

A Simple, Fast, and Flexible Framework for Matrix Completion with Infinite Width Networks

Adit Radha

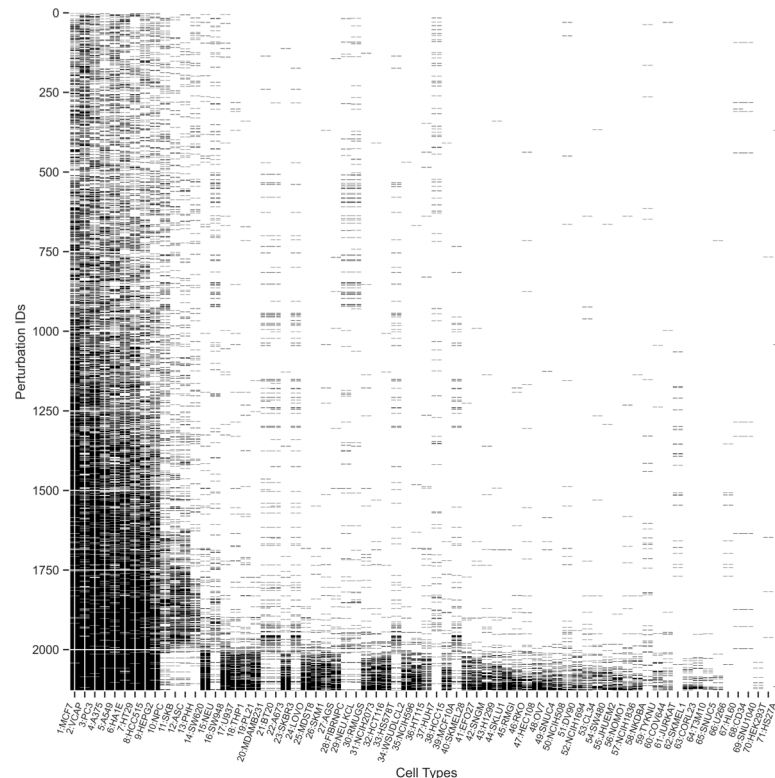
Joint work with George Stefanakis, Mikhail Belkin, and Caroline Uhler

October 8, 2021

Motivating Application: Virtual Drug Screening

- CMap: ~1.3 million gene expression vectors across ~70 cell types and ~20000 perturbations
 - [A. Subramanian, R. Narayan, S.M. Corsello, et al., Cell, 2017]
- Sequencing all drug and cell type combinations is computationally difficult.
- **Goal:** Predict expression for unseen cell type and drug combinations from observed data.

Availability in a subset of CMap

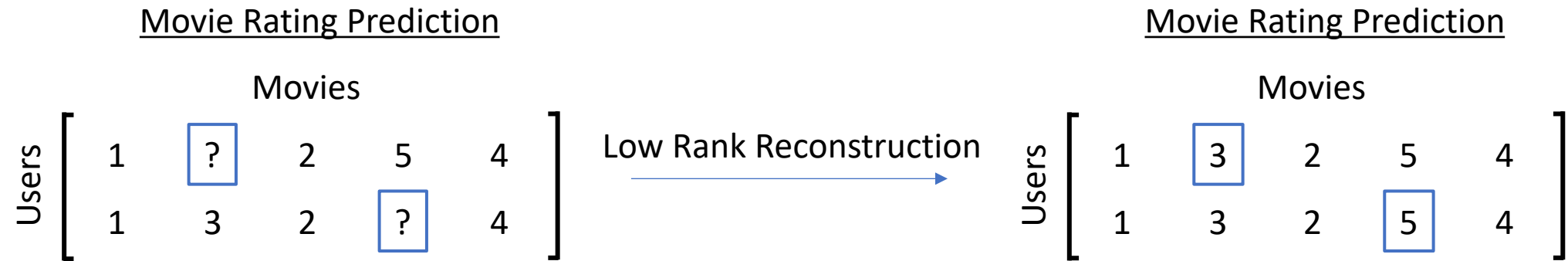


Formulate as
Matrix Completion

Drug Imputation

Gene Expression	Drug				
	.9	.8	.1	?	?
	.1	.1	.3	?	?
	.5	.4	.7	?	?
	.3	.2	.5	?	?

Standard Approaches for Matrix Completion



$$\underbrace{\begin{bmatrix} 1 & ? & 2 & 5 & 4 \\ 1 & 3 & 2 & ? & 4 \end{bmatrix}}_Y = \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{W_1} \times \underbrace{\begin{bmatrix} 1 & 3 & 2 & 5 & 4 \end{bmatrix}}_{W_2}$$

Low Rank Completions are not always Ideal

Drug Imputation

Gene Expression	Drug				
	.9	.8	.1	?	?
	.1	.1	.3	?	?
	.5	.4	.7	?	?
	.3	.2	.5	?	?

