Nucleic Acids Research, 2020 **1**
doi: 10.1093/nar/gkaa881

A flexible, interpretable, and accurate approach for imputing the expression of unmeasured genes

Christopher A. Mancuso [1,†], Jacob L. Canfield[1,2,†], Deepak Singla[1,3] and Arjun Krishnan [1,2,*]

[1]Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI 48824, USA, [2]Department of Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI 48824, USA and [3]Indian Institute of Technology, Delhi, India

Received April 09, 2020; Revised August 24, 2020; Editorial Decision September 23, 2020; Accepted September 28, 2020

ABSTRACT

While there are >2 million publicly-available human microarray gene-expression profiles, these profiles were measured using a variety of platforms that each cover a pre-defined, limited set of genes. Therefore, key to reanalyzing and integrating this massive data collection are methods that can computationally reconstitute the complete transcriptome in partially-measured microarray samples by imputing the expression of unmeasured genes. Current state-of-the-art imputation methods are tailored to samples from a specific platform and rely on gene-gene relationships regardless of the biological context of the target sample. We show that sparse regression models that capture sample-sample relationships (termed *SampleLASSO*), built on-the-fly for each new target sample to be imputed, outperform models based on fixed gene relationships. Extensive evaluation involving three machine learning algorithms (LASSO, k-nearest-neighbors, and deep-neural-networks), two gene subsets (GPL96–570 and LINCS), and multiple imputation tasks (within and across microarray/RNA-seq datasets) establishes that *SampleLASSO* is the most accurate model. Additionally, we demonstrate the biological interpretability of this method by showing that, for imputing a target sample from a certain tissue, *SampleLASSO* automatically leverages training samples from the same tissue. Thus, *SampleLASSO* is a simple, yet powerful and flexible approach for harmonizing large-scale gene-expression data.

INTRODUCTION

High-throughput gene expression technologies—especially microarray (1) and RNA-sequencing (RNA-seq) (2)—have revolutionized our ability to capture and understand the large-scale cellular context of many biological systems in humans and several model organisms (3,4). Fortunately, due to community-wide norms and funding requirements, nearly all of the resulting transcriptomes have been deposited in publicly-available repositories (5–8). For example, as of 29 January 2020, there are >2 million human microarray samples from >24k datasets along with about half as much human RNA-seq data (>583k samples from ∼12k datasets) contained in the NCBI Gene Expression Omnibus (GEO) database (7,8).

The purpose of these publicly-available data is to enable other researchers to use published datasets to reproduce original findings, reuse datasets in new ways to answer new questions (9), or combine thousands of datasets to build integrative models (10) towards precision medicine (11). However, a major hurdle in realizing these goals is the fact that microarray profiles have been measured using a number of different platforms that each measure a different number of pre-defined genes (ranging from a few hundred genes to ∼20k genes). For instance, the most popular genome-scale platform *Affymetrix Human Genome U133 Plus 2.0 Array* (GEO ID: *GPL570*) accounts for only 22% of the >2 million samples. The next most popular *Affymetrix Human Genome U133A Array* (GEO ID: *GPL96*) accounts for another 11% of the samples, but only covers <12k genes. Therefore, it is a significant challenge to gain insights about the full complement of genes in the human genome across the diversity of biological samples and unique experimental conditions in existing microarray data.

In addition to these researcher-submitted microarray datasets, concerted effort has also been put into defining a reduced set of genes that can be measured and then be used to accurately recover the expression of all the other genes (12,13). The most prominent example of this effort is the Library of Integrated Network-Based Cellular Signatures (LINCS) microarray program (14), which has shown that measuring 978 'landmark' genes, costing only

*To whom correspondence should be addressed. Email: arjun@msu.edu
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

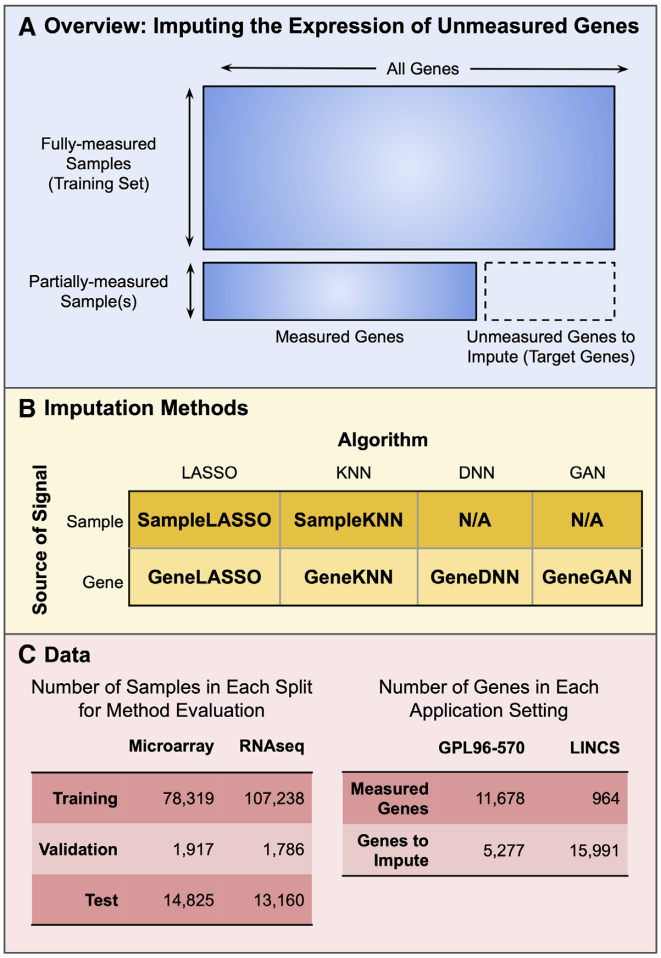

Figure 1. Overview of gene expression imputation. (**A**) Schematic of the problem of 'imputing the expression of unmeasured genes'. A training dataset is used to fill in the expression values of genes from partially-measured samples. The six methods (**B**) and summary of the data (**C**) used in this study. The methods are named using a combination of the machine learning algorithm and biological signal that the method uses.

$5 per sample (15), is sufficient to then use to impute the expression of all other (tens of thousands of) genes. There are currently 1.3 million microarray samples in the LINCS data repository capturing the effect of numerous chemical and genetic perturbations on gene expression (14).

With either of these massive data collections—the >1 million public transcriptomes from various microarray platforms or the 1.3 million LINCS profiles—restricting analysis and integration to the measured genes common to all platforms/samples will result in a tremendous loss of valuable data. Therefore, effectively leveraging the full data compendia on a genome-scale necessitates computational methods that can use the expression levels of the measured genes in a *partially-measured microarray sample* to impute the expression of all *unmeasured genes* in that sample to reconstitute a complete transcriptome (Figure 1A). A few previous studies have indeed proposed methods to solve this problem in various settings. Sparse regression models that use *gene-gene correlation signals* have been shown to be effective in imputing gene expression in samples from the GPL96 (<12k genes) microarray platform based on samples from the GPL570 (whole-genome) platform (16). Others have developed methods that use gene correlations based on low-rank regression (17) and deep neural networks (18,19), specifically for the LINCS dataset. These methods rely on training machine learning models that map the relationship between fixed sets of measured and unmeasured genes in a specific setting, be it sparse gene-based regression for the GPL570-96 setting (16) or deep learning for the LINCS setting (18,19). Methods have also been proposed to address the problem of identifying the best reduced set of genes to measure to enable subsequent imputation of all other genes, again within the scope of specific large datasets (12,13,20). However, all these methods lack the flexibility for broad adoption since public datasets come from many different expression-profiling technologies, with each measuring the expression of a different subset of genes in the genome. All current methods are hard to adapt for imputing unmeasured gene-expression in an arbitrary experiment since they require training completely new models for every microarray platform (or every new reduced gene set design), which, in turn, requires very large datasets for model-training.

Leveraging gene–gene correlations in data was an important component of gene expression imputation that focussed on the related-yet-distinct 'missing value' problem (Supplementary Figure S1), concerned with recovering the expression values of individual genes that were lost *within a single dataset* due to arbitrary technical error in samples, i.e. filling arbitrary empty cells within a larger data matrix (21–23). Many methods have been proposed to tackle this problem (24–30), and, in general, imputing missing values has been shown to improve downstream tasks such as clustering, classification, co-expression network building, and differential expression (31–34). Although these seminal works on the missing-value problem guide imputation methods today, the fact that we can now leverage information from >100k samples at a time to improve the imputation of unmeasured genes requires a rethinking of imputation strategies. Thus, it is critical that new imputation methods select only the most relevant samples to the target sample, as gene–gene correlations change across different biological contexts (35).

