PNAS RESEARCH ARTICLE | COMPUTER SCIENCES

Simple, fast, and flexible framework for matrix completion with infinite width neural networks

Adityanarayanan Radhakrishnan[a,b], George Stefanakis[a,b], Mikhail Belkin[c], and Caroline Uhler[a,b,d,1]

Edited by Terrence Sejnowski, Salk Institute for Biological Studies, La Jolla, CA; received August 15, 2021; accepted January 17, 2022

Matrix completion problems arise in many applications including recommendation systems, computer vision, and genomics. Increasingly larger neural networks have been successful in many of these applications but at considerable computational costs. Remarkably, taking the width of a neural network to infinity allows for improved computational performance. In this work, we develop an infinite width neural network framework for matrix completion that is simple, fast, and flexible. Simplicity and speed come from the connection between the infinite width limit of neural networks and kernels known as neural tangent kernels (NTK). In particular, we derive the NTK for fully connected and convolutional neural networks for matrix completion. The flexibility stems from a feature prior, which allows encoding relationships between coordinates of the target matrix, akin to semisupervised learning. The effectiveness of our framework is demonstrated through competitive results for virtual drug screening and image inpainting/reconstruction. We also provide an implementation in Python to make our framework accessible on standard hardware to a broad audience.

matrix completion | infinite width neural networks | neural tangent kernel | drug response imputation | image inpainting

Significance

Matrix completion is a fundamental problem in machine learning that arises in various applications. We envision that our infinite width neural network framework for matrix completion will be easily deployable and produce strong baselines for a wide range of applications at limited computational costs. We demonstrate the flexibility of our framework through competitive results on virtual drug screening and image inpainting/reconstruction. Simplicity and speed are showcased by the fact that most results in this work require only a central processing unit and commodity hardware. Through its connection to semisupervised learning, our framework provides a principled approach for matrix completion that can be easily applied to problems well beyond those of image completion and virtual drug screening considered in this paper.

Matrix completion is a fundamental problem in machine learning, arising in a variety of applications from collaborative filtering to virtual drug screening and image inpainting/reconstruction. Given a matrix Y with only a subset of coordinates observed, the goal of matrix completion is to impute the unobserved entries in Y. For example, in collaborative filtering (Fig. 1*A*), matrix completion is used to infer the interests of a user from the interests of other users. A prominent example is the Netflix challenge of inferring movie preferences from sparsely populated matrices of user ratings (1). For virtual drug screening (Fig. 1*B*), matrix completion is used to predict the effect of a drug on a cell type/state given other drug and cell type/state combinations. For image inpainting (Fig. 1*C*) and image reconstruction (Fig. 1*D*), matrix completion is used to restore missing pixels in a corrupted image.

Standard approaches to matrix completion such as nuclear norm minimization (2–4) or deep matrix factorization (5) aim for a completion that yields a low-rank matrix. While such methods can be effective in applications like collaborative filtering, where low rank can capture user similarity, such an objective function can lead to ineffective solutions for applications including drug response imputation, image inpainting, or image reconstruction. For example, in the case of drug response imputation, imputing a new drug would involve predicting the values of an entirely missing vector of gene responses (in contrast to the aforementioned Netflix problem, which involves imputing single scalar entries of the matrix). In this case, a low-rank reconstruction would replace all missing entries with a fixed constant, thereby leading to poor predictive performance. Similarly, for image inpainting and reconstruction, a low-rank completion is generally ineffective since it does not take into account local image structure (6, 7). Thus, there is a need for a more general approach to matrix completion that can easily adapt to the structures in different applications.

In this work, we provide a simple, fast, and flexible framework for matrix completion. To accomplish this, we view matrix completion as an inverse problem; given a matrix $Y \in \mathbb{R}^{m\times n}$ such that a subset of coordinates $S = \{(i,j)\} \subset [m] \times [n]$ are observed and the other entries are missing, we aim to construct $\hat{Y} \in \mathbb{R}^{m\times n}$ such that $\hat{Y}_{i,j} \approx Y_{i,j}$ for all observed coordinates $(i,j) \in S$. We use neural networks to model the observations in Y and use gradient descent to minimize

$$\mathcal{L}(\mathbf{W}) = \sum_{(i,j)\in S} \left(Y_{i,j} - [W_d \phi(W_{d-1}\phi(\dots W_2\phi(W_1 Z)\dots))]_{i,j} \right)^2, \qquad [1]$$

where $\mathbf{W} = \{W_\ell\}_{\ell=1}^{d}$ are the weights of a neural network with each $W_\ell \in \mathbb{R}^{k_{\ell+1}\times k_\ell}$ and $k_{d+1} = m$, $k_1 = p$; $\phi : \mathbb{R} \to \mathbb{R}$ is a fixed element-wise nonlinearity; and $Z \in \mathbb{R}^{p\times n}$ is a

Author contributions: A.R., M.B., and C.U. designed research; A.R., G.S., M.B., and C.U. performed research; A.R., G.S., M.B., and C.U. analyzed data; and A.R., M.B., and C.U. wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.

[1]To whom correspondence may be addressed. Email: cuhler@mit.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2115064119/-/DCSupplemental.

Published April 11, 2022.

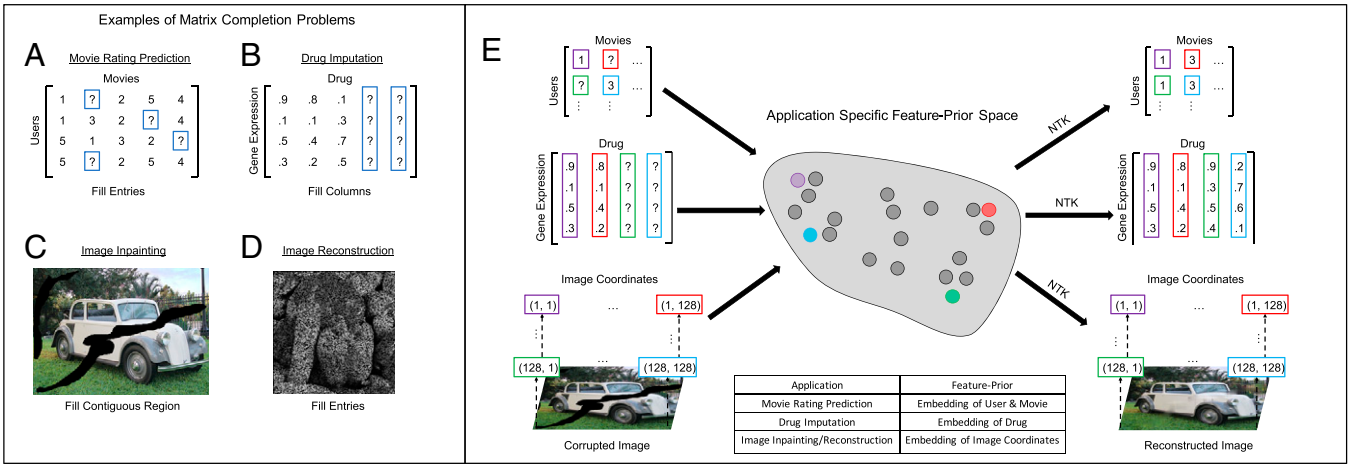


Fig. 1. An overview of matrix completion applications. (*A*) Collaborative filtering example (the Netflix problem), where the goal is to predict how a user would rate (on a scale of 1 to 5) an unseen movie. (*B*) Virtual drug screening, where the problem is to predict the gene expression profile for an unobserved drug/cell type combination. In this application, entire columns are unobserved. (*C* and *D*) Image inpainting and reconstruction involves reconstructing a corrupted region of an image (shown as black pixels). Question marks in *A* and *B* and zero (black) pixels in *C* and *D* represent unobserved entries. (*E*) Our NTK matrix completion framework is easily adapted to solve all of the above problems by selecting a feature prior that represents an embedding of application specific metadata.

fixed application-dependent matrix, which we call the feature prior (described in detail in the section *Flexibility through Feature Prior*). The completed matrix $\hat{Y}$ is then obtained using the forward model with the trained weights, i.e., $\hat{Y} = W_d\phi(W_{d-1}(\dots W_2\phi(W_1 Z)\dots))$. The main contribution of this work is showing that minimizing the loss in Eq. **1** when the width $\{k_\ell\}_{\ell=2}^{d}$ of the neural network tends to infinity gives rise to a simple, fast, and flexible framework for matrix completion suitable for a range of applications.