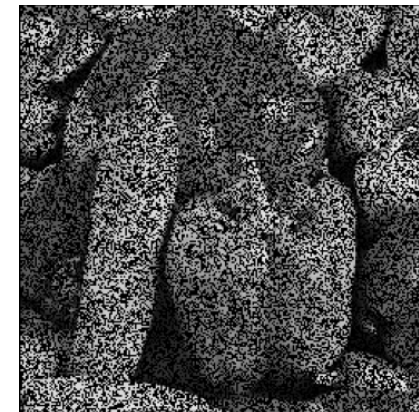
Low Rank Reconstruction

Gene Expression	Drug				
	.9	.8	.1	0	0
	.1	.1	.3	0	0
	.5	.4	.7	0	0
	.3	.2	.5	0	0

Image Inpainting



Image Reconstruction



Nonlinear Framework for Matrix Completion

Standard Linear Approach

$$Y = f(\mathbf{W})$$

Example: $Y = W_1 W_2$

$$Y = W_1 W_2 Z$$

$$=$$

$$\times$$

--	--	--

$$\times$$

1	0	0
0	1	0
0	0	1

Our Proposed Framework

$$Y = f_Z(\mathbf{W})$$

Example: $Y = W_1 \phi(W_2 Z)$

$$Y = W_1 W_2 Z$$

$$=$$

$$\times \phi \left[\right.$$

--	--

$$\times$$

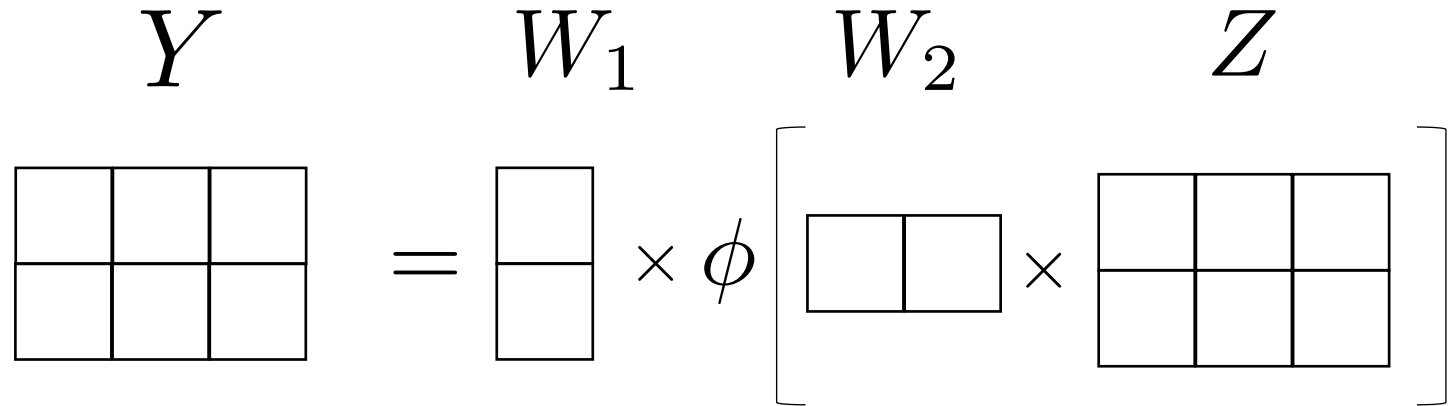
$$\left. \right]$$

What makes this framework simple, fast, and flexible?

Our Proposed Framework

$$Y = f_Z(\mathbf{W})$$

Example: $Y = W_1 \phi(W_2 Z)$

$$Y = W_1 W_2 Z$$


Recent Findings Powering our Framework:

- Larger (e.g. wider) neural networks generalize better.
 - [Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal, PNAS, 2019]
- Training infinite width neural networks is equivalent to solving kernel regression.
 - [Arthur Jacot, Franck Gabriel, Clément Hongler, NeurIPS, 2018]

An Over-parameterized Perspective

We solve: $\arg \min_{\mathbf{W}} \sum_{i,j} (Y_{i,j} - f_Z(\mathbf{W})_{i,j})^2$

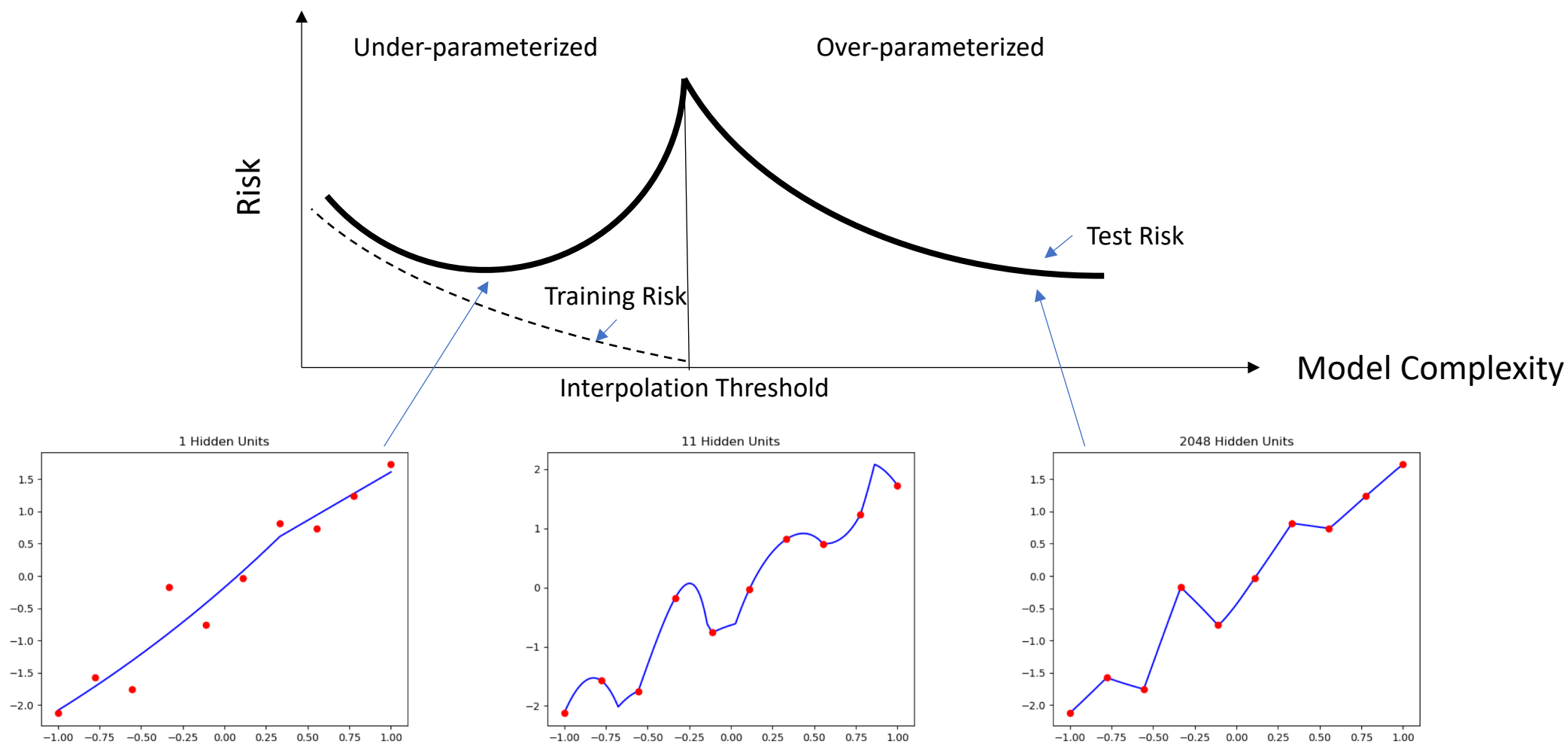
$$Y = W_1 \times \phi \left[W_2 \times Z \right]$$

How should we select network width, k ?

$$Y = W_1 \in \mathbb{R}^{2 \times k} \times \dots \times \phi \left[W_2 \in \mathbb{R}^{k \times 2} \times Z \right]$$

Key Idea:
Consider k extremely large (k goes to infinity).