In this study, we demonstrate that using a sparse-regression method that leverages information from the most similar samples provides more accurate predictions than other methods while also providing a highly interpretable underlying model. Current state-of-the-art imputation methods train machine learning models that capture the relationship of the predetermined set of measured genes to each predefined unmeasured gene (or set of unmeasured genes). Then, during imputation in an expression sample with the same measured/unmeasured genes, these methods use the pretrained models to impute the expression of the unmeasured genes. We propose a variant of this approach that we call *SampleLASSO* in which, for every new expression sample to be imputed, a new sparse regression model is trained on-the-fly that captures the relationship of this expression sample to all others in the training set based on the genes measured in that given sample. We compare our method to four other imputation

methods based on three different algorithms—*k*-nearest neighbors, regularized linear regression, and deep neural networks—that leverage gene-gene or sample-sample relationships (Figure 1B). All these methods are evaluated on imputation within the same-technology (microarray and RNA-seq), as well as using RNA-seq data to impute microarray data. The evaluation is carried out for two different, practical unmeasured gene settings: (i) a high number of measured genes and low number of unmeasured genes (GPL96–570 gene subset) and (ii) a small number of measured genes and large number of unmeasured genes (LINCS gene subset) (Figure 1C). Extensive evaluations using multiple accuracy metrics within a rigorous temporally-split and dataset-preserving scheme showed that the flexible sample-based imputation method (*SampleLASSO*) always shows competitive performance with the best performing methods, and in particular is always the best performing method when using data from one expression platform to impute data in another expression platform. We also demonstrate the biological interpretability of this method by showing that, for imputing a given sample from a certain tissue, the *SampleLASSO* model automatically up-weights training samples from the same tissue type.

MATERIALS AND METHODS

Data

We used gene expression data from both microarray and RNA-seq technologies. For the microarray data, we downloaded all human samples from the *Affymetrix Human Genome U133 Plus 2.0 Array* from NCBI GEO (8) as raw CEL files and performed background subtraction, quantile transformation, and summarization using fRMA (36) based on a custom CDF (37) mapping probes to Entrez gene IDs. This yielded 108 205 samples with 19 702 genes. For the RNA-seq data, we downloaded all 133 776 TPM-normalized human samples from *ARCHS4* (5), and further processed the data by converting ENST IDs to Entrez gene IDs for only the genes found in the microarray data. Genes that could not be mapped this way were discarded from both microarray and RNA-seq data. This yielded a total of 16 955 genes. Finally, the RNA-seq data was then transformed using the inverse hyperbolic sine (archsinh) function.

We additionally downloaded preprocessed data for 10 gene expression platforms contained in the SEEK database (38). For each gene expression platform in SEEK, we constrained the data to only include genes common to all experiments associated with the platform, and if an experiment contained any samples with missing values, that entire experiment was removed. More information on data processing is provided in Section 1.2 of the Supplemental Material.

Validation scheme

Subsetting genes. To evaluate the imputation methods, we chose to split genes into measured and unmeasured sets to represent two very different practical scenarios (Figure 1C). First, we considered the situation in which we have a large number of measured genes that we could use to impute a smaller number of unmeasured genes. This scenario presents itself in the problem of using the 11 678 genes measured in the older human microarray platform *Affymetrix Human Genome U133A Array* (i.e. GPL96) to then impute the expression of an additional 5 277 genes that are only present in the newer genome-scale platform *Affymetrix Human Genome U133 Plus 2.0 Array* (i.e. GPL570) (16). This gene-split is referred to as the *GPL96–570 gene subset* in this work. Second, we considered the situation in which we have a small number of measured genes that we could use to impute a large number of unmeasured genes. For this scenario, we used 964 'landmark' genes from LINCS as the measured genes to impute the expression of all the other genes in the genome-scale *Affymetrix Human Genome U133 Plus 2.0 Array* (15 991 unmeasured genes) (14,18). This gene-split is referred to as the *LINCS gene subset* in this work.

To determine the set of unmeasured genes to impute in the data from multiple platforms downloaded from the SEEK database, we first found genes that were both common among all ten platforms in SEEK and also contained in the 16 955 genes from the *Affymetrix Human Genome U133 Plus 2.0 Array* and *ARCHS4* data we processed. We used a file that contains the number of PubMed articles a gene was mentioned in (39) to break this set of genes up into three categories; highly-, moderately-, or scarcely-studied genes. We then chose 30 genes from each of these three sets, resulting in the same set of 90 unmeasured genes to be imputed for all the platforms contained in the SEEK database. Each platform in SEEK had a different set of measured genes which was determined by using genes common between that platform and the 16 955 genes from the *Affymetrix Human Genome U133 Plus 2.0 Array* and *ARCHS4* data we processed, but excluded any gene in the set of the 90 genes used in the unmeasured set (Supplementary Table S1).

Splitting samples. We divided the expression samples into training, validation, and testing sets. The training data was used to fit the models, the validation data was used for hyperparameter tuning, and the testing data was used in the final evaluations of the models (shown in all the figures in the main text). To mitigate data leakage, we ensured that entire datasets were assigned to splits, thus keeping all expression samples from the same experiment (dataset) together in the same split. The data was also temporally split, with the oldest expression samples being placed in the training and validation sets, and the newest samples going into the test set. To speed up hyperparameter tuning, which consisted of training >500 000 individual models, we further subsetted the validation set by taking 10% of the expression samples from each experiment in the full validation set (or at least two expression samples, if the number of samples in an experiment was <20) (see Section 1.2 in Supplemental Material and Supplementary Figure S2).

For the analysis of imputing data from the SEEK database, we used the same training data for every platform: the training data set from the *Affymetrix Human Genome U133 Plus 2.0 Array* described above. For each platform in

SEEK, we generated a test set from 90 randomly selected samples contained in that platform to ensure that both sample- and gene-based methods had the same number of examples to predict during testing.

For all imputation methods, we standardized each feature in the training set by subtracting the mean and dividing by the standard deviation of the given feature. Correspondingly, each feature in the validation and test sets was standardized using the mean and standard deviation obtained from the training set.

Imputation methods

In this study, we evaluated imputation methods using four distinct machine-learning algorithms: *k*-nearest neighbors (*KNN*), least absolute shrinkage and selection operator (*LASSO*), a fully-connected feedforward deep neural network (*DNN*), and a conditional generative adversarial network (*GAN*).

KNN is a machine learning algorithm that predicts the target variable for every new example based on the target variables of the k most similar examples in the training data. To impute a given target variable, we used a weighted average of the measured target variable from the k most similar examples based on Euclidean distance, with the weight equal to the inverse of the distance. KNN imputation is a widely-used imputation method for gene expression and provides a strong baseline (12,18,22,24).

LASSO is a linear regression method in the family of least-squares optimizers (40). LASSO builds a sparse model for a given target variable using the following cost function:

$$min_{\beta} \frac{1}{2N} ||X\beta - y||_2^2 + \alpha||\beta||_1 \quad (1)$$

where N is the number of training examples, β is the vector of learned parameters, X is the training data, y is the target variable, α is the hyperparameter that determines the extent of L1-regularization ($||\beta||_1$). L1-regularization prevents overfitting by setting many of the elements of β to 0. While a variety of least-squares optimizers have been applied to the gene expression imputation problem (22,25,41,42), LASSO is the method most suited when the number of features is large (12,18,43). In this study, we used the KNN and LASSO implementations contained in the Python package *scikit-learn* (44).

A *DNN* is a multi-layer feedforward neural network with bespoke architectures designed for each machine learning task. A *GAN* is a deep neural network consisting of two main parts: a 'generator' that generates imputed values and a 'discriminator' that attempts to discriminate between imputed values and the ground truth expression values. For both the *DNN* and *GAN* models, we used architectures from recently published works that evaluated the utility of these models for imputing gene expressions using the LINCS landmark genes; namely the *D-GEX* model for *DNN* (18) and the *GGAN* model for *GAN* (19). The *DNN* and *GAN* models were trained using Nvidia Tesla k80 GPUs and implemented using the Python package *Keras* (45) with a *Tensorflow* backend (46). For more information on the deep learning methods, see Section 1.3 of the Supplemental Material.

We used these four algorithms—KNN, LASSO, DNN, GAN—to leverage two distinct types of signals—gene–gene similarities (across samples) and sample–sample similarities (across genes)—for imputing the expression of unmeasured genes in a new partially-measured sample, resulting in six methods referred to in this study as *SampleKNN*, *GeneKNN*, *SampleLASSO* (proposed here), *GeneLASSO*, *GeneDNN* and *GeneGAN*. For intuitive, pictorial schematics of the methods, see Supplementary Figures S3-S7 in Section 1.3 of the Supplemental Material.

SampleKNN and *GeneKNN* are the most straightforward and popular implementation of KNN for gene expression imputation. For a new partially-measured expression sample to be imputed, *SampleKNN* works by first finding the k most similar samples in the training set based on the expression of all measured genes, and then imputing the expression of each unmeasured gene with the weighted average of that gene's expression in the most similar training samples. Thus, the major biological signal used is the similarity between samples (across genes). Conversely, *GeneKNN* works on a gene-by-gene basis. For each gene that is missing (unmeasured) in a new sample, the method first finds the k measured genes most similar in their expression pattern across all the samples in the training set, and then imputes the expression of the unmeasured gene with the weighted average of the expression of those k genes in that new sample. Thus, the major biological signal used is the similarity between genes (across samples).