Superficially, the formulation in Eq. **1** appears similar to that of traditional supervised learning, where a neural network is trained to map data (which would correspond to Z in our formulation) to corresponding labels Y. However, it is important to note that in our formulation, Z can be independent of the observations Y (Z could, for example, be the identity matrix or a random matrix). Thus, Z should be interpreted as a prior that can be chosen in an application-dependent manner. We will discuss the effect of this prior as well as how to choose it for very different applications like virtual drug screening and image inpainting.

Simple and Fast Algorithm for Matrix Completion through Infinite Width Networks

A trend for improving neural network performance is to make models larger (in multiple respects) (8–11). Underscoring this trend, several recent works have empirically demonstrated the advantage of larger (in particular, wider) networks with respect to generalization and performance for classification and representation learning tasks (12–15). There is also an emerging theoretical understanding of the benefit of larger models (16–18). The extreme case where network width approaches infinity is what we consider in this paper in the setting of matrix completion.

While generally larger neural networks require more computational resources for training, quite unintuitively, the limit as network width approaches infinity may yield computational savings. Namely, it was recently shown that training infinite width networks is equivalent to solving kernel regression with a particular kernel known as the neural tangent kernel (NTK) (19). For fully connected networks, the NTK can be computed efficiently in closed form (19), and thus, training an infinite width network reduces to solving a linear system. While this may still be computationally expensive when the number of examples is large, we will use recent preconditioner methods (20–22) to overcome this limitation.

For convolutional networks, no efficient computation of the NTK (the so-called CNTK) has been known (23–25). A major contribution of this work is to provide a memory and runtime efficient algorithm for computing the exact CNTK for matrix completion for a class of practical neural network architectures. As a consequence, our framework can be used to inpaint or reconstruct high-resolution images with hundreds of thousands of pixels. We also provide software for constructing the CNTK as well as precomputed kernels. The simplicity and speed of our framework is exhibited by the fact that most of the results in this work require only a central processing unit (CPU) and can be run efficiently on a laptop.

Flexibility through Feature Prior

The matrix Z in Eq. **1** is key to making our framework easily adaptable to different applications. Unlike traditional supervised learning where the goal is to learn a mapping from data X to labels Y, the matrix Z in our framework can be independent of the observations in Y. We refer to Z as a feature prior since, as we will see, by minimizing the loss in Eq. **1**, the entries of Z encode structure between the coordinates of Y (Fig. 1*E*).

We will demonstrate the flexibility of our framework by using it in two very different applications, namely, for drug response imputation and image inpainting/reconstruction. For drug response imputation, we will select feature priors that encode information about cell and drug type combinations. For image inpainting and reconstruction, we will select feature priors that encode information about image coordinates. In addition to being flexible, we will show that our approach is competitive in terms of speed and accuracy with prior approaches that were specifically developed for drug response imputation (26, 27) or image inpainting/reconstruction (28–30).

Matrix Completion with the NTK

In this section, we derive the NTK for matrix completion when using fully connected networks. Our derivation provides a principled method for selecting the feature prior, Z; namely, we show that Z should be an embedding of coordinate metadata, i.e.,

information describing the coordinates of Y. For example, in drug response imputation, each column of Z could correspond to a different drug, and two columns of Z should be similar if the drug metadata is similar (e.g., the molecular structures are similar). The resulting method is then equivalent to performing semisupervised learning to map from the columns of Z to observed entries in each row of Y. In *Virtual Drug Screening with the NTK*, we utilize this theoretical result to select an effective feature prior for virtual drug screening.

Since the NTK forms the backbone of our framework, we start with the definition of the NTK (19) and briefly review how solving kernel regression with the NTK connects to training infinitely wide neural networks.

Definition (NTK): Let $f(w;x):\mathbb{R}^p\times\mathbb{R}^d\to\mathbb{R}$ denote a neural network with parameters w. The corresponding NTK, $K:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$, is a symmetric, continuous, positive definite function given by

$$K(x,x')=\langle\nabla_w f(w^{(0)};x),\nabla_w f(w^{(0)};x')\rangle,$$

where $w^{(0)}\in\mathbb{R}^p$ are the network parameters at initialization.

For a review of kernel regression and kernel functions, see ref. 31. Given training data $(x^{(i)},y^{(i)})\in\mathbb{R}^d\times\mathbb{R}$ for $i=1,\dots,n$, solving kernel regression with the NTK involves minimizing the loss:

$$\mathcal{L}(\alpha)=\|y-\alpha\hat{K}\|_2^2, \qquad [2]$$

where $\alpha\in\mathbb{R}^{1\times n}$, $y=[y^{(1)},\dots,y^{(n)}]^T$, and $\hat{K}\in\mathbb{R}^{n\times n}$ with $\hat{K}_{i,j}=K(x^{(i)},x^{(j)})$. The work of ref. 19 established that using kernel regression with the NTK is equivalent (under mild assumptions) to training a neural network to map $x^{(i)}$ to $y^{(i)}$ using the mean squared error, in the limit as the network width tends to infinity. Throughout this work, we assume that $w_i^{(0)}\overset{i.i.d}{\sim}\mathcal{N}(0,1)$ and that the nonlinearity ϕ in Eq. **1** is homogeneous (which includes, for example, the rectified linear unit [ReLU], a widely used nonlinearity) so that the NTK corresponding to a fully connected network can be computed efficiently in closed form (19, 32, 33); see *SI Appendix*, Appendix A, for a short review of the relevant literature and notation.

Feature Prior Provides a Flexible Approach for Matrix Completion through Connection with Semisupervised Learning. A natural approach for imputing missing entries in a matrix, Y, is to first obtain an embedding of the coordinates of Y [e.g., a map from coordinates (i,j) to $\mathbb{R}^p$] and then learn a map from the coordinate embedding to the observed entries in Y (e.g., a map from $\mathbb{R}^p$ to $Y_{i,j}\in\mathbb{R}$) (see also ref. 34, chap. 1). For example, for virtual drug screening, one could first embed the drugs based on their molecular properties and then learn a map from this embedding to the measured output, such as gene expression. Such an approach in which a map is learned from an embedding to the observed samples is referred to as semisupervised learning (ref. 35, chap. 15). In this section, we prove that minimizing the loss in Eq. **1** is equivalent to using a semisupervised learning approach for matrix completion. Namely, we show that the columns of Z represent an embedding of the coordinates of Y and that the NTK is used to map from the columns of Z to the entries in Y.