Benefits of Over-parameterization: Double Descent



[Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal, PNAS 2019]

Benefits of Extreme Over-parameterization

Typically, neural nets as maps on data:

$$f(x) = W_1 \phi(W_2 x) \quad ; \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

Instead, neural nets as maps on parameters:

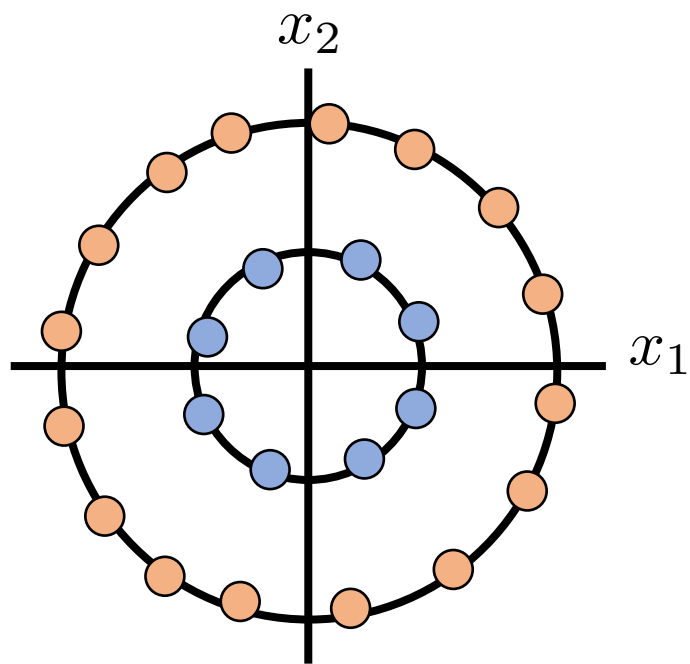
$$f_x(\mathbf{W}) = W_1 \phi(W_2 x) \quad ; \quad f : \mathbb{R}^p \rightarrow \mathbb{R}$$

Linearizing the above system around an initial set of weights:

$$\tilde{f}_x(\mathbf{W}) = f_x(\mathbf{W}^{(0)}) + \nabla f_x(\mathbf{W}^{(0)})^T (\mathbf{W} - \mathbf{W}^{(0)}) \quad \leftarrow \text{This is a linear system in } \mathbf{W}.$$

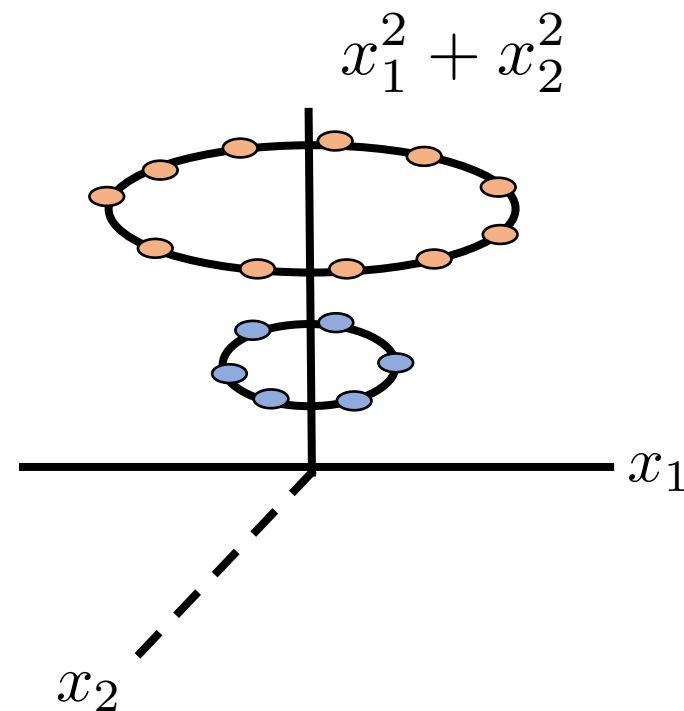
Training the linear system above is equivalent to training the neural network, as width approaches infinity.

A Visualization to Provide Intuition



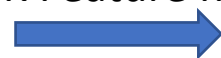
Data, X , in 2D that is not linearly separable.

$$x \in \mathbb{R}^d$$



After transformation, 3D data is linearly separable.

NTK Feature Map



$$\nabla f_x(\mathbf{W}) \in \mathbb{R}^p$$

Neural Tangent Kernel (NTK)

Given a dataset: $\{x^{(i)}, y^{(i)}\}_{i=1}^n$

As network widths go to infinity, these two problems yield the same functional solution:

$$\arg \min_{\mathbf{W}} \sum_{i=1}^n (y^{(i)} - f_{x^{(i)}}(\mathbf{W}))^2 \quad \longleftrightarrow \quad \arg \min_w \sum_{i=1}^n (y^{(i)} - w \nabla f_{x^{(i)}}(\mathbf{W}^{(0)}))^2$$

For infinite width, solve kernel regression with the Neural Tangent Kernel (NTK):

$$K(x, x') = \langle \nabla f_x(\mathbf{W}^{(0)}), \nabla f_{x'}(\mathbf{W}^{(0)}) \rangle$$

Key Takeaways:

- There is no need to train very wide neural networks. Instead, we can solve kernel regression.
- Kernel regression can be solved efficiently on the GPU via EigenPro. [Siyuan Ma, Mikhail Belkin, 2019]

Simple and Fast Matrix Completion with the NTK

Recall our goal for matrix completion: $\arg \min_{\mathbf{W}} \sum_{i,j} (Y_{i,j} - f_Z(\mathbf{W})_{i,j})^2$