GeneLASSO is the traditional, widely adopted means of implementing LASSO for gene expression imputation. Here, using the fully-measured training set, a separate sparse regression model is trained for each unmeasured gene, to predict its expression based on a linear combination of all the measured genes. Then, given a new partially-measured sample, the expression of every unmeasured gene is imputed using that gene's pre-trained model, with the predicted expression being equal to the sum of the expression of the measured genes in the new sample weighted by the model coefficients. Akin to *GeneKNN*, the main source of biological signal for *GeneLASSO* is gene–gene expression similarities. As an alternative to *GeneLASSO*, which requires information about which genes are unmeasured in a new sample and a pre-trained model for each of those genes, in this study, we propose a simple alternative called *SampleLASSO*. Given a new partially-measured sample, *SampleLASSO* builds a single model on-the-fly that predicts that sample's expression profile based on a sparse linear combination of all the samples in the training set only using the subset of genes measured in the new sample. Here, for every sample to be imputed, the coefficients of the trained model in essence finds the relationship of that sample to all samples in the training set. Then, all the unmeasured genes are imputed using this trained sample-specific model. The main source of biological signal in SampleLASSO, thus, comes from sample similarities. We note a method similar to *SampleLASSO* has been reported before (called *LS_array*) (25). However, the implementation of that method was focused on the missing value problem and has never been applied to the unmeasured gene problem.

GeneDNN and *GeneGAN* use deep learning to predict the expression of (a fixed set of) unmeasured genes using a single model that captures complex, nonlinear relationships between the unmeasured genes and the measured genes using the training set. Thus, the main biological signal in *GeneDNN* and *GeneGAN* is also from gene-gene relationships.

Hyperparameter tuning

For both KNN and LASSO, there is only one hyperparameter that needs to be tuned; *k:* the number of most similar training examples to consider in KNN and α: the parameter that sets the strength of the L1-regularization term in LASSO. Although there are many hyperparameters to tune in a DNN or GAN, we fixed most parameters based on optimal values found in the previously published works (18,19), and just tuned the optimizer and learning rate. Hyperparameter tuning was done using the validation set data (Section 1.4 in Supplemental Material, Supplementary Figures S9–S15, and Supplementary Table S2), and the optimal hyperparameters were then used for final evaluation using the test set. We note that, for *GeneDNN and GeneGAN*, we used all the ∼13k samples in the validation set for hyperparameter tuning as: (i) this most closely mimics the setup in the original papers and (ii) deep learning models additionally use the validation set to determine which epoch (i.e. how many passes through the data the model goes through) yields the best model. For the analysis using the SEEK data, the best hyperparameter across all the tasks and gene subsets using the *Affymetrix Human Genome U133 Plus 2.0 Array* and *ARCHS4* data was used for all ten platforms. For more information on hyperparameter tuning, see Section 1.4 of the Supplemental Material.

Evaluation metrics

A commonly used metric for evaluating gene expression imputation methods is Normalized Root Mean Square Error (NRMSE). The NRMSE for a gene (g_i) is given by:

$$NRMSE(g_i) = \frac{RMSE(g_i)}{Mean(g_i)} = \frac{\sqrt{\sum_{j=1}^{S}\left(\hat{g}_{i,j} - g_{i,j}\right)^2/S}}{\sum_{j=1}^{S} g_{i,j}/S} \quad (2)$$

where *RMSE* is the root mean square error, S is the number of samples, and $\hat{g}_{i,j}$, $g_{i,j}$ are the imputed and real expression values, respectively, for the ith gene in the jth sample.

In addition to NRMSE, we also report evaluation results using the Spearman correlation coefficient and the Mean Absolute Error (MAE) in Section 2.2 of the Supplemental Material.

Interpreting *SampleLASSO* models

We evaluated the interpretability of *SampleLASSO* models by examining if the β-coefficients of a model trained for a particular sample recapitulated that sample's tissue-of-origin by assigning high positive β values to samples in the training set from the same tissue relative to samples from all other tissues.

Specifically, for a given target sample s that we built a *SampleLASSO* model for, we calculated a z-score, $z_{s,T}$, for each tissue T in this annotated set based on the β values of training samples from that tissue:

$$z_{s,T} = \frac{\left(\sum_{j:tissue(j)=T}\beta_j\right)/|T| - \mu_s}{\sigma_s/\sqrt{|T|}} \quad (3)$$

where $|T|$ is the number of labeled samples for tissue T, β_j is the value of the β-coefficient from the *SampleLASSO* model for the jth sample, and $\mu_s$and σ_s are the mean and standard deviation of the β-coefficients of all samples in the training set that have any tissue label.

To perform this analysis we used a large set of expression samples that were manually-curated to their tissue-of-origin (>15k samples) (47). However, due to the initial temporal split of the data into training, validation, and test sets, all the labeled expression samples were in the original training set. Hence, just for this interpretability analysis, we separated out a subset of the tissue-labeled samples in the original training set into a new manually-curated test set. We created this subset so that it spanned six tissues that were sufficiently diverse and were labeled to at least 10 samples from at least three different datasets in both the training and the test sets. This resulted in the new test set having 222 expression samples from 29 different datasets. The full training set consisted of all samples from the original training set, expect we removed any sample that was part of the same experiment as any test set sample. Of these 77 893 training samples, 11 618 samples had a manually curated tissue label, with 4397 expression samples from 120 different datasets having a label pertaining specifically to the six tissues the analysis was carried out for (Table S4). To calculate μ_s and σ_s, we used any sample that had a tissue label, regardless of tissue type, allowing us to use 11 618 samples for these calculations. A *SampleLASSO* model was trained for each manually labeled test sample and used in the z-score analysis above (Equation 3).

RESULTS

In this study, we compare imputation methods that use four distinct machine learning algorithms: least absolute shrinkage and selection operator (LASSO), k-Nearest Neighbors (KNN), a fully-connected feedforward neural network (DNN), and a conditional generative adversarial network (GAN) (Figure 1B). Combining these algorithms with the source of the data signal—gene–gene or sample–sample relationships—resulted in six imputation methods: *SampleLASSO*, *GeneLASSO*, *SampleKNN, GeneKNN, GeneDNN* and *GeneGAN*. These methods are evaluated in two settings with different sets of unmeasured genes (Figure 1C): (i) the GPL96–570 gene subset, which uses a relatively large number of genes (∼11 000) to impute the expression of a smaller number of genes (∼5000) and (ii) the LINCS gene subset, which uses a relatively small number of genes (∼1000) to impute the expression of a large number of genes (∼16 000). We consider imputation using data from the same technology (using microarray

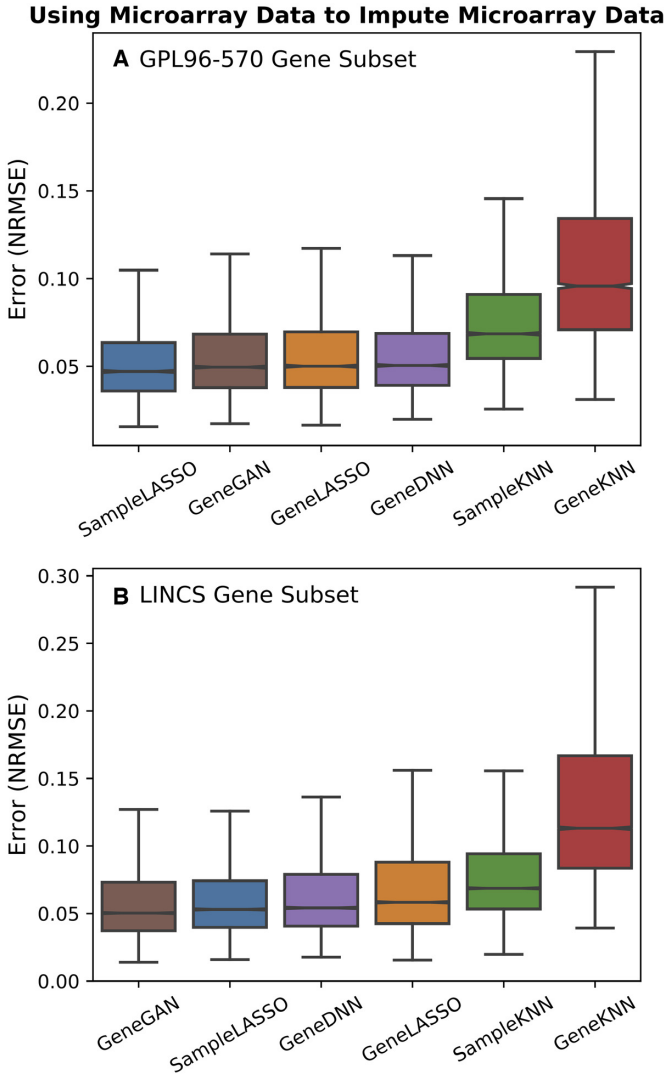

Figure 2. Performance of imputation methods for microarray data. Boxplots showing the performance of the six imputation methods (*SampleLASSO*, *GeneGAN*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*) across two gene subsets (**A**: GPL96-570 and **B**: LINCS), trained and imputed on microarray data. The evaluation metric is NRMSE, with lower values indicating better performance, and the methods are ordered by the median value.

to impute microarray, and RNA-seq to impute RNA-seq) as well as across technologies (using RNA-seq to impute microarray data). We evaluate methods using both the scale-free regression error metric (normalized root mean squared error; NRMSE), as well as Spearman correlation and mean absolute error. We also evaluate the performance of the imputation methods on data from ten additional gene expression platforms downloaded from the SEEK database (38). Lastly, we examine the model coefficients learned by *SampleLASSO* for biological interpretability.