It is a priori unclear how to compute the NTK for matrix completion since this requires training examples and labels. For this, we note the following equivalent formulation of Eq. **1**:

$$\mathcal{L}(\mathbf{W})=\sum_{(i,j)\in S}(Y_{i,j}-\langle f_Z(\mathbf{W}),M_{\{(i,j)\}}\rangle)^2,$$
$$f_Z(\mathbf{W})=W^{(d)}C_d\phi(W^{(d-1)}C_{d-1}\phi(\dots W^{(2)}C_2\phi(W^{(1)}Z))\dots), \qquad [3]$$

where $C_\ell=c/\sqrt{k_\ell}$ for a constant c; $\langle A,B\rangle=tr(A^TB)$ denotes the trace inner product; and $M_{\{(i,j)\}}$ is an indicator matrix, i.e., it has a 1 in the (i,j) entry and zeros everywhere else. To ease notation, we will use M_{ij} to denote the indicator matrix $M_{\{(i,j)\}}$. The formulation in Eq. **3** shows that we can view matrix completion as a problem where the training examples are indicator matrices M_{ij} and the labels are the corresponding entries $Y_{i,j}$. This reformulation yields the following closed form for the NTK for matrix completion, where $\check{\phi}:[-1,1]\to\mathbb{R}$ denotes the dual activation function (36) to ϕ. To keep notation simple, we here provide the theorem when ϕ is the ReLU activation function, but this result holds generally for homogeneous nonlinearities (*SI Appendix*, Appendix B).

Theorem 1. *Assume $Z=\{z^{(i)}\}_{i=1}^n\in\mathbb{R}^{p\times n}$, where each column is normalized with $\|z^{(i)}\|_2=1$. Let $f_Z(\mathbf{W})$ be a d layer fully connected network with nonlinearity $\phi(x)=\max(x,0)$ and $c=\sqrt{2}$ in Eq. **3**. Then, as widths $k_2,k_3,\dots,k_d\to\infty$, the NTK for matrix completion with $f_Z(\mathbf{W})$ is given by*

$$K(M_{ij},M_{i'j'})=\begin{cases}\kappa_d({z^{(j)}}^T z^{(j')}) & \text{if } i=i'\\ 0 & \text{if } i\neq i'\end{cases},$$

where $\kappa_d(\xi)=\check{\phi}^{(d)}(\xi)+\kappa_{d-1}(\xi)\frac{d\check{\phi}}{d\xi}(\check{\phi}^{(d-1)}(\xi))$, and $\check{\phi}^{(h)}(\xi)=\check{\phi}(\check{\phi}^{(h-1)}(\xi))$ for $h\geq 1$ and $\check{\phi}^{(0)}(\xi)=\xi$.

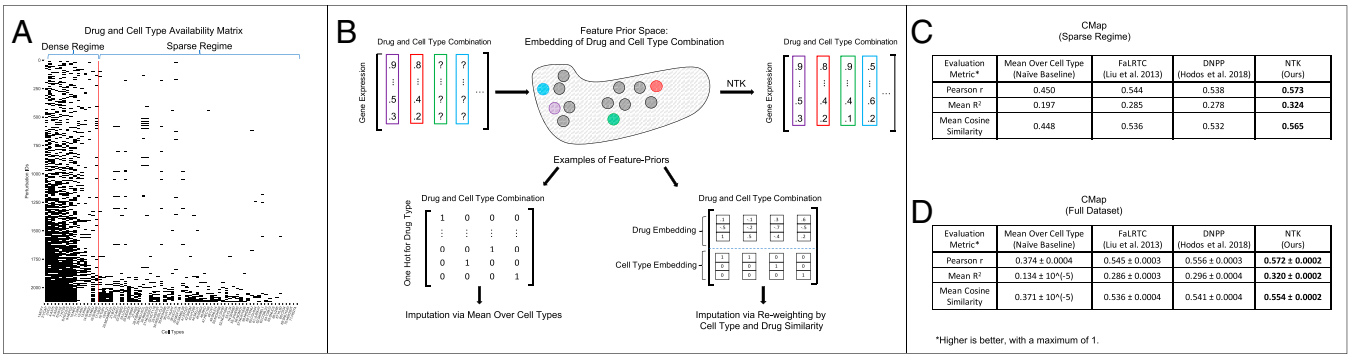
The proof as well as an example showing how *Theorem 1* can be used in practice to compute the NTK for matrix completion is presented in *SI Appendix*, Appendix B. Since the kernel value between M_{ij} and $M_{i'j'}$ is a function of columns j and j' of Z, *Theorem 1* implies that the NTK for matrix completion maps columns of Z to entries $Y_{i,j}$, and thus, the columns of Z encode structure between the coordinates of Y.

By varying the nonlinearity ϕ, depth d, and feature prior Z, our framework encapsulates a variety of semisupervised learning approaches. To provide a nontrivial example, we prove in *SI Appendix*, Appendix B, that our framework for matrix completion generalizes Laplacian-based semisupervised learning (37). This insight regarding the connection between our framework for matrix completion and semisupervised learning represents the backbone for a simple and competitive approach to virtual drug screening described in *Virtual Drug Screening with the NTK*.

Virtual Drug Screening with the NTK

The Connectivity Map (CMAP) is a prominent, large-scale, publicly available drug screen that considers 20,413 different compounds and 72 different cell lines (38). Experiments in CMAP were performed on a subset of 201,484 drug/cell line pairs; for each of these pairs the gene expression profile of 978 landmark genes was measured. CMAP has been an important resource for computational approaches to drug discovery and drug repurposing (38–40). In these applications, the goal is to use a subset of observed drug/cell type pairs to predict the gene expression profile of new drug/cell type pairs. These profiles are then used to identify drug candidates of interest that can be tested experimentally (41, 42).

The CMAP dataset can be viewed as a three-dimensional tensor (drugs, cell lines, and genes), where many of the entries are



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**

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**

Fig. 2. Our infinite width neural network framework outperforms DNPP (26), FaLRTC (27), and mean over cell types for drug response imputation on CMAP. (*A*) We visualize the availability of cell type and drug combinations of the subset from ref. 26. (*B*) Our method corresponds to first providing an embedding of cell type and drug combinations as the feature prior and then applying the NTK. We show that 1) using a feature prior consisting of one-hot vectors for drugs corresponds to imputation by performing mean across observations for each cell type and 2) using a feature prior that captures similarity between drugs and cell types is effective for imputation. (*C* and *D*) Our infinite width neural network framework (denoted NTK) outperforms DNPP and mean over cell type across three evaluation metrics. We use five rounds of 10-fold cross-validation to determine that the difference between our method and the next best method, DNPP, is statistically significant ($P < 10^{-20}$).

missing. In the following, we use the same preprocessing of the data as in ref. 26 to filter out drug/cell line combinations with very few or inconsistent samples; a description and a link to the dataset is provided in *SI Appendix*, Appendix C. The resulting drug/cell line combinations are shown in Fig. 2*A*. The three-dimensional tensor can be flattened into a matrix, where the columns correspond to drug/cell line combinations and the rows represent genes (Fig. 2*B*); i.e., following the notation from *Virtual Drug Screening with the NTK*, entry Y_{ij} of the resulting flattened matrix is a real-valued number quantifying the gene expression of gene i in drug and cell type combination j. This matrix has a missing column for every missing drug/cell line combination. Classical low-rank matrix factorization methods would prove ineffective in this setting since they would replace each missing column by the same constant column. On the other hand, *Theorem 1* suggests the NTK as an effective way for imputing the missing gene expression profiles by selecting the feature prior Z such that two columns of Z are similar if they correspond to similar drug/cell line pairs. In the following, we discuss three different feature priors for this application; for a full description of these priors, see *SI Appendix*, Appendix D.