How do we compute the NTK? What are the training examples and what are the labels?

$$f_Z(\mathbf{W})_{i,j} = M_{ij}^T f_Z(\mathbf{W}) \quad M_{ij} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (1 \text{ in coordinate } (i,j), 0 \text{ everywhere else})$$

Dataset is actually: $\{M_{ij}, Y_{i,j}\}_{i,j}$

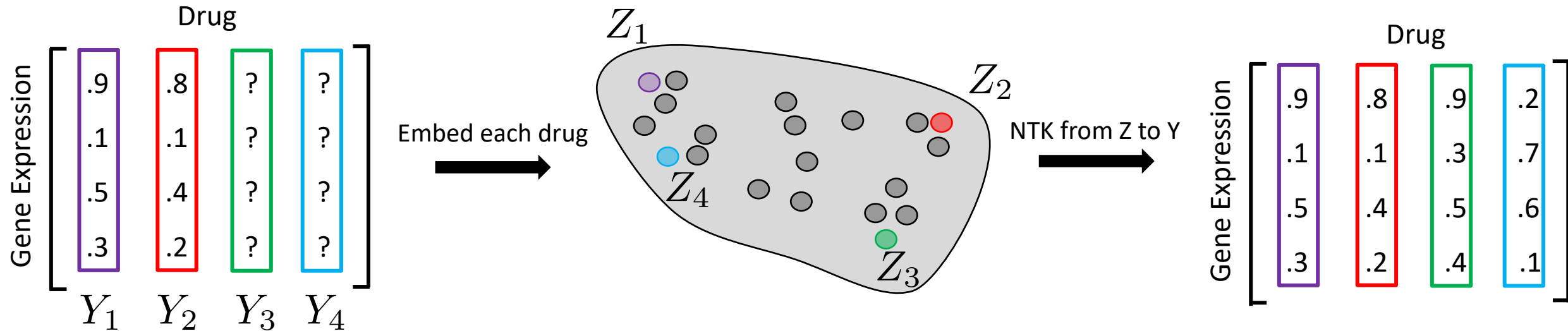
Kernel is actually: $K(M_{ij}, M_{i'j'}) = K[i, j, i', j']$

Key Insight: Matrix completion with neural networks simply involves learning a map from coordinates to observations.

Simple and Fast Matrix Completion with the NTK

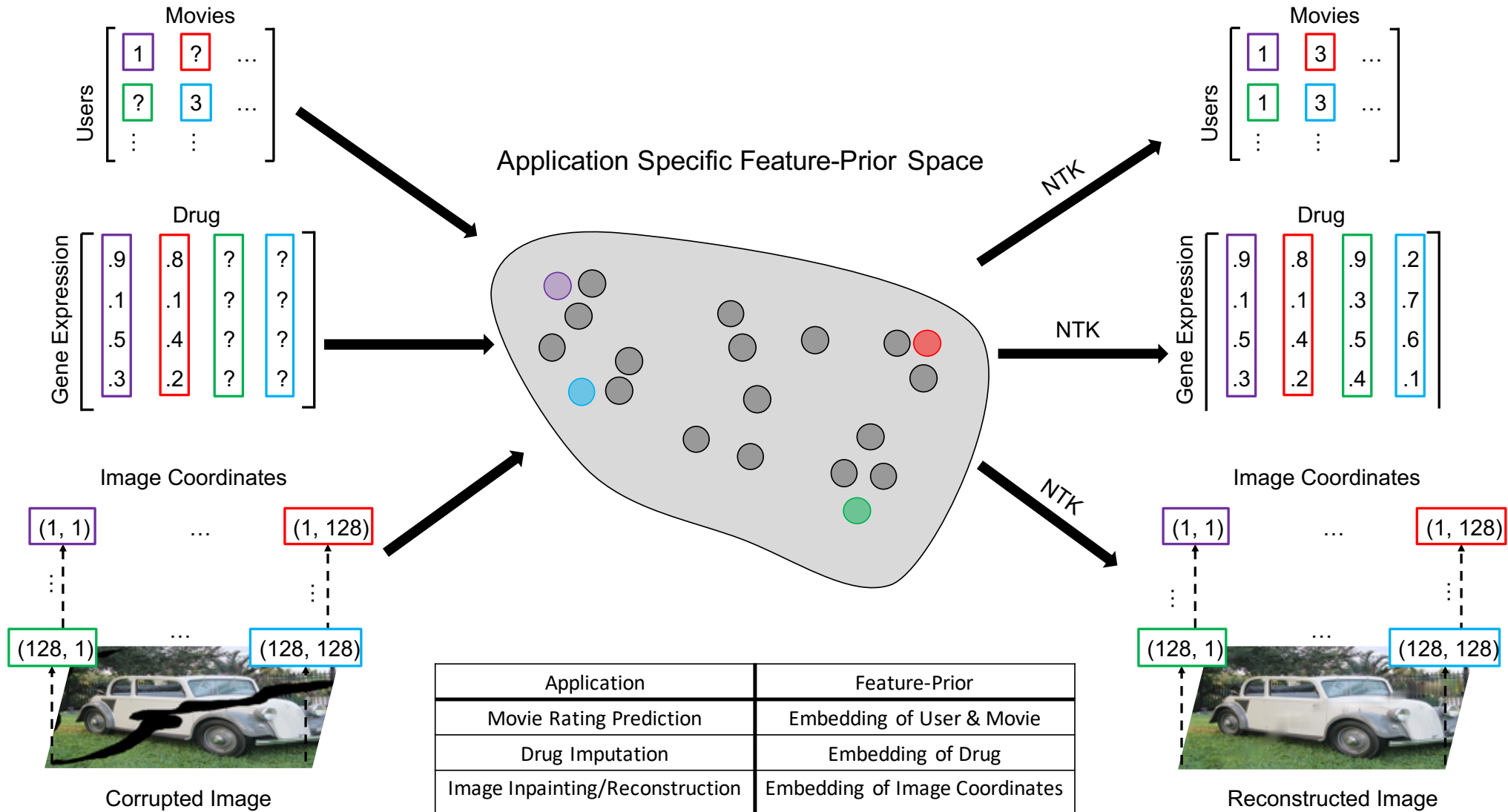
$$Y = f_Z(\mathbf{W})$$

$$\text{Example: } Y = W_1 \phi(W_2 Z)$$



Z encodes information about similarity between drugs. This is a *semi-supervised* approach to matrix completion.