We first evaluated the performance of the six imputation techniques using microarray data to impute microarray data (the microarray data here all comes from the *Affymetrix Human Genome U133 Plus 2.0 Array*) (Figure 2). For the GPL96–570 gene subset task, *SampleLASSO* is the best performing model, whereas for the LINCS gene subset, *GeneGAN* is the best performing method. For both gene subsets, both KNN methods perform relatively poorly. For the GPL96–570 gene subset, *SampleLASSO* outperforms *GeneGAN* 92% of the time, *GeneLASSO* 91% of the time, *GeneDNN* 95% of the time, *SampleKNN* 100% of the time, and *GeneKNN* 100% of the time. For the LINCS gene subset, these percentages are 25%, 91%, 77%, 98% and 100%, respectively. Statistical tests and effect sizes between *SampleLASSO* and the other methods can be found in Supplementary Table S3.

Although microarray platforms like the *Affymetrix Human Genome U133 Plus 2.0 Array* are able to quantify the expression of nearly all protein-coding genes, RNA-seq technology enables the quantification of nearly all cellular transcripts from both annotated and unannotated genes. Hence, it would be valuable to use RNA-seq data to predict the expression of genes missing in microarrays, enabling (i) re-analysis of novel genes in experimental settings captured in the vast number of microarray datasets and (ii) joint analysis and integration of RNA-seq and microarray data based on a common set of genes. We evaluated the performance of using *ARCHS4* RNA-seq data to impute *Affymetrix Human Genome U133 Plus 2.0 Array* microarray data using the GPL96–570 and LINCS gene subsets (Figure 3). *SampleLASSO* is the best performing method for both gene subsets. For the GPL96–570 gene split, *SampleLASSO* outperforms *GeneGAN* 71% of the time, *GeneLASSO* 71% of the time, *GeneDNN* 91% of the time, *SampleKNN* 80% of the time, and *GeneKNN* 70% of the time. For the LINCS gene split, these percentages change to 56%, 62%, 76%, 76% and 84%, respectively. The Wilcoxon ranked-sum test between *SampleLASSO* and the other methods showed that the performance increase of *SampleLASSO* was always statistically significant (P-value $<< 0.001$).

We also evaluated the methods using RNA-seq data to impute RNA-seq data (the RNA-seq data here all comes from the *ARCHS4* database) (Figure 4). We note that using RNA-seq data to impute RNA-seq data does not have as many obvious applications as the microarray setting since RNA-seq technologies do not require pre-determining a set of genes to measure, and thus have high gene coverage. For this task, one of the deep learning methods is the best performing method for both gene subsets. For the GPL96–570 gene split, *SampleLASSO* outperforms *GeneGAN* 40% of the time, *GeneLASSO* 41% of the time, *GeneDNN* 39% of the time, *SampleKNN* 93% of the time, and *GeneKNN* 98% of the time. For the LINCS gene split, these percentages change to 31%, 62%, 13%, 84% and 100%, respectively. Statistical tests and effect sizes between *SampleLASSO* and the other methods can be found in Supplementary Table S3. For all three imputation tasks and both gene subsets, we find similar results when measuring imputation accuracy using Spearman correlation and mean absolute error (Supplementary Figure S16–S18), except for the RNA-seq to microarray imputation task on the LINCS gene subset, where for Spearman correlation *GeneGAN* and *GeneLASSO* both outperform *SampleLASSO*.

When imputing gene expression values from one platform or technology to another, the imputation error

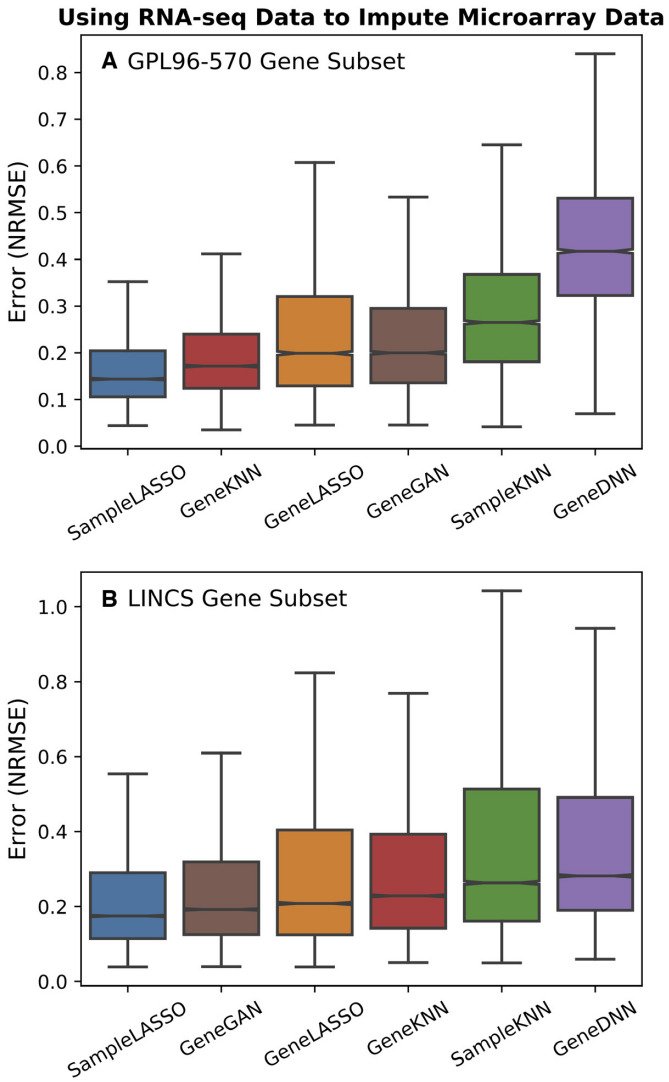

Figure 3. Performance of imputation methods for cross-technology imputation. Boxplots showing the performance of the six imputation methods (*SampleLASSO*, *GeneGAN*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*) across two gene subsets (**A**: GPL96-570 and **B**: LINCS) using RNA-seq data to impute microarray data. The evaluation metric is NRMSE, with lower values indicating better performance, and the methods are ordered by the median value.

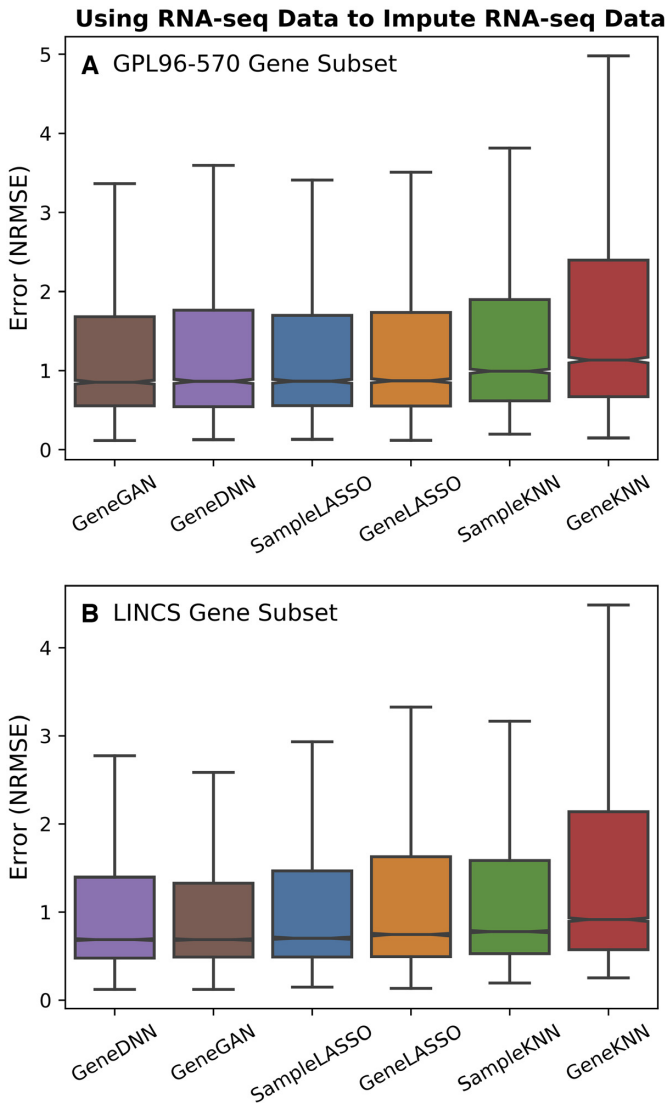

Figure 4. Performance of imputation methods for RNA-seq data. Boxplots showing the performance of the six imputation methods (*SampleLASSO*, *GeneGAN*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*) across two gene subsets (**A**: GPL96-570 and **B**: LINCS) using RNA-seq data to impute RNA-seq data. The evaluation metric is NRMSE, with lower values indicating better performance, and the methods are ordered by the median value.

stands to be lowered by jointly normalizing the data in an appropriate way before imputation. This is not an easy task as the distribution of the unmeasured genes for the samples to be imputed is inherently not known. However, to get a sense of how much even applying basic normalization helps the RNA-seq to microarray imputation task, we used the known ground truth values for the unmeasured geneset to jointly quantile normalize all of the data together; a similar normalization strategy was applied in recent imputation studies (18,19). Performing this joint quantile normalization of the two datasets results in the expected decrease in imputation error over the no-normalization case (Supplementary Figure S19).