Feature Prior Corresponding to the Mean over Cell Type Baseline. A simple baseline is to impute the gene expression profiles for each missing drug for a given cell line by the mean over all observed drugs for this cell line. Quite surprisingly, this simple approach gives rise to a strong baseline (26, 43) since cell type is the dominant factor, while drugs have subtle effects on gene expression.

While it is generally nontrivial to improve upon this simple baseline without constructing a specialized algorithm (26, 44–46), our NTK framework provides an easy way for doing so. In particular, our framework makes it evident that the feature prior corresponding to the mean over cell type baseline is trivial since it corresponds to an embedding in which drugs are encoded via one-hot vectors (*SI Appendix*, Appendix E). Thus, to improve upon this baseline, we select any feature prior that can capture similarities between drugs.

Feature Prior Corresponding to Previous Algorithms. We now demonstrate that our framework provides a direct approach to improve on previous methods for virtual drug screening by using the output of previous methods as a feature prior in our framework. Namely, if a method is used to produce an imputation, $\hat{Y}$, then the columns in $\hat{Y}$ should represent an embedding of drug and cell type combinations that captures their similarity. Hence, we can use $Z = \hat{Y}$ as the feature prior in our method. For illustration, we apply this approach to two state-of-the-art methods for virtual drug screening: 1) drug neighbor profile prediction (DNPP) (26), which is a weighted nearest neighbor scheme, and 2) fast low-rank tensor completion (FaLRTC) (27), which involves low-rank matrix completion along each slice of the CMAP tensor. We show that our framework using these feature priors yields an improvement over the individual methods (*SI Appendix*, Appendix F).

Proposed Feature Prior for Drug Response Imputation. Observing the pattern of data availability in Fig. 2*A*, it is apparent that a subset of cell lines have observations for many (>150) drugs (dense regime), while many cell lines have observations for only a few (≤150) drugs (sparse regime). While previous methods such as DNPP are quite effective in the dense regime, they are not as effective in the sparse regime (Fig. 2*C* and *SI Appendix*, Appendix G). This can be explained by the fact that in the sparse regime, DNPP roughly imputes using the simple mean over cell type baseline.

For effective drug response imputation in the sparse regime, our framework can be used to construct a simple feature prior by concatenating embeddings for cell types and drugs. In particular, we can use the gene expression values for a reference cell type for which there are a lot of drug observations (e.g., MCF7 in CMAP) as the embedding of drugs and the mean gene expression across all observations for a given cell type as the embedding of cell type. Fig. 2*C* shows that the NTK with this simple feature prior outperforms mean over cell type, FaLRTC, and DNPP in the sparse regime. We compare across Pearson's r value, mean R^2, and mean cosine similarity. A description of all evaluation metrics is provided in *SI Appendix*, Appendix H. By combining our feature prior for the sparse regime with the FaLRTC-based feature prior for the dense regime, we obtain a drug imputation method that significantly outperforms DNPP, FaLRTC, and mean over cell type on the full dataset (Fig. 2*D*) ($P < 10^{-20}$ based on five rounds of 10-fold cross validation, with an improvement on every fold of every round across all metrics; *SI Appendix*, Appendix I).

Matrix Completion with the Convolutional NTK

While we have thus far derived and applied the NTK for matrix completion using fully connected networks, these architectures are not nearly as effective as convolutional networks for matrix

completion tasks in which the target matrix is an image. Similar to the case of fully connected networks, a closed form for the NTK corresponding to convolutional networks (the so-called CNTK) is known in the regression setting (23), but it has not been considered in the setting of matrix completion. Moreover, the runtime for computing the CNTK for regression scales quadratically with each image dimension. In this section, we derive the CNTK for matrix completion and provide a computationally efficient method for computing the CNTK for matrix completion for a class of feature priors that are effective for image inpainting and reconstruction.

We begin by deriving the CNTK for matrix completion for a simple class of convolutional networks, when there are no downsampling or upsampling layers. We show that in this setting, the CNTK for matrix completion can be computed using terms from the CNTK for classification. In the following proposition (proof in *SI Appendix*, Appendix J), $\Theta^{(d)} \in \mathbb{R}^{m\times n\times m\times n}$ denotes the tensor corresponding to the CNTK of a d layer convolutional network in the classification setting (ref. 23, section 4).

Proposition 1. *Let $f_Z(\mathbf{W})$ be a d layer convolutional network used to map from feature prior, $Z \in \mathbb{R}^{c\times m\times n}$, to the target matrix, $Y \in \mathbb{R}^{m\times n}$. Then as the number of convolutional filters per layer approaches infinity, the CNTK of $f_Z(\mathbf{W})$ is given by*

$$K(M_{ij}, M_{i'j'}) = [\Theta^{(d)}(Z, Z)]_{i,j,i',j'}, \qquad [4]$$

where $M_{ij}, M_{i'j'} \in \mathbb{R}^{m\times n}$ denote indicator matrices.

CNTK Performs Semisupervised Learning Using Image Coordinate Features. In *Matrix Completion with the NTK*, we established a connection between semisupervised learning and matrix completion using the NTK. We now establish a similar connection between semisupervised learning and matrix completion with the CNTK for a class of feature priors defined in *Theorem 2*. This class includes feature priors that are heavily used in image inpainting applications, namely, where the channels of Z are drawn independently and identically distributed (i.i.d.) from a stationary distribution (24, 30). The following theorem (proof in *SI Appendix*, Appendix K), which is analogous to *Theorem 1* for the NTK, implies that using the CNTK for matrix completion is equivalent to mapping from coordinate features to observed entries in the target matrix Y.

Theorem 2. *Consider a convolutional network of depth d with homogeneous activation and in which all filters have size q and circular padding. Let $Z \in \mathbb{R}^{c\times m\times n}$ satisfy*

$$\sum_{\ell=1}^{c} \sum_{-\alpha\le a,b\le\alpha} Z_{\ell,i+a,j+b} Z_{\ell,i'+a,j'+b} = \psi(|i-i'|, |j-j'|)$$

for some $\psi : \mathbb{R}^2 \to \mathbb{R}$ with maximum at $(0,0)$ and $\alpha = \frac{q-1}{2}$ (odd q). Then as the number of convolutional filters per layer goes to infinity, the CNTK simplifies to

$$K(M_{ij}, M_{i'j'}) = \tilde{\psi}(|i-i'|, |j-j'|),$$

where $\tilde{\psi} : \mathbb{R}^2 \to \mathbb{R}$ is a function that can be computed from ψ (a recursive formula is provided in SI Appendix, Appendix K).

Since the function $\tilde{\psi}$ depends only on the positions of the coordinates, *Theorem 2* shows that the CNTK for matrix completion is equivalent to semisupervised learning using kernels on features corresponding to coordinates.

Closed Form for the CNTK of Modern Architectures for Matrix Completion. Unlike the convolutional networks considered thus far, state-of-the-art architectures for unsupervised image inpainting such as refs. 24, 30 incorporate a variety of layer structures including strided convolution, nearest neighbor and bilinear upsampling, skip connections, and batch normalization. We derive (in *SI Appendix*, Appendix L) the CNTK for matrix completion using convolutional networks with the following layer structures: 1) downsampling through strided convolution, 2) nearest neighbor upsampling, and 3) bilinear upsampling.*

Efficient Computation of the CNTK of Modern Architectures for Matrix Completion. A key insight that we use to speed up the computation of the CNTK is that the kernel in Eq. **4** depends only on the feature prior and not on the values of the observed pixels in an image. Hence, the CNTK need only be computed once for all images of a given resolution. This enables a drastic speedup over recomputing the kernel for every new image, as is currently required in classification.