Flexibility through Feature Prior



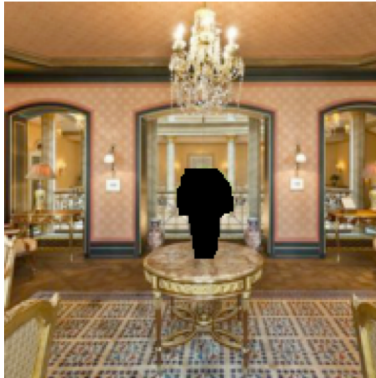
Application 1: Image Inpainting/Reconstruction

Large Hole Inpainting

Ground Truth

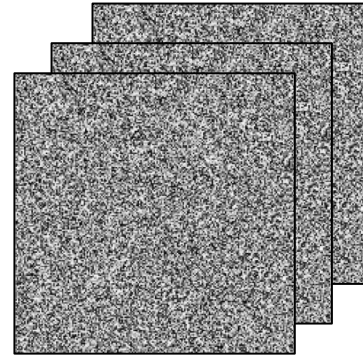


Corrupted Image (Y)



Deep Image Prior

[Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky, CVPR, 2018]

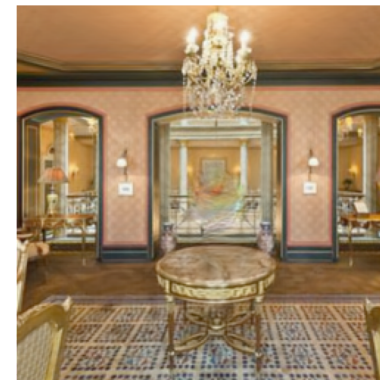
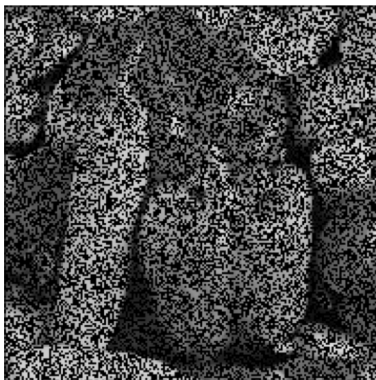


Convolutional Net
(U-Net)