We additionally looked at the performance of the imputation methods as a function of the mean expression and variance of the unmeasured genes. For each unmeasured gene, we placed it into a low, medium or high bin based on its mean expression and variance in the context of the known expression values across all the samples (Supplementary Figures S20–S25). Although the relative performance of the imputation methods does not change too much when looking at the different categories of mean expression and variance, it can be seen in some cases that *SampleLASSO* performs particularly well for genes with low mean expression and high variance, which is the hardest category of genes to impute.

As stated before, since there are millions of microarray samples from dozens of gene expression platforms, each with a different subset of measured genes, it is critical that

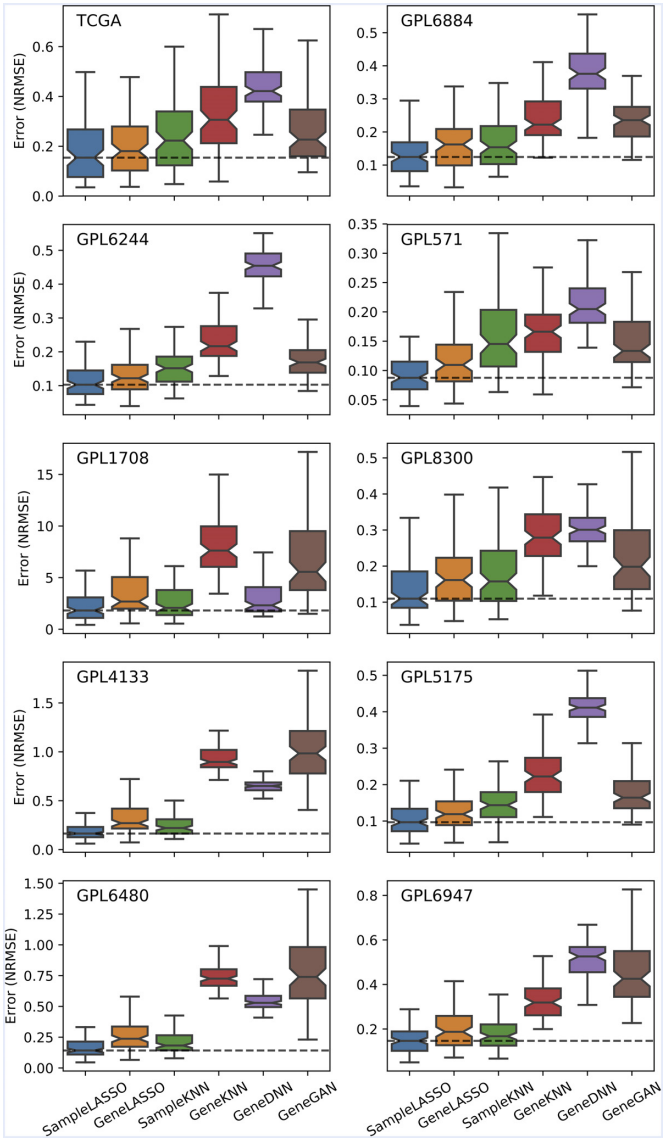

Figure 5. Performance of imputation methods on ten expression platforms. Boxplots showing the performance of the six imputation methods (*SampleLASSO*, *GeneGAN*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*) across ten gene expression platforms. The evaluation metric is NRMSE, with lower values indicating better performance. The horizontal dashed line indicates the median value of *SampleLASSO*.

a good imputation method can accurately impute gene expression values in all these different settings. To evaluate this practical application, we tested the six imputation methods on their ability to impute gene expression values in data from ten platforms (obtained from the SEEK database) (38) (Figure 5, Supplementary Figure S26). We note that this analysis of imputing expression values across many different platforms has not been conducted in any of the recent studies on gene expression imputation.

For this analysis, we choose a set of 90 genes common to all platforms, and containing highly-, moderately-, and scarcely-studied genes, as the set of unmeasured genes. For all ten platforms, we used the same 78 319 training samples from the *Affymetrix Human Genome U133 Plus 2.0 Array* data. For each of the 10 platforms, the set of measured genes was determined by taking the genes common to a given platform and the *Affymetrix Human Genome U133 Plus 2.0 Array* data, excluding any gene that was in the set of 90 unmeasured genes. It can be seen that *SampleLASSO* is the best performing method across all ten platforms, whereas both deep learning models perform quite poorly. For more information on the analysis using the SEEK data, see Section 2.5 of the Supplemental Material.

Although reducing the error of imputation methods is of the utmost importance, the applicability of an imputation method is also greatly increased if the model offers biological insight into how the predictions were generated. For instance, *SampleKNN* offers immediate interpretability via the biological/experimental contexts of the *k*-nearest training samples picked by the method, however it is not very accurate in imputing unmeasured genes. Since we devised *SampleLASSO* with this desirable property in mind, we tested if this new method also offers biological interpretability in addition to providing very accurate imputation. Since gene expression samples have clear signals pertaining to their tissue-of-origin (35), we focused on testing if the *SampleLASSO* model trained for a new sample from a particular tissue up-weighted training samples from that same tissue relative to training samples from other tissues.

To perform this analysis we used a large set of expression samples that were manually labeled to their tissue-of-origin (47). Due to limited labeled tissues and their representation in our data, the analysis was restricted to six sufficiently-diverse tissues that had at least 10 samples from at least three different datasets in both the training set and the test set (Supplementary Table S4). Then, for each test sample, we trained a SampleLASSO model and used the model coefficients (β-coefficients) to calculate a z-score for each of the six tissues that represents the aggregate β-coefficients corresponding to training samples just from that tissue relative to the background distribution of β-coefficients of all labeled training samples (not just samples labeled from the six tissues). Thus, a large positive z-score for a particular tissue means that samples from that tissue were more informative than others. See *Materials and Methods* for more details. Implementing this analysis on the entire labeled test set, we observe that for most test samples, the strongest signal captured by *SampleLASSO* comes from training samples from the same tissue as the sample being imputed (Figure 6, Supplementary Figure S27).

To aid in reproducibility, we have publicly released all the processed data that we used on Zenodo (https://doi.org/10.5281/zenodo.3971092) as well as all the code to re-generate the results and figures on GitHub (https://github.com/krishnanlab/Expresto). In addition, we also provide a user-friendly function that performs imputation using *SampleLASSO* given a file of expression data. The function will return the imputed expression values as well as a list of the most utilized samples from the training set (see Supplementary Table S5 for an example).

DISCUSSION

In this study, we propose a simple, new method termed *SampleLASSO* for imputing the expression of unmeasured genes in partially-measured gene-expression profiles. The

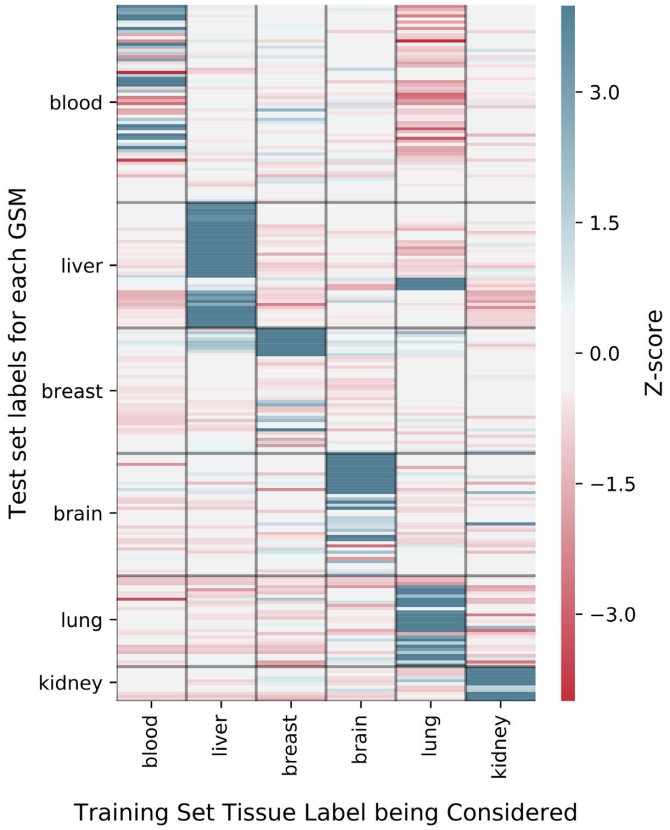

Figure 6. *SampleLASSO* captures biologically relevant information. Biological interpretability was evaluated using expression samples labeled with tissue-of-origin to determine if *SampleLASSO* up-weights training samples of the same tissue as the test sample in the sparse model. The rows represent the 222 tissue-labeled samples (GSMs) in the test set. The black horizontal lines separate test samples from different tissues. The columns correspond to the tissue type of the training samples. The colors of the heatmap represent the z-scores. Calculated per test sample (row), the z-score per tissue (column) corresponds to the normalized aggregate of the model coefficients of all the training samples from that tissue. See Materials and Methods for more details. The diagonal blocks correspond to the case where the z-scores were calculated for the same tissue type as the tissue-of-origin of the test sample.

fundamental benefit of imputing unmeasured genes in samples from a partially-measured platform is the ability for researchers to reanalyze the data from all these samples on a genome-scale. Thus, important common tasks such as building co-expression networks, differential expression analysis and gene set enrichment will now include the full complement of genes in the genome. This benefit is particularly evident for data in the LINCS database, where samples only measure expression from ∼1000 genes. An additional major benefit comes from the ability to perform integrative analysis of data from multiple platforms with maximum gene coverage. This increase in coverage is critical for data-integration efforts or big data-driven machine learning models that pull together data across many platforms. These efforts will be severely stifled when they are forced to only include genes common to all platforms, resulting in a huge loss of information. For example, for combining all the data in the SEEK database, with imputation, we can build a gene-expression set that contains the expression of 19 124 genes (the number of genes in the platform with the highest gene coverage), whereas if we only included genes common to all the platforms, that expression set would only contain the expression of 6017 genes.