However, using such a direct approach to compute the CNTK is still computationally prohibitive for high-resolution images. In particular, computing the CNTK for a network with d convolutional layers to complete an image of size $2^p \times 2^q$ requires $O(p^2q^2d)$ runtime and $O(2^{2p+2q})$ space. In order to overcome these limitations, prior work (25) used the Nyström method (47) to approximate the kernel. Instead of relying on such approximations, we here present an algorithm for computing the exact CNTK in a memory and runtime efficient manner for any convolutional neural network with circular padding, strided convolution, and nearest neighbor upsampling layers, when using a feature prior with i.i.d. random entries. Such networks and feature priors are heavily used for image completion tasks (30).

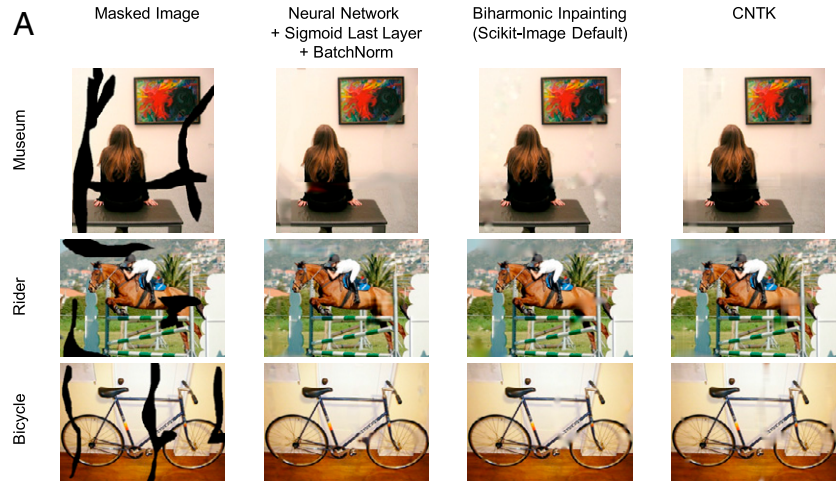
Our main insight that enables such an algorithm is that for convolutional networks with strided convolution and nearest neighbor upsampling layers, the CNTK for low-resolution images can be expanded to high-resolution images for any feature prior with i.i.d. random entries. In particular, if a neural network with s downsampling and upsampling layers is used to inpaint images of resolution $2^p \times 2^q$, our algorithm requires only an array of size 2^{2s+p+q}, while storing the full CNTK requires an array of size 2^{2p+2q}. In practice, s is exponentially smaller than p, q, and so our method is significantly more memory efficient; see the following specific example. In addition, since our method only requires computing the CNTK for images of size $2^{s+1} \times 2^{s+1}$, the runtime of our method is $O(2^{4s})$ instead of $O(2^{2p+2q})$, and thus, our method is significantly faster than a direct computation. A detailed description and proof of our expansion algorithm is presented in *SI Appendix*, Appendix M.

Example. Let $f_Z(\mathbf{W})$ represent a convolutional neural network with circular padding, three layers of strided convolution with a stride size of 2 in each direction, and three nearest neighbor upsampling layers with a feature prior $Z \in \mathbb{R}^{c\times 512\times 512}$ satisfying

$$\sum_{p=1}^{c} Z_{p,i,j} Z_{p,i',j'} = \begin{cases} C_1 & i = i',\ j = j' \\ C_2 & \text{otherwise} \end{cases},$$

where $C_1, C_2 > 0$ are constants. Suppose $f_Z(\mathbf{W})$ is used to inpaint images of size 512×512. Then, by computing the CNTK for 16×16 resolution images, $K_\ell \in \mathbb{R}^{16^2\times 16^2}$, we can expand up to the exact CNTK for 512×512 images. Computing K_ℓ

*The impact of linear downsampling and upsampling on the CNTK is briefly described in appendix F of ref. 25, but the explicit forms are not computed nor used in the experiments.



B

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

Fig. 3. Large hole inpainting using 1) the CNTK, 2) neural networks with sigmoid last layer and batch normalization layers that are trained with Adam, and 3) biharmonic functions. (*A*) Qualitative comparison of inpainting results across the three methods. Results for all images are provided in *SI Appendix*, Fig. S5. (*B*) Comparison of PSNR across three methods with the CNTK providing the highest average PSNR. Runtime and SSIM for the three methods are provided in *SI Appendix*, Fig. S4.

takes roughly 11 s when using a CPU with 1 thread, and $\tilde{K}$ uses less than 100 MB of memory with floating point precision. On the other hand, even storing the true kernel $K \in \mathbb{R}^{512^2 \times 512^2}$ would require roughly 256 GB memory when using floating point precision. This is twice the amount of random-access memory (RAM) available on our server and 16 times the amount of RAM available on most laptops.

Image Inpainting and Reconstruction with the CNTK

We now utilize the results of *Matrix Completion with the Convolutional NTK* to perform large hole image inpainting and reconstruction. As illustrated in Fig. 1 *C* and *D*, large hole inpainting involves imputing a large contiguous region in an image, while image reconstruction involves imputing random missing pixels in an image. Recent work (30) demonstrated that using convolutional neural networks with downsampling and upsampling layers to impute the missing pixels in images leads to competitive results for these applications.

The methods from ref. 30 are a special case of our framework in Eq. **1**, namely, using convolutional layers and letting the feature prior, Z, be a tensor with i.i.d. uniform random entries. Thus, we can use our framework for performing image completion tasks, and instead of training deep networks, we can simply solve kernel regression with the CNTK. We will demonstrate that this gives rise to a simple, fast, flexible, and competitive alternative to training deep networks for high-resolution image completion problems. Moreover, we will demonstrate that our framework can be used to identify the role of architecture and feature prior on image completion problems and aid in identifying effective architectures and feature priors.

Application 1: Large Hole Inpainting with the CNTK. We utilize the CNTK for large hole inpainting tasks from refs. 24, 30. We compute the CNTK for the architecture used in ref. 24 with six downsampling and nearest neighbor upsampling layers for the feature prior Z with i.i.d. entries $Z_{\ell,i,j} \sim U[0,.1]$, where $\ell \in \mathbb{Z}_+$ and $i,j \in [m] \times [n]$. We compute the CNTK on 128×128 resolution images and then expand it to the CNTK for high-resolution images via our expansion technique in *Matrix Completion with the Convolutional NTK*. We compare our method against neural networks of the same architecture using the training procedures from refs. 24, 30 (see *SI Appendix*, Appendix N, for details). We also compare our method against inpainting with biharmonic functions (28), which is currently the default inpainting method in scikit-image (29).

Fig. 3*A* shows examples of the resulting reconstructions, and Fig. 3*B* shows the peak signal-to-noise ratio (PSNR) across all methods. Our method on average outperforms both inpainting with finite width neural networks and inpainting with biharmonic functions.[†] In *SI Appendix*, Fig. S4, we show that our method also outperforms the other methods in terms of structural similarity index measure (SSIM) and that the runtime is comparable (within 2 min on average) across all methods in this setting. The reconstructions across all images and methods are provided in *SI Appendix*, Fig. S5.

