
</tr>
<tr>
<td>SWIFT (R=5)</td>
<td>.759 ± .013</td>
<td>.364 ± .063</td>
</tr>
<tr>
<td>SWIFT (R=40)</td>
<td><strong>.818 ± .020</strong></td>
<td><strong>.374 ± .044</strong></td>
</tr>
</tbody>
</table>

SWIFT with rank of 5 already outperforms all other classifiers compared.
Experimental Results: Classification Performance on Noisy Data

We inject random noise to the BBC News data and run all models using the noisy data:

SWIFT outperforms all baselines, especially for medium and high noise levels.
We set $R = 40$, switch off all parallelization of SWIFT for fair comparison and measure the running time of all models. SWIFT is as scalable as other CP factorization models.
### Experimental Results: Interpretability of SWIFT

We interpret the factor matrices learned for Sutter datasets. Following are three examples:

- Each group (phenotype) contains clinically relevant diagnoses and medications.
- The weight indicates the lasso logistic regression coefficient for heart failure (HF) prediction.
- First two groups are clinically relevant to HF, but the third is not.
- The clinical meaningfulness is endorsed by a medical expert.

---

<table>
<thead>
<tr>
<th>Atrial Fibrillation (Weight= 21.93)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dx-Essential hypertension [98.]</td>
</tr>
<tr>
<td>Dx-Disorders of lipid metabolism [53.]</td>
</tr>
<tr>
<td>Dx-Cardiac dysrhythmias [106.]</td>
</tr>
<tr>
<td>Rx-Calcium Channel Blockers</td>
</tr>
<tr>
<td>Rx-Alpha-Beta Blockers</td>
</tr>
<tr>
<td>Rx-Angiotensin II Receptor Antagonists</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cardiometabolic Disease (Weight= 19.58)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dx-Diabetes mellitus without complication [49.]</td>
</tr>
<tr>
<td>Dx-Essential hypertension [98.]</td>
</tr>
<tr>
<td>Dx-Disorders of lipid metabolism [53.]</td>
</tr>
<tr>
<td>Rx-Diagnostic Tests</td>
</tr>
<tr>
<td>Rx-Biguanides</td>
</tr>
<tr>
<td>Rx-Diabetic Supplies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mental Disorder (Weight= -16.22)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dx-Anxiety disorders [651]</td>
</tr>
<tr>
<td>Dx-Menopausal disorders [173.]</td>
</tr>
<tr>
<td>Dx-Depressive disorders [6572]</td>
</tr>
<tr>
<td>Rx-Benzodiazepines</td>
</tr>
<tr>
<td>Rx-Selective Serotonin Reuptake Inhibitors (SSRIs)</td>
</tr>
<tr>
<td>Rx-Serotonin Modulators</td>
</tr>
</tbody>
</table>

SWIFT yields interpretable factor matrices.
We define the Wasserstein distance between two tensors and propose SWIFT, a Wasserstein tensor factorization model.

We derive an efficient learning algorithm by exploring the sparsity structure and introducing efficient rearrangement operator.

Empirical evaluations demonstrate that SWIFT consistently outperforms baselines in downstream prediction tasks, even in the presence of heavy noise.

SWIFT is also shown scalable and interpretable.
Thank you!

All questions and comments are greatly appreciated!