Only reconstruct
observed pixels

Reconstruction



Reconstruction from Convolutional Net

Application 1: Image Inpainting/Reconstruction

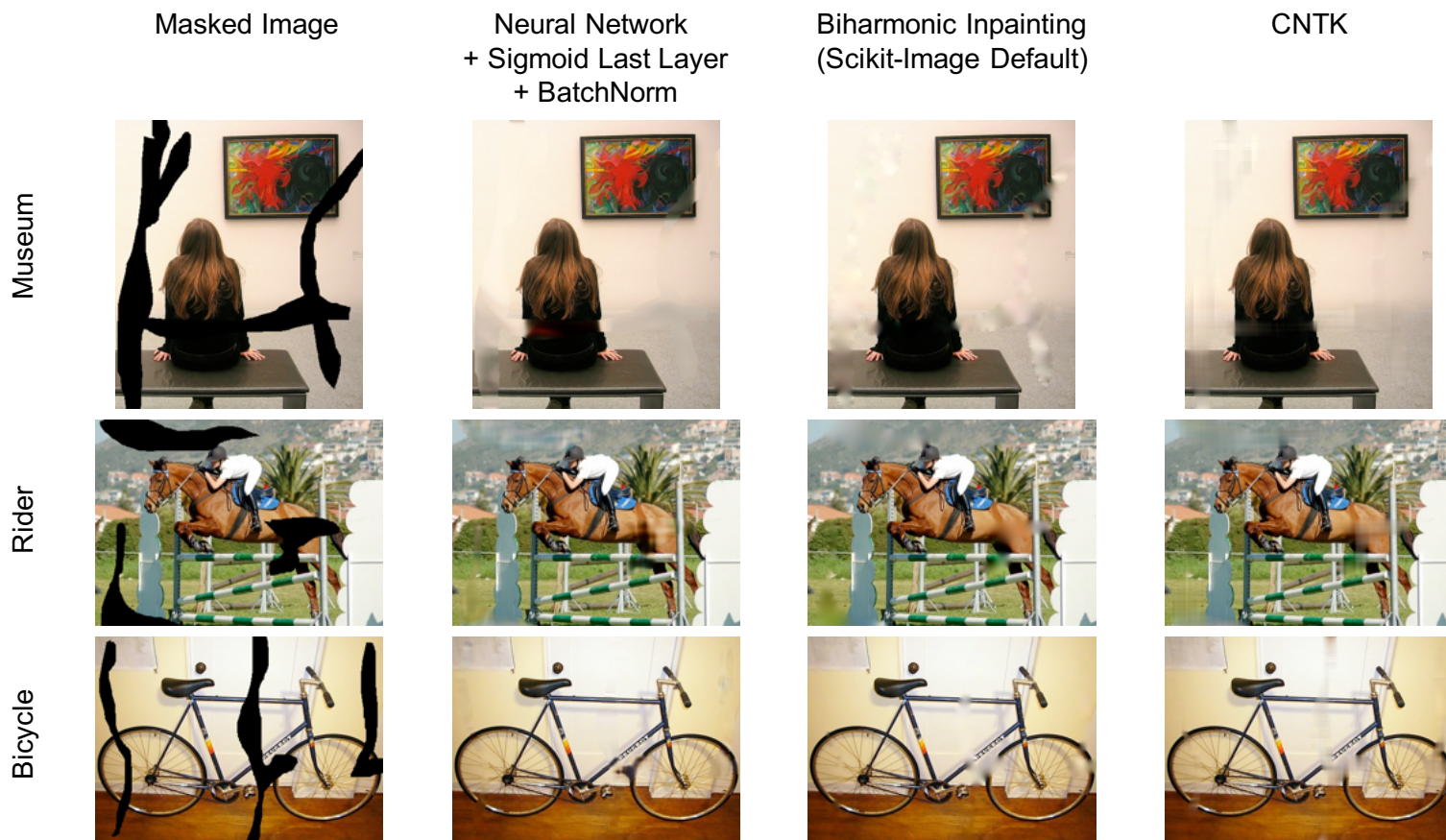


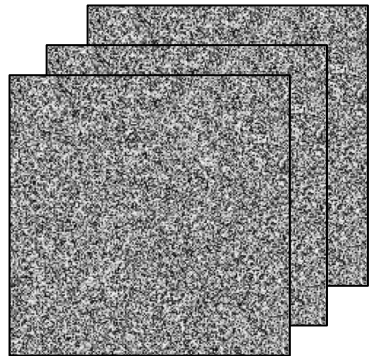
Image	CNTK (PSNR)	Neural Network + Sigmoid Last Layer + BatchNorm (PSNR)	Biharmonic (PSNR)
Museum	31.90	30.69	30.03
White Car	28.66	28.73	26.20
Bicycle	27.67	28.57	28.67
Chair	29.87	29.88	27.81
Car Field	28.91	29.94	27.67
Rider	29.20	27.76	29.38
Library	21.73	20.76	17.71
Vase	31.75	31.51	28.96
Pool	34.51	33.08	34.62
Average	29.36	28.99	27.89

Interpretability through CNTK

Any imputed pixel is the linear combinations of other pixels. Kernel gives us weighting.

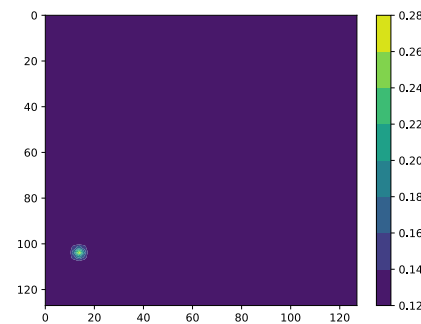
Recall kernel is actually: $K(M_{ij}, M_{i'j'}) = K[i, j, i', j']$

Uniform Random Feature Prior

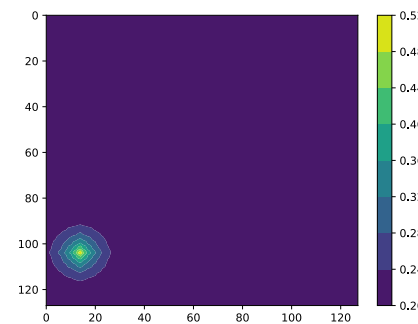


Kernel Visualization for Coordinate (104, 14)

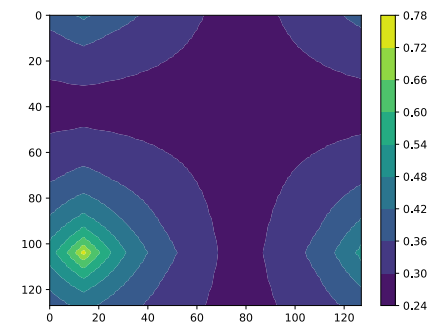
2 Downsampling + Upsampling



4 Downsampling + Upsampling



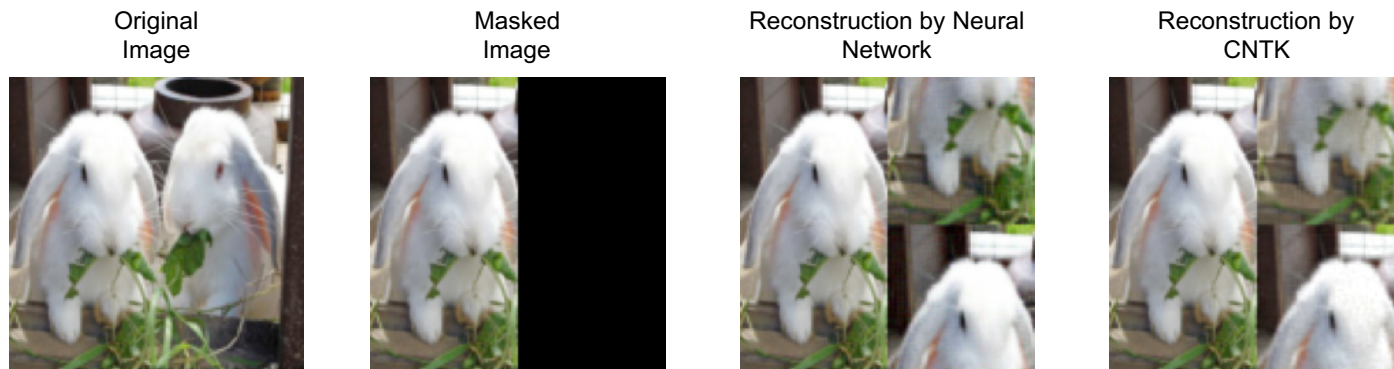
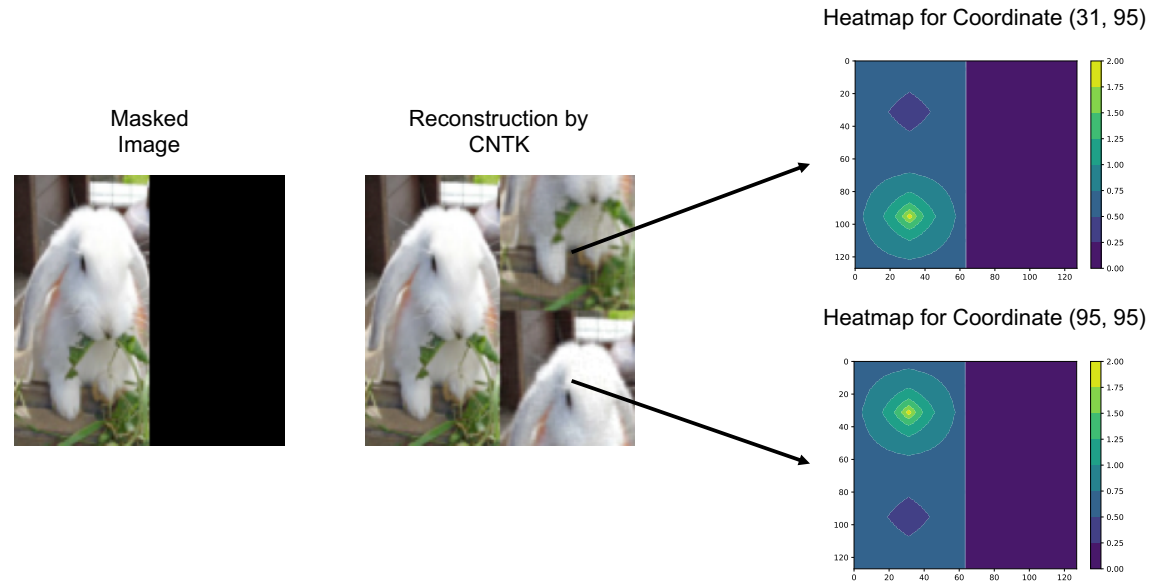
6 Downsampling + Upsampling