Application 2: Image Reconstruction with the CNTK. We next analyze the performance of the CNTK on the image reconstruction tasks considered in (30). While the networks considered in refs. 24, 30 make use of skip connections for image reconstruction, we only consider architectures without skip connections for which we can derive the CNTK exactly (see *SI Appendix*, Appendix N, for details). We again compare the CNTK to neural networks of the same architecture and to biharmonic inpainting. For this comparison, we use networks with 128 filters per layer, as is done in refs. 24, 30. In *SI Appendix*, Fig. S6, we show that our model performs comparably to inpainting with biharmonic functions and outperforms neural networks of the same architecture. In *SI Appendix*, Fig. S6, we additionally show that our method performs comparably to biharmonic inpainting in terms of SSIM and that our method is up to 10 times faster than using small width neural networks on the same hardware. While our method performs comparably to inpainting with biharmonic functions in this application, our framework is more flexible since we can adjust architecture and feature prior, and it outperforms inpainting with biharmonic functions for the problem of large hole inpainting (see *Application 1: Large Hole Inpainting with the CNTK*). Since methods such as Adam with Langevin dynamics (24) have enabled performance boosts for neural networks (*SI Appendix*, Figs. S4 and S5), an interesting direction for future work could be to incorporate such techniques for image completion applications using the CNTK.

[†]While the PSNR values for these images are also presented in ref. 24, they appear to be computed without replacement of the observed pixel values. We reran these experiments with replacement for fair comparison with biharmonic inpainting.

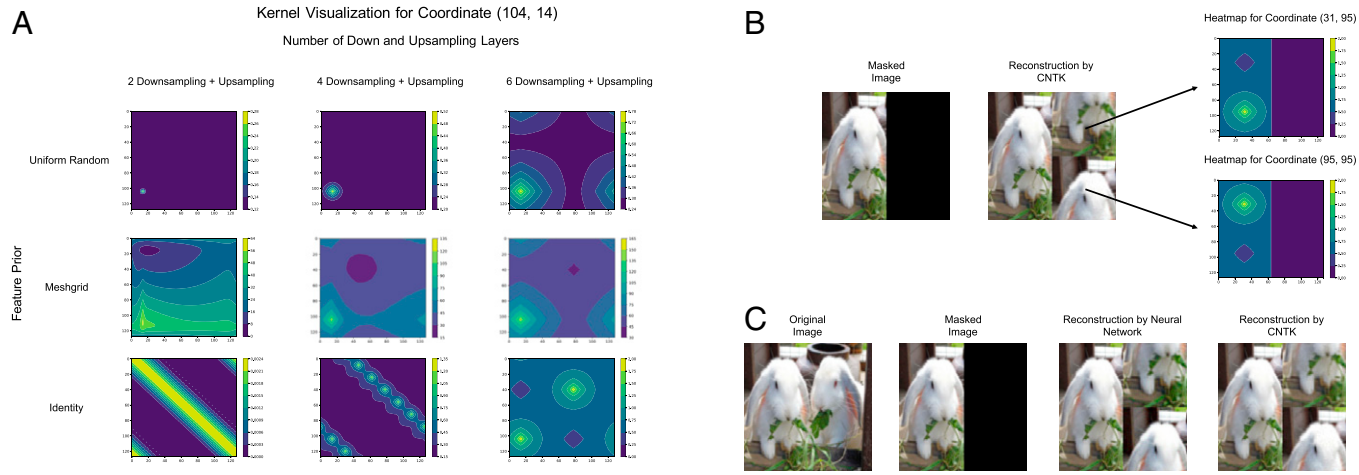


Fig. 4. We use the CNTK to understand the impact of architecture and input on image inpainting. (*A*) Heat map visualizations of the CNTK when varying the number of downsampling/upsampling layers and input. The visualization makes clear that the uniform random feature prior, unlike other feature priors, results in kernels that use the region surrounding a missing pixel value for imputation regardless of the number of downsampling layers. (*B*) The heat map visualizations of the CNTK make transparent which observed pixels are being used to inpaint a given missing pixel when using the identity feature prior. (*C*) A comparison between inpainting a 128 × 128 resolution image of a rabbit with a finite width neural network and with the CNTK when the feature prior is the identity. The CNTK is able to accurately predict the unexpected behavior of the neural network.

Using Our Framework to Select Feature Prior and Architecture for Image Completion. In the following, we demonstrate that our framework provides a theoretical underpinning for understanding how a given architecture and feature prior influence image completion. In particular, we use our framework to explain why the uniform random feature prior and architectures with downsampling and upsampling layers are effective for image completion while other feature priors such as the identity feature prior are ineffective for this application.

The key observation enabling such interpretability is that for kernel methods, every prediction (a missing pixel value) is a linear combination of training examples (observed pixel values). Hence, for each imputed pixel, the CNTK can be used to provide a heat map describing which observed pixels were most heavily weighted in the linear combination. In order to generate such heat maps, we reshape the CNTK into a four-dimensional tensor. Namely, given a CNTK $K \in \mathbb{R}^{mn \times mn}$, we reshape K to a tensor $K_T \in \mathbb{R}^{m \times n \times m \times n}$ where $K(M_{ij}, M_{i'j'}) = K_T(i, j, i', j')$. To generate a heat map for a given a coordinate (i, j), we visualize the matrix $K_T(i, j, :, :) \in \mathbb{R}^{m \times n}$. This visualization allows us to decipher how architecture and feature prior change the resulting imputation from a neural network.

The Uniform Random Feature Prior and Modern Architectures Are Effective for Image Completion. In Fig. 4*A*, we visualize the kernel values $K(104, 14, :, :)$ computed for a 128 × 128 image when varying the number of down and upsampling layers and as well as the feature prior Z. Namely, we consider the cases where Z is the identity, the mesh grid from ref. 30, or the uniform random tensor used in large hole inpainting experiments of ref. 30. A key observation is that the kernel values for the uniform random feature prior are highest around the coordinate of interest regardless of the amount of down and upsampling, which is in stark contrast to other feature priors.[‡] This implies that neighboring pixels are most heavily used when imputing using the uniform random feature prior (see *SI Appendix*, Fig. S7, for additional visualizations). Moreover, when using the uniform random feature prior, the amount of downsampling and upsampling increase (by powers of 2) the size of the region considered for imputation (see the first row of Fig. 4*A*). These heat maps identify the minimum amount of downsampling necessary for large hole inpainting: if there is an $m \times m$ region of missing pixels ($m \geq 1$), we need least $\lfloor \log_2(m+1) \rfloor$ layers of downsampling to ensure that no pixel is filled in as an average of all other pixels. This result explains the observation from ref. 30, which showed that using neural networks with four or fewer downsampling and upsampling layers led to worse large hole inpainting performance on images with large missing regions.

The Identity Feature Prior Is Ineffective for Image Completion. The standard feature prior for matrix completion is given by choosing Z to be the identity matrix (3, 5, 48). As shown in Fig. 4*A*, unlike the uniform random feature prior, the identity feature prior uses pixel observations from nonlocal regions for completion. Thus, we expect this feature prior to be ineffective for image completion tasks.

Fig. 4*B* shows the result of using the CNTK for a network with six downsampling and upsampling layers and the identity feature prior to impute a 128 × 128 rabbit image. The identity feature prior visually appears to translate observed pixels from a nonlocal region to perform imputation. The regions that are being translated are precisely those given by the corresponding heat maps; e.g., the upper right quadrant is imputed using the lower left quadrant in Fig. 4*B*.

We note that our framework accurately predicts the behavior of finite width neural networks used for image inpainting. In Fig. 4*C*, we show the result of using a neural network with six downsampling and upsampling layers, sigmoid activation on the last layer, and identity feature prior. We observe that the neural network completes the image by translating observed pixels similarly to the imputation provided by the corresponding CNTK. This example highlights the power of using our framework for rapidly prototyping feature priors and architectures for image inpainting tasks.

Discussion

In this work, we presented a simple, fast, and flexible framework for matrix completion using the infinite width limit of neural networks, i.e., the NTK. Below, we highlight the aspects of our framework that enable such simplicity, speed, and flexibility.