SampleLASSO is a sparse regression model that trains a machine learning model on-the-fly for every expression profile that needs to have expression values imputed. We demonstrate that *SampleLASSO* is a highly accurate method for imputing unmeasured genes based on an extensive evaluation on three different imputation tasks (within and across technologies), two imputation settings that differ in the number of measured genes by an order of magnitude, and across multiple gene expression platforms. *SampleLASSO*'s strength comes from its ability to effectively leverage information from samples from the same biological context.

In addition to helping estimate the performance of imputation methods, our analyses in different imputation settings highlight various data standardization/normalization scenarios. When using microarray data to impute microarray data, the training data and validation/testing data (including both measured and unmeasured genes) are quantile normalized to the same distribution. When using RNA-seq data to impute RNA-seq data, samples only undergo within-sample normalization (using TPM) without any between-sample normalization. When using RNA-seq data to impute microarray, the training set data is not jointly normalized but the validation/testing data are. The fact that microarray data has a much lower imputation error than the RNA-seq also points to RNA-seq profiles having very different data distributions due to not being (quantile) normalized across samples, coming from many different sequencing platforms, and having a broader dynamic range than microarray data. While we have performed an analysis showing that jointly normalizing RNA-seq and microarray data improves imputation (Supplementary Figure S19), future work is required to examine the effect of data normalization and transformation and to develop strategies to perhaps transform data just based on measured genes and, upon imputation, recast the unmeasured genes into the original data space (see Section 2.3 in the Supplemental Material for a further discussion on this topic).

The performance of the various imputation methods in the cross-technology imputation task, where the influence of data transformation is most evident, highlights how each method works to impute gene expression. *SampleKNN* performs poorly because, for a given microarray sample to impute, it finds the closest RNA-seq samples, which come from a different data distribution. On the other hand, *GeneKNN* has relatively good performance because it works completely within the training data (RNA-seq) to find the genes nearest to a particular gene and then uses these gene relationships for imputation within the microarray data. Even though *GeneLASSO* similarly captures gene relationships only using the training data (RNA-seq), the mapping in the form of model coefficients does not transfer to microarray samples as easily as nearest neighbors. Similar issues, in addition to potential overfitting to the training set, thwart the performance of *GeneDNN*

(Supplementary Figure S28), although the additional complexity of *GeneGAN* helps deep neural networks to better transfer information from one technology to the other. *SampleLASSO*'s top performance in this setting stems from having the unique property of learning a supervised model that, in addition to learning meaningful sample relationships, naturally captures the scaling factors required to closely map the RNA-seq data distribution to the microarray data distribution.

Our results indicate that the deep learning models show good performance for imputation in the LINCS subset using microarray data to impute microarray data. This is to be expected as LINCS is the application for which these methods were optimized. The deep learning methods additionally show superior performance on the task of using RNA-seq data to impute RNA-seq data. This could be due to having more training examples for this task compared to the microarray to microarray task. The deep-learning methods further benefit from minimal variation in the RNA-seq data across platforms due to the uniform processing of the samples in the ARCHS4 database.

Nevertheless, deep-learning-based imputation techniques have a number of practical drawbacks. Foremost among them is the very large number of hyperparameters, in addition to other aspects of model optimization such as regularization and dropout, that need to be tuned to build an accurate model. This drawback is further amplified by the fact that these optimal parameters are likely to change depending on the platform or technology in question. Deep learning models are also hard to scale in terms of model size, with the number of weights growing nonlinearly with increasing layer number and layer size. Additionally, the current state of the standard hardware is such that only ∼32GB of a model can fit into the memory of a GPU, and anything more requires the utilization of multiple GPUs. For example, this is why methods such as *SampleDNN* or *SampleGAN* could not be implemented, as in these cases the input layer would be comprised of >70 000 units (i.e. the number of units equals the number of samples in the training set), and even a neural network with just one hidden layer of 9000 units would be over double the size of a *GeneDNN* model with three hidden layers of 9000 units each in the LINCS gene subset setting.

The analysis using the SEEK data—which has not been carried out in any other study on imputation and which reflects the real world task of imputing data from many expression platforms—highlights a few more limitations of the deep learning methods. First, the time it takes to get imputed expression values from deep learning methods is determined by the size of the training set, and once the model is trained, predicting test set values is relatively fast. Therefore, there was no difference in runtime for the deep learning models whether we imputed 10 000 samples or just 90, and this runtime ended up being >20 h for *GeneDNN* and >40 h for *GeneGAN* per platform using expensive GPU hardware. In contrast, *SampleLASSO* took just 4 h per platform in SEEK using only one processor on one computational node. As each *SampleLASSO* model is independent of the others, this is an 'embarrassingly parallel' task and can be greatly sped up by running on many CPUs at once.

Secondly, in the SEEK data analysis, the deep learning methods perform noticeably poorly for almost all platforms. This is likely due to the fact that, since we are only imputing 90 genes, deep learning methods no longer benefit from the multi-task learning that happens when many thousands of genes are all predicted at once. In contrast, *SampleLASSO* and *GeneLASSO* train a model for every sample or gene, respectively. This is an important real world consideration as a researcher only may wish to add a limited number of 'genes of interest' to a given set of microarray samples, and *SampleLASSO* allows for a much quicker exploratory analysis of combining datasets across platforms.

The performance of the deep learning methods on the SEEK data could be improved if the model hyperparameters were optimized for each platform. This is not realistic in practice as there are numerous ways of combining data from expression platforms, and the computational time and the technical expertise required to optimize a deep learning model in a given imputation setting are not available to most researchers. We also note that the deep learning methods perform best when the training and testing data are similar (i.e. analysis shown in Figures 2 and 4). However, the analysis using the SEEK data more closely mimics the situation of using RNA-seq data to impute microarray data where the training and testing data comes from different distributions. More broadly, cross-platform imputation is the most meaningful setting because imputing samples from a platform using training data of the same platform is not needed as both training and testing samples will measure the expression of the same set of genes. In this cross-platform setting, interesting future work could be in applying domain adaptation techniques to further decrease the imputation error.

SampleLASSO is a simple, intuitive, flexible model. The benefits of *SampleLASSO* emanate from the fact that a new machine learning model is trained on-the-fly for each new target sample that needs to be imputed based on the set of measured genes in that sample. Any set of genes can be measured/unmeasured in this setup, obviating the need for fixed pretrained models. While we acknowledge that there are many more deep learning architectures that could be applied to the unmeasured gene expression problem, we highlight the fact there also exists many ways to optimize simple and elegant methods like *SampleLASSO* by exploring different regularization schemes and shallow nonlinear algorithms, or by reducing the number of features using feature selection methods or dimensionality reduction techniques.

Finally, these benefits come along with *SampleLASSO*'s ability to leverage biological information specific to the new target sample, enabling easy interpretability. We specifically evaluated *SampleLASSO*'s interpretability on a large-scale using samples labeled to various tissues of origin. This analysis shows that, when imputing a sample from a specific tissue, the *SampleLASSO* model up-weights training samples from the same tissue in majority of the cases (Figure 6 and Supplementary Figure S27). The rest of the cases could be due to high tissue heterogeneity (as in blood) or factors other than tissue-type (e.g. disease status, drug dosage) being the dominant signal.

In conclusion, we propose *SampleLASSO*, a simple method for imputing the expression of unmeasured genes. Extensive evaluations and analyses demonstrate that *SampleLASSO* is accurate, flexible, and interpretable. We have made all the data and code from this study freely available on Zenodo and GitHub (https://github.com/krishnanlab/Expresto) to aid in reproducing all our findings. Using a convenient function in our code, researchers can also use *SampleLASSO* to readily impute unmeasured genes in their samples of interest in any of the following practical settings: (i) complete the expression profile of publicly-available microarray samples from any platform to make them comparable to the human whole-genome microarray, (ii) predict the expression of genes absent in standard microarrays (e.g. most non-protein coding genes) using RNA-seq to impute microarray samples and (iii) fill in and effectively use genome-scale chemical and genetic perturbation expression data from LINCS based on the measured landmark genes.

DATA AVAILABILITY

The data from this study is available on Zenodo (https://doi.org/10.5281/zenodo.3971092) and all code is available on Github (https://github.com/krishnanlab/Expresto).

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We thank Kayla A. Johnson, Nathaniel Hawkins, and the rest of the Krishnan Lab for valuable discussions and feedback on the manuscript.

FUNDING

US National Institutes of Health (NIH) [R35GM128765 to A.K.]; MSU start-up funds (to A.K.); NIH F32 Fellowship [F32GM134595 to C.M.] (in part). Funding for open access charge: MSU Start-up Funds [to A.K.].
Conflict of interest statement. None declared.