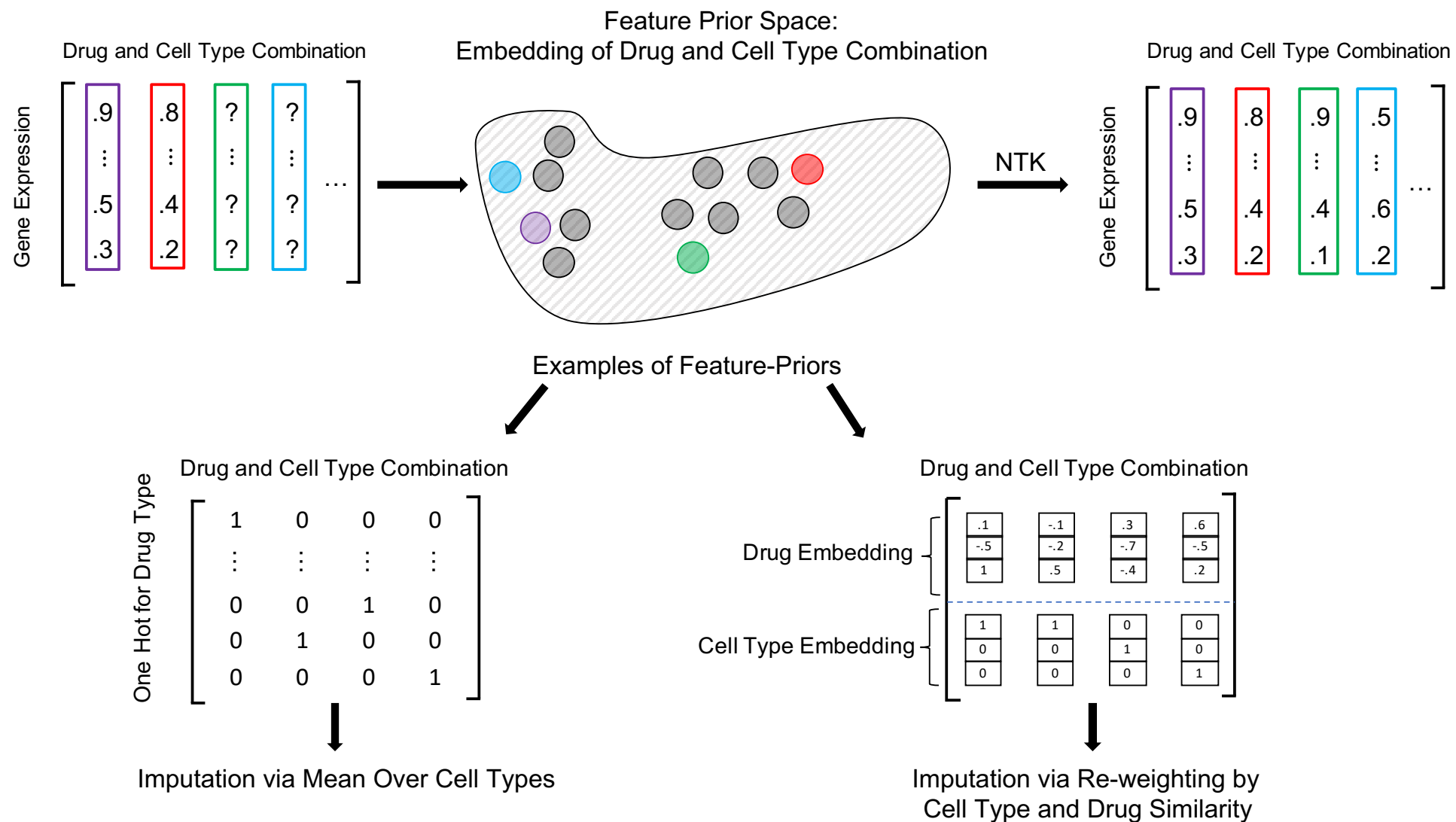
This is a good prior for images: each pixel is imputed as a combination of nearby pixels.