[‡]When there are no downsampling and upsampling layers, this follows immediately from *Theorem 2*.

- Our framework is conceptually simple since we are using kernels to learn a map from features of coordinates, (i, j), to entries in the target matrix, $Y_{i,j}$. Our framework is computationally simple since solving kernel regression involves solving a linear system of equations.
- Our framework is naturally fast when using the NTK of fully connected networks for matrix completion due to the simple closed form of the kernel (*Theorem 1*). We develop a memory and runtime efficient algorithm to compute and use the NTK of convolutional networks (the CNTK) for matrix completion (*Matrix Completion with the Convolutional NTK*).
- Our framework is easily adapted to various applications by the choice of the feature prior, thereby making our framework flexible. Moreover, we provided a principled approach for selecting the feature prior by establishing a connection with semisupervised learning (*Theorems 1* and *2*) and providing a visualization of the effect of the feature prior (*Image Inpainting and Reconstruction with the CNTK*).

The simplicity and speed of our framework is illustrated by the fact that many of our results (including inpainting high-resolution images) can be run on a CPU and even on a laptop (see *Materials and Methods* for a link to our code). We demonstrated that our framework is flexible by using it to achieve competitive results for virtual drug screening (*Virtual Drug Screening with the NTK*) and image inpainting/reconstruction (*Image Inpainting and Reconstruction with the CNTK*). We envision that our work provides a simple and accessible framework for producing strong baselines for several matrix completion applications. We conclude with a discussion of possible future extensions and applications.

Future Applications of Our Framework. In this work, we demonstrated the flexibility of our framework by constructing feature priors for two different applications, namely, virtual drug screening and image completion. An interesting future direction is the extension of our framework to other modalities such as tensors, video, or audio data. For example, by using a feature prior that captures the structure of coordinates in three-dimensional images, we could apply our framework to impute missing regions in three-dimensional data.

Efficient Computation of the CNTK. In classification and regression settings, a major hindrance for using the CNTK in practice is the computational complexity in computing the kernel for a large image dataset. In this work, we presented an expansion technique to efficiently compute and store the exact CNTK for inpainting high-resolution images, which was previously considered infeasible (24, 25). By understanding the properties of the CNTK that make it effective for image problems, we envision that similar techniques could be applied to produce efficient kernel machines for image classification.

Developing Techniques to Improve the Performance of the NTK. While a large number of techniques such as skip connections, batch normalization, etc., have been developed to augment the performance of neural networks, such techniques have yet to be adapted to improve the performance of kernels. The simplicity and effectiveness of the NTK and CNTK based on simple architectures considered in this work motivates the development of techniques to further boost the performance of the NTK and kernel methods in general.

Materials and Methods

For solving kernel regression with the NTK, we use the direct linear system solver from ref. 49 when the number of equations is fewer than 30,000, and we use EigenPro (20, 22) otherwise. For training neural networks, we use the PyTorch library (50). All methods requiring a graphics processing unit (GPU) are run on a single NVIDIA Titan RTX GPU. Our experiments are run on a shared server with 4 Titan RTX GPUs, 128 GB CPU RAM, and 64 threads.

For the virtual drug screening experiments, we use the subset of the CMAP dataset (38) provided in ref. 26. A detailed description of all the methods (including random seeds and hyperparameters for DNPP and FaLRTC) and evaluation metrics for the virtual drug screening experiments is provided in *SI Appendix*, Appendixes C–H. A description of the t test used for determining the significance of our results for virtual drug screening is presented in *SI Appendix*, Appendix I. We provide code to replicate our results for the virtual drug screening experiments with the NTK, DNPP, FaLRTC, and mean over cell type at https://github.com/uhlerlab/ntk_matrix_completion. We use the codebase from ref. 26 for performing imputation with FaLRTC.

For the image completion applications, we use the datasets from refs. 24, 30. The rabbit image used in Fig. 4 is from ref. 51 and is provided in our codebase (linked above). For the neural network and NTK methods used in our image inpainting and reconstruction experiments, we provide a description of all architectures and training hyperparameters in *SI Appendix*, Appendix N.

We provide a library for computing and using the CNTK for image inpainting and reconstruction applications in the codebase linked above. Our library lets the user define a custom neural network (similarly to network definitions in PyTorch) and then provides a function to compute the CNTK from the given architecture. Our method for computing the CNTK runs entirely on the CPU, and we enable parallelization across CPU threads. Our library includes functions for computing the CNTK for networks with nearest neighbor and bilinear upsampling layers, which are not readily available in the Neural Tangents library (52). We additionally provide functions to solve kernel regression using the CNTK via a linear system solver or EigenPro. A full description of the library and an example of how to use our library for image inpainting is provided in Jupyter notebooks in our linked code. We additionally release several precomputed kernels that can be used for high-resolution inpainting and reconstruction.

Data Availability. Code data have been deposited in GitHub (https://github.com/uhlerlab/ntk_matrix_completion). All other study data are included in the article and/or *SI Appendix*.

ACKNOWLEDGMENTS. A.R., G.S., and C.U. were partially supported by NSF (grant DMS-1651995), Office of Naval Research (grants N00014-17-1-2147 and N00014-18-1-2765), the MIT-IBM Watson AI Lab, the Eric and Wendy Schmidt Center at the Broad Institute, and a Simons Investigator Award (to C.U.). M.B. acknowledges support from NSF grant IIS-1815697 and NSF grant DMS-2031883/Simons Foundation Award 814639.

Author affiliations: [a]Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139; [b]Institute for Data, Systems, and Society, Massachusetts Institute of Technology, Cambridge, MA 02142; [c]Halıcıoğlu Data Science Institute, University of California San Diego, La Jolla, CA 92093; and [d]Broad Institute of Massachusetts Institute of Technology and Harvard, Cambridge, MA 02142

1. Netflix Prize Rules (2009). https://web.archive.org/web/20211017094107/https://www.netflixprize.com/assets/rules.pdf. Accessed 22 January 2022.
2. B. Recht, M. Fazel, P. A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Soc. for Ind. Appl. Math. Rev.* **52**, 471–501 (2010).
3. E. Candès, B. Recht, Exact matrix completion via convex optimization. *Commun. Assoc. Comput. Mach* **55**, 111–119 (2012).
4. E. J. Candès, T. Tao, The power of convex relaxation: Near-optimal matrix completion. *Inst. Electr. Electron. Eng. Trans. Inf. Theory* **56**, 2053–2080 (2010).
5. S. Arora, N. Cohen, W. Hu, Y. Luo, "Implicit regularization in deep matrix factorization" in *Advances in Neural Information Processing Systems*, H. Wallach *et al.*, Eds. (Curran Associates, Inc., 2019), pp. 7413–7424.
6. Z. Li, Z. Q. J. Xu, T. Luo, H. Wang, A regularized deep matrix factorized model of matrix completion for image restoration. arXiv [Preprint] (2020). https://arxiv.org/abs/2007.14581. Accessed 15 August 2021.
7. H. Xue, S. Zhang, D. Cai, Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *Inst. Electr. Electron. Eng. IEEE Trans. Image Process.* **26**, 4311–4320 (2017).