REFERENCES

1. Heller,M.J. (2002) DNA microarray technology: devices, systems, and applications. *Annu. Rev. Biomed. Eng.*, **4**, 129–153.
2. Wang,Z., Gerstein,M. and Snyder,M. (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, **10**, 57–63.
3. Stark,R., Grzelak,M. and Hadfield,J. (2019) RNA sequencing: the teenage years. *Nat. Rev. Genet.*, **20**, 631–656.
4. Hoheisel,J.D. (2006) Microarray technology: beyond transcript profiling and genotype analysis. *Nat. Rev. Genet.*, **7**, 200–210.
5. Lachmann,A., Torre,D., Keenan,A.B., Jagodnik,K.M., Lee,H.J., Wang,L., Silverstein,M.C. and Ma'ayan,A. (2018) Massive mining of publicly available RNA-seq data from human and mouse. *Nat. Commun.*, **9**, 1366.
6. Athar,A., Füllgrabe,A., George,N., Iqbal,H., Huerta,L., Ali,A., Snow,C., Fonseca,N.A., Petryszak,R., Papatheodorou,I. *et al.* (2019) ArrayExpress update – from bulk to single-cell expression data. *Nucleic Acids Res.*, **47**, D711–D715.
7. Edgar,R., Domrachev,M. and Lash,A.E. (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, **30**, 207–210.
8. Barrett,T., Wilhite,S.E., Ledoux,P., Evangelista,C., Kim,I.F., Tomashevsky,M., Marshall,K.A., Phillippy,K.H., Sherman,P.M., Holko,M. *et al.* (2013) NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Res.*, **41**, D991–D995.
9. Rung,J. and Brazma,A. (2013) Reuse of public genome-wide gene expression data. *Nat. Rev. Genet.*, **14**, 89–99.
10. Greene,C.S., Krishnan,A., Wong,A.K., Ricciotti,E., Zelaya,R.A., Himmelstein,D.S., Zhang,R., Hartmann,B.M., Zaslavsky,E., Sealfon,S.C. *et al.* (2015) Understanding multicellular function and disease with human tissue-specific networks. *Nat. Genet.*, **47**, 569–576.
11. Alyass,A., Turcotte,M. and Meyre,D. (2015) From big data analysis to personalized medicine for all: challenges and opportunities. *BMC Med. Genomics*, **8**, 33.
12. Donner,Y., Feng,T., Benoist,C. and Koller,D. (2012) Imputing gene expression from selectively reduced probe sets. *Nat. Methods*, **9**, 1120–1125.
13. Rudd,J., Zelaya,R.A., Demidenko,E., Goode,E.L., Greene,C.S. and Doherty,J.A. (2015) Leveraging global gene expression patterns to predict expression of unmeasured genes. *BMC Genomics*, **16**, 1065.
14. Subramanian,A., Narayan,R., Corsello,S.M., Peck,D.D., Natoli,T.E., Lu,X., Gould,J., Davis,J.F., Tubelli,A.A., Asiedu,J.K. *et al.* (2017) A next generation connectivity Map: L1000 platform and the first 1,000,000 profiles. *Cell*, **171**, 1437–1452.
15. Peck,D., Crawford,E.D., Ross,K.N., Stegmaier,K., Golub,T.R. and Lamb,J. (2006) A method for high-throughput gene expression signature analysis. *Genome Biol.*, **7**, R61.
16. Zhou,W., Han,L. and Altman,R.B. (2017) Imputing gene expression to maximize platform compatibility. *Bioinformatics*, **33**, 522–528.
17. Ye,G., Tang,M., Cai,J.-F., Nie,Q. and Xie,X. (2013) Low-rank regularization for learning gene expression programs. *PLOS ONE*, **8**, e82146.
18. Chen,Y., Li,Y., Narayan,R., Subramanian,A. and Xie,X. (2016) Gene expression inference with deep learning. *Bioinformatics*, **32**, 1832–1839.
19. Wang,X., Ghasedi Dizaji,K. and Huang,H. (2018) Conditional generative adversarial network for gene expression inference. *Bioinformatics*, **34**, i603–i611.
20. Abid,A., Balin,M.F. and Zou,J. (2019) Concrete autoencoders for differentiable feature selection and reconstruction. arXiv doi: https://arxiv.org/abs/1901.09346, 31 January 2019, preprint: not peer reviewed.
21. Aittokallio,T. (2010) Dealing with missing values in large-scale studies: microarray data imputation and beyond. *Brief. Bioinform.*, **11**, 253–264.
22. Brock,G.N., Shaffer,J.R., Blakesley,R.E., Lotz,M.J. and Tseng,G.C. (2008) Which missing value imputation method to use in expression profiles: a comparative study and two selection schemes. *BMC Bioinformatics*, **9**, 12.
23. Liew,A.W.-C., Law,N.-F. and Yan,H. (2011) Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Brief. Bioinform.*, **12**, 498–513.
24. Troyanskaya,O., Cantor,M., Sherlock,G., Brown,P., Hastie,T., Tibshirani,R., Botstein,D. and Altman,R.B. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.
25. Bø,T.H., Dysvik,B. and Jonassen,I. (2004) LSimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Res.*, **32**, e34.
26. Kim,H., Golub,G.H. and Park,H. (2005) Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, **21**, 187–198.
27. Wang,X., Li,A., Jiang,Z. and Feng,H. (2006) Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, **7**, 32.
28. Oba,S., Sato,M., Takemasa,I., Monden,M., Matsubara,K. and Ishii,S. (2003) A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, **19**, 2088–2096.
29. Kim,K.-Y., Kim,B.-J. and Yi,G.-S. (2004) Reuse of imputed data in microarray analysis increases imputation efficiency. *BMC Bioinformatics*, **5**, 160.
30. Celton,M., Malpertuy,A., Lelandais,G. and de Brevern,A.G. (2010) Comparative analysis of missing value imputation methods to

improve clustering and interpretation of microarray experiments. *BMC Genomics*, **11**, 15.
31. de Brevern,A.G., Hazout,S. and Malpertuy,A. (2004) Influence of microarrays experiments missing values on the stability of gene groups by hierarchical clustering. *BMC Bioinformatics*, **5**, 114.
32. Tuikkala,J., Elo,L.L., Nevalainen,O.S. and Aittokallio,T. (2008) Missing value imputation improves clustering and interpretation of gene expression microarray data. *BMC Bioinformatics*, **9**, 202.
33. Wang,D., Lv,Y., Guo,Z., Li,X., Li,Y., Zhu,J., Yang,D., Xu,J., Wang,C., Rao,S. *et al.* (2006) Effects of replacing the unreliable cDNA microarray measurements on the disease classification based on gene expression profiles and functional modules. *Bioinformatics*, **22**, 2883–2889.
34. Oh,S., Kang,D.D., Brock,G.N. and Tseng,G.C. (2011) Biological impact of missing-value imputation on downstream analyses of gene expression profiles. *Bioinformatics*, **27**, 78–86.
35. Melé,M., Ferreira,P.G., Reverter,F., DeLuca,D.S., Monlong,J., Sammeth,M., Young,T.R., Goldmann,J.M., Pervouchine,D.D., Sullivan,T.J. *et al.* (2015) The human transcriptome across tissues and individuals. *Science*, **348**, 660–665.
36. McCall,M.N., Bolstad,B.M. and Irizarry,R.A. (2010) Frozen robust multiarray analysis (fRMA). *Biostatistics*, **11**, 242–253.
37. Dai,M., Wang,P., Boyd,A.D., Kostov,G., Athey,B., Jones,E.G., Bunney,W.E., Myers,R.M., Speed,T.P., Akil,H. *et al.* (2005) Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data. *Nucleic Acids Res.*, **33**, e175.
38. Zhu,Q., Wong,A.K., Krishnan,A., Aure,M.R., Tadych,A., Zhang,R., Corney,D.C., Greene,C.S., Bongo,L.A., Kristensen,V.N. *et al.* (2015) Targeted exploration and analysis of large cross-platform human transcriptomic compendia. *Nat. Methods*, **12**, 211–214.
39. Brown,G.R., Hem,V., Katz,K.S., Ovetsky,M., Wallin,C., Ermolaeva,O., Tolstoy,I., Tatusova,T., Pruitt,K.D., Maglott,D.R. *et al.* (2015) Gene: a gene-centered information resource at NCBI. *Nucleic Acids Res.*, **43**, D36–D42.
40. Tibshirani,R. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Methodol.*, **58**, 267–288.
41. Nguyen,D.V., Wang,N. and Carroll,R.J. (2004) Evaluation of missing value estimation for microarray data. *J. Data Sci.*, **2**, 347-370
42. Hu,J., Li,H., Waterman,M.S. and Zhou,X.J. (2006) Integrative missing value estimation for microarray data. *BMC Bioinformatics*, **7**, 449.
43. Zhang,C.-H. and Huang,J. (2008) The sparsity and bias of the Lasso selection in high-dimensional linear regression. *Ann. Stat.*, **36**, 1567–1594.
44. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R., Dubourg,V. *et al.* (2011) Scikit-learn: machine Learning in Python. *J. Mach. Learn. Res.*,**12**, 2825–2830.
45. Chollet,F. (2015) *Kera*s, https://keras.io.
46. Abadi,M., Agarwal,A., Barham,P., Brevdo,E., Chen,Z., Citro,C., Corrado,G.S., Davis,A., Dean,J., Devin,M. *et al.* (2016) TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv doi: https://arxiv.org/abs/1603.04467, 16 March 2016, preprint: not peer reviewed.
47. Lee,Y., Krishnan,A., Zhu,Q. and Troyanskaya,O.G. (2013) Ontology-aware classification of tissue and cell-type signals in gene expression profiles across platforms and technologies. *Bioinformatics*, **29**, 3036–3044.