Other Feature Priors are not as Effective

Identity Feature Prior

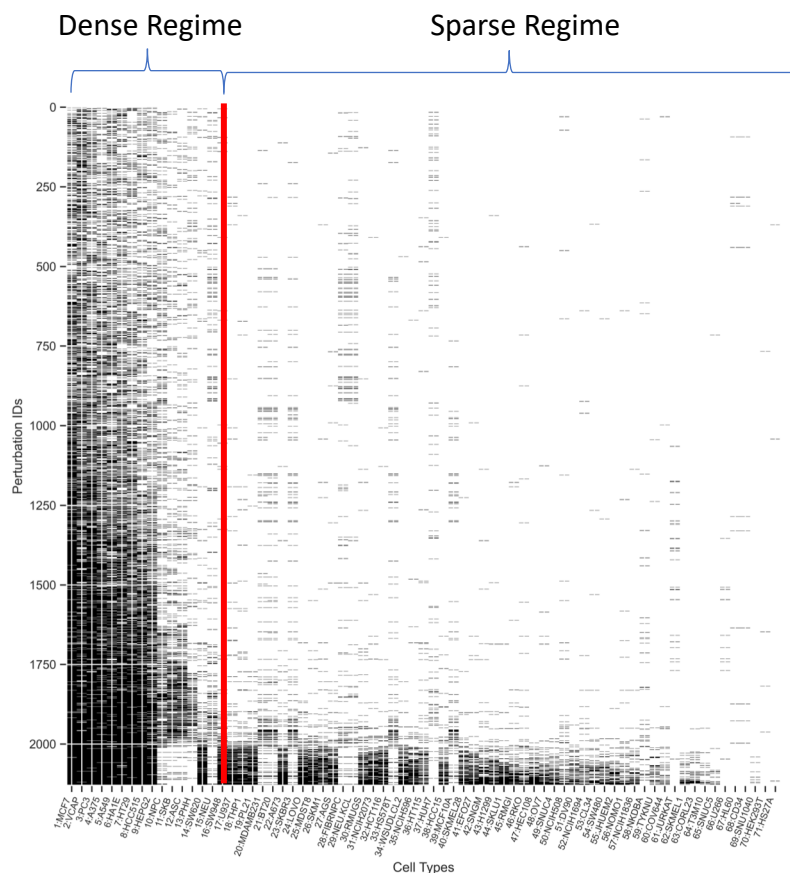


Application 2: Virtual Drug Screening



Application 2: Virtual Drug Screening

Drug and Cell Type Availability Matrix



Dataset: 978 genes x 2130 drugs x 71 cell types
14,336 observations

CMap
(Sparse Regime)

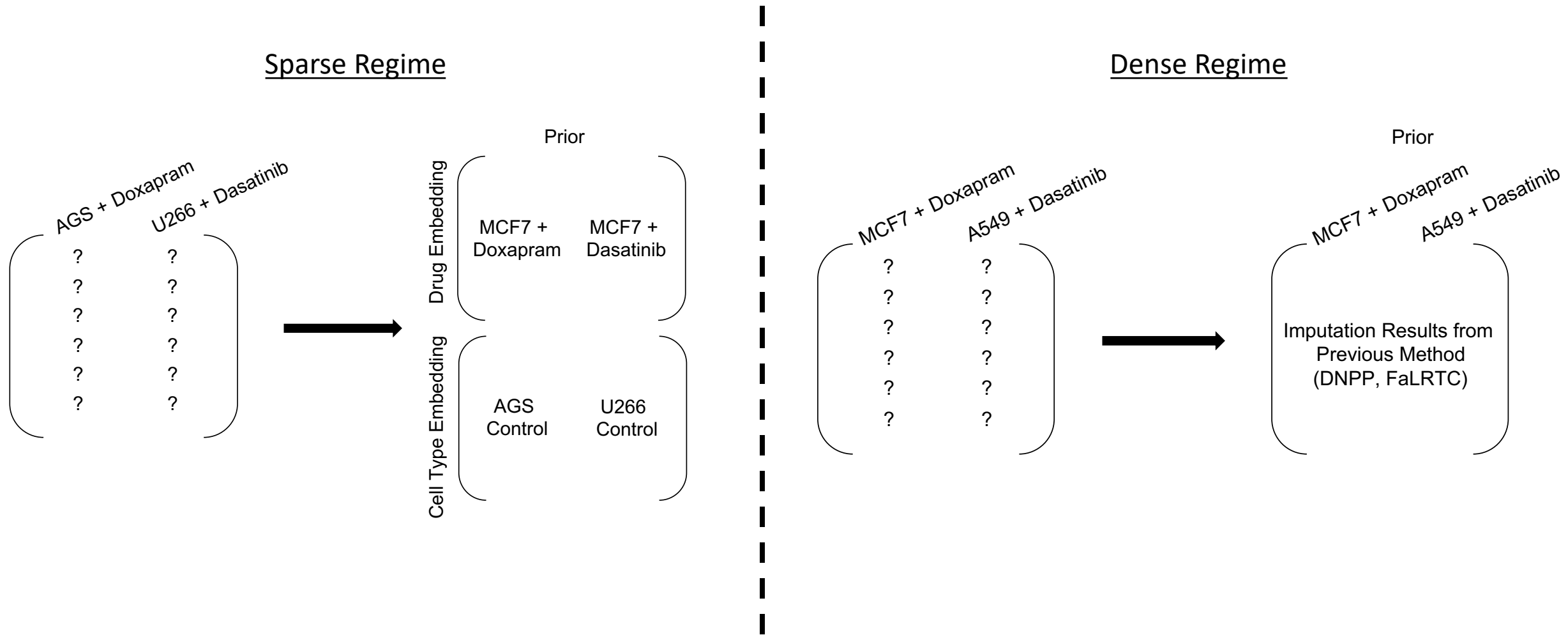
Evaluation Metric*	Mean Over Cell Type (Naïve Baseline)	FaLRTC (Liu et al. 2013)	DNPP (Hodos et al. 2018)	NTK (Ours)
Pearson r	0.450	0.544	0.538	0.573
Mean R ²	0.197	0.285	0.278	0.324
Mean Cosine Similarity	0.448	0.536	0.532	0.565

CMap
(Full Dataset)

Evaluation Metric*	Mean Over Cell Type (Naïve Baseline)	FaLRTC (Liu et al. 2013)	DNPP (Hodos et al. 2018)	NTK (Ours)
Pearson r	0.374 ± 0.0004	0.545 ± 0.0003	0.556 ± 0.0003	0.572 ± 0.0002
Mean R ²	0.134 ± 10 ⁻⁵	0.286 ± 0.0003	0.296 ± 0.0004	0.320 ± 0.0002
Mean Cosine Similarity	0.371 ± 10 ⁻⁵	0.536 ± 0.0004	0.541 ± 0.0004	0.554 ± 0.0002

*Higher is better, with a maximum of 1.

Application 2: Virtual Drug Screening



Overview

- Benefits of Our Framework:

- Simple, fast: Connection between infinite width nets and kernel methods.
- Flexible: Connection between feature prior and semi-supervised learning.

- Resources:

- Our Paper: <https://arxiv.org/abs/2108.00131>
- Our Matrix Completion Code: https://github.com/uhrerlab/ntk_matrix_completion
- Tutorial on NTK: https://github.com/aradha5772/deep_learning_theory_tutorial

- Significance/Future Directions:

- Our NTK framework is an accessible way to achieve competitive results for a variety of matrix completion applications.
- Utilize multimodal data and extend to additional modalities e.g., tensors, audio, and video.