8. J. Kaplan *et al.*, Scaling laws for neural language models. arXiv [Preprint] (2020). https://arxiv.org/abs/2001.08361. Accessed 15 August 2021.
9. K. He, X. Zhang, S. Ren, J. Sun, *Deep Residual Learning for Image Recognition in Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, 2016).
10. O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional networks for biomedical image segmentation" in *International Conference on Medical Image Computing and Computer Assisted Intervention*, N. Navab, J. Hornegger, W. Wells, A. Frangi, Eds. (Springer, 2015), pp. 234–241.
11. S. Zagoruyko, N. Komodakis, "Wide residual networks" in *Proceedings of the British Machine Vision Conference*, R. C. Wilson, E. R. Hancock, W. A. P. Smith, Eds. (British Machine Vision Association Press, 2016), pp. 87.1–87.12.
12. C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, "Understanding deep learning requires rethinking generalization" in *International Conference on Learning Representations*. (2017).
13. M. Belkin, D. Hsu, S. Ma, S. Mandal, Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 15849–15854 (2019).
14. P. Nakkiran *et al.*, "Deep double descent: Where bigger models and more data hurt" in *International Conference in Learning Representations* (2020).
15. A. Radhakrishnan, M. Belkin, C. Uhler, Overparameterized neural networks implement associative memory. *Proc. Natl. Acad. Sci. U.S.A.* **117**, 27162–27170 (2020).
16. M. Belkin, D. Hsu, J. Xu, Two models of double descent for weak features. *Soc. Ind. Appl. Math. J. Math. Data Sci.* **2**, 1167–1180 (2020).
17. T. Hastie, A. Montanari, S. Rosset, R. J. Tibshirani, Surprises in high-dimensional ridgeless least squares interpolation. arXiv [Preprint] (2019). https://arxiv.org/abs/1903.08560. Accessed 15 August 2021.
18. P. L. Bartlett, P. M. Long, G. Lugosi, A. Tsigler, Benign overfitting in linear regression. *Proc. Natl. Acad. Sci. U.S.A.* **117**, 30063–30070 (2020).
19. A. Jacot, F. Gabriel, C. Hongler, "Neural Tangent Kernel: Convergence and generalization in neural networks" in *Advances in Neural Information Processing Systems*, S. Bengio *et al.*, Eds. (Curran Associates, Inc., 2018), pp. 8580–8589.
20. S. Ma, M. Belkin, "Diving into the shallows: A computational perspective on large-scale shallow learning" in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. (Curran Associates, Inc., 2017), pp. 3781–3790.
21. G. Meanti, L. Carratino, L. Rosasco, A. Rudi, "Kernel methods through the roof: Handling billions of points efficiently" in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin, Eds. (Curran Associates, Inc., 2020), pp. 14410–14422.
22. S. Ma, M. Belkin, "Kernel machines that adapt to GPUs for effective large batch training" in *Conference on Machine Learning and Systems*, A. Talwalkar, V. Smith, M. Zaharia, Eds. (JMLR, Inc. and Microtome Publishing, 2019), pp. 360–373.
23. S. Arora *et al.*, "On exact computation with an infinitely wide neural net" in *Advances in Neural Information Processing Systems*, H. Wallach *et al.*, Eds. (Curran Associates, Inc., 2019), pp. 8141–8150.
24. Z. Cheng, M. Gadelha, S. Maji, D. Sheldon, "A Bayesian perspective on the deep image prior" in *Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, 2019), pp. 5438–5446.
25. J. Tachella, J. Tang, M. Davies, The neural tangent link between CNN denoisers and non-local filters. arXiv [Preprint] (2020). https://arxiv.org/abs/2006.02379. Accessed 15 August 2021.
26. R. Hodos *et al.*, Cell-specific prediction and application of drug-induced gene expression profiles. *Pac. Symp. Biocomput.* **23**, 32–43 (2018).
27. J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data. *Inst. Electr. Electron. Eng. Trans. Pattern Analysis Mach. Intell.* **35**, 208–220 (2013).
28. S. B. Damelin, N. S. Hoang, On surface completion and image inpainting by biharmonic functions: Numerical aspects. *Int. J. Math. Math. Sci.* **2018**, 3950312 (2018).
29. S. van der Walt *et al.*; scikit-image contributors, scikit-image: Image processing in Python. *PeerJ* **2**, e453 (2014).
30. D. Ulyanov, A. Vedaldi, V. Lempitsky, "Deep image prior" in *Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, 2018), pp. 9446–9454.
31. B. Scholkopf, A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Cambridge, MA, 2001).
32. Y. Cho, L. Saul, "Kernel methods for deep learning" in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, A. Culotta, Eds. (Curran Associates, Inc., 2009), pp. 342–350.
33. R. Tsuchida, F. Roosta-Khorasani, M. Gallagher, "Invariance of weight distributions in rectified MLPs" in *International Conference on Machine Learning*, J. Dy, A. Krause, Eds. (Proceedings of Machine Learning Research, 2018) vol. 80, pp. 4995–5004.
34. C. C. Aggarwal, *Recommender Systems: The Textbook* (Springer, ed. 1, 2016).
35. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016), vol. 1.
36. A. Daniely, R. F. Frostig, Y. Singer, "Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity" in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett, Eds. (Curran Associates, Inc., 2016), pp. 2261–2269.
37. M. Belkin, P. Niyogi, Semi-supervised learning on riemannian manifolds. *Mach. Learn.* **56**, 209–239 (2004).
38. A. Subramanian *et al.*, A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *Cell* **171**, 1437–1452.e17 (2017).
39. A. Belyaeva *et al.*, Causal network models of SARS-CoV-2 expression and aging to identify candidates for drug repurposing. *Nat. Commun.* **12**, 1024 (2021).
40. S. Pushpakom *et al.*, Drug repurposing: Progress, challenges and recommendations. *Nat. Rev. Drug Discov.* **18**, 41–58 (2019).
41. O. S. Kwon *et al.*, Connectivity map-based drug repositioning of bortezomib to reverse the metastatic effect of GALNT14 in lung cancer. *Oncogene* **39**, 4567–4580 (2020).
42. G. Williams *et al.*, Drug repurposing for Alzheimer's disease based on transcriptional profiling of human iPSC-derived cortical neurons. *Transl. Psychiatry* **9**, 220 (2019).
43. C. Squires, D. Shen, A. Agarwal, D. Shah, C. Uhler, Causal imputation via synthetic interventions. arXiv [Preprint] (2020). https://arxiv.org/abs/2011.03127. Accessed 15 August 2021.
44. T. H. Pham, Y. Qiu, J. Zeng, L. Xie, P. Zhang, A deep learning framework for high-throughput mechanism-driven phenotype compound screening and its application to COVID-19 drug repurposing. *Nat. Mach. Intell.* **3**, 247–257 (2021).
45. M. Iwata *et al.*, Predicting drug-induced transcriptome responses of a wide range of human cell lines by a novel tensor-train decomposition algorithm. *Bioinformatics* **35**, i191–i199 (2019).
46. T. Becker *et al.*, Predicting compound activity from phenotypic profiles and chemical structures. bioRxiv [Preprint] (2020). https://www.biorxiv.org/content/10.1101/2020.12.15.422887v3. Accessed 15 August 2021.
47. C. Williams, M. Seeger, "Using the Nyström method to speed up kernel machines" in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, V. Tresp, Eds. (MIT Press, 2000), pp. 682–688.
48. S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, N. Srebro, "Implicit regularization in matrix factorization" in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. (Curran Associates, Inc., 2017), vol. 30, pp. 6152–6160.
49. T. E. Oliphant, *A Guide to NumPy* (Trelgol Publishing USA, 2006), vol. 1.
50. A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library" in *Advances in Neural Information Processing Systems*, H. Wallach *et al.*, Eds. (Curran Associates, Inc., 2019), pp. 8026–8037.
51. O. Russakovsky *et al.*, ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2015).
52. R. Novak *et al.*, "Neural tangents: Fast and easy infinite neural networks in Python" in *International Conference on Learning Representations* (2020).