A Flexible, Interpretable, and Accurate Approach for Imputing the Expression of Unmeasured Genes - Supplemental Material

Section 1: Supplemental Material for Methods

Section 1.1: Pictorial Representation of Unmeasured Gene Imputation

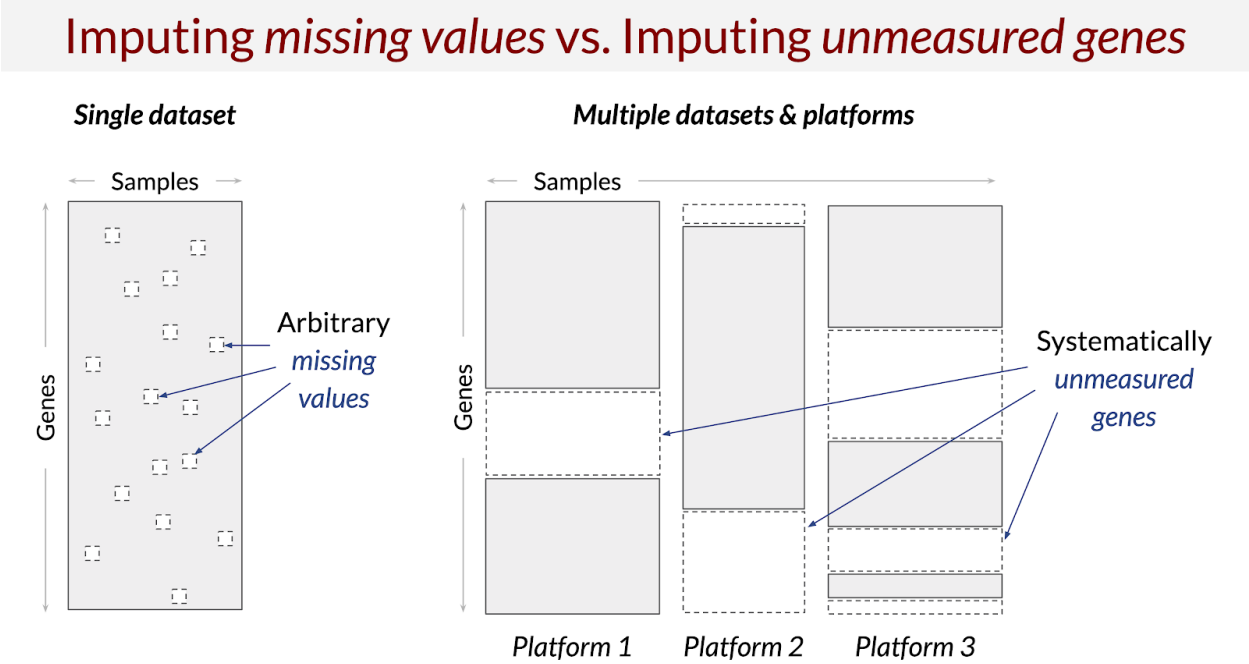

Fig. S1. Schematic of the difference between "missing value" imputation versus "unmeasured gene" imputation. In the missing value imputation problem, gene expression values from a single dataset that were unmeasured due to technical errors are imputed. In the unmeasured gene imputation problem, genes not measured at all by a given platform are imputed.

Section 1.2: Description of data processing steps

We downloaded the microarray data on Dec 6th 2017, and downloaded the RNA-seq data from the Sept 14th 2018 release of ARCHS4. In the microarray data, log transformation and quantile normalization was performed using Frozen Robust Multi-array Analysis (1) and the probes were mapped to Entrez space using a custom CDF (2). For the RNA-seq data, we converted the ENST IDs to Entrez IDs by taking the sum of all ENSTs mapped to a given ENSG, where the ENST to ENSG mapping was given by using the *gene2ensembl.gz* file available on the NCBI website on June 11th 2019. We note that the microarray data can be readily quantile normalized

as all the data comes from the *Affymetrix Human Genome U133 Plus 2.0 Array* platform. In contrast, quantile normalization of the RNA-seq data is not straightforward as the data was generated using many different sequencing platforms, and it is difficult to perform batch effect correction across this large number of expression samples. We mention that although there are 984 “landmark” genes in original LINCS data, the software package we used to map from microarray probes to Entrez gene IDs only contained 964 of the LINCS genes.

The procedure used to split the expression samples into the training, validation and test sets was; 1) samples were first grouped together if they appear in the same experiments, 2) a date is assigned to every one of these experiment groups by selecting the oldest date of an expression sample associated within the group, and 3) the experimental groups are then temporally split into training, validation and testing sets with the oldest experimental groups going into the training set and most recent experimental groups going into the test set. The split ratios were such that we had 80% of the data in the training, 10% of the data in the validation and testing sets. We then further subset the validation to 10% of it’s full size as described in the main text. We note, for the DNN and GAN methods we used all the validation data to most closely replicate the methods reported in (3, 4).

To get an idea if subsetting the validation data was sufficient we plotted the results for both the validation and test test for each method using the optimal hyperparameter [Fig. S2]. For both the Microarray-Microarray and RNAseq-Microarray cases, the performance between the two sets is nearly identical. For the situation where the validation and test data is RNA-seq (RNAseq-RNAseq), the test set performance is slightly worse than the validation set. However, we do not believe this is due to too few validation samples, rather it is due to the fact that since the data is split temporally, this is a batch effect as the sequencing technology is changing over time.

We downloaded the SEEK data (5) from http://seek.princeton.edu/download.jsp. We used all the platforms contained in this database except for *Affymetrix Human Genome U133 Plus 2.0 Array* and *Affymetrix Human Genome U133A Array* as this gene split was already considered in the first part of the manuscript. Detailed information about the SEEK data can be seen in Table S1.

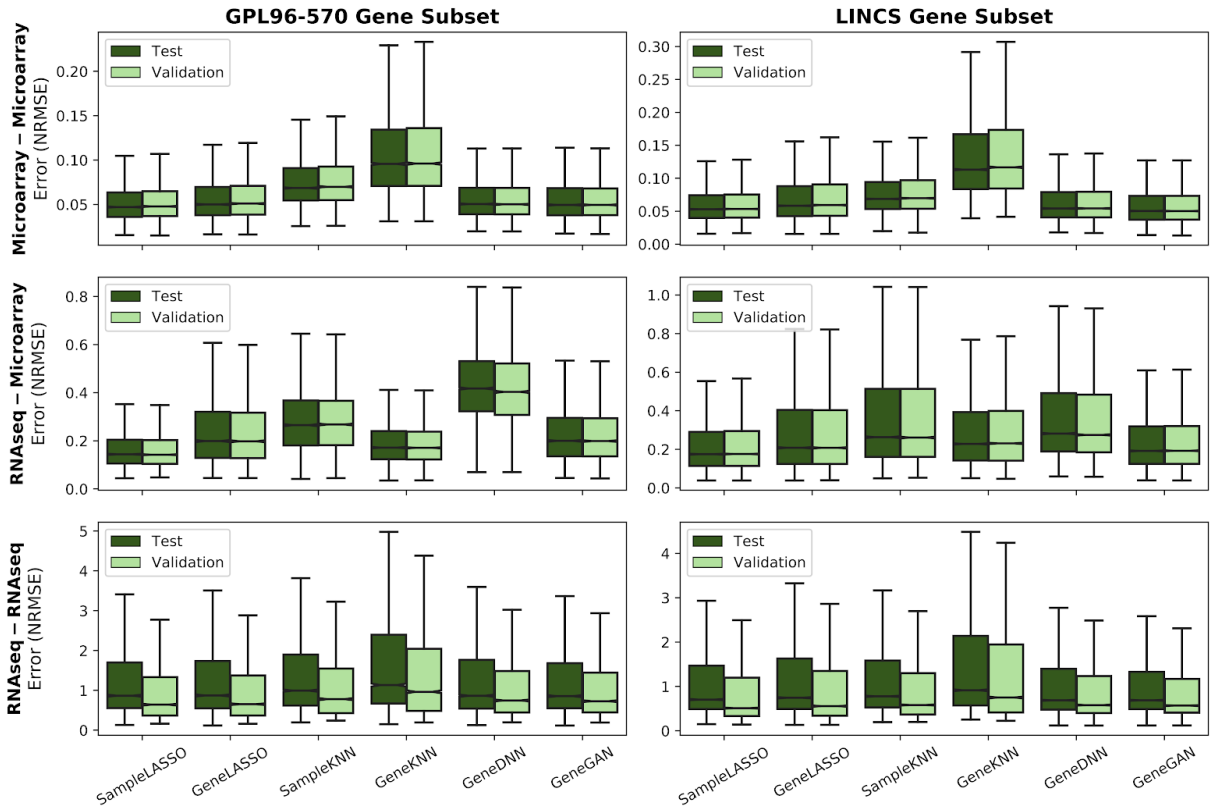

Fig. S2. Comparison of imputation methods across both validation and test sets. The performance of the methods is compared for the validation and test sets to see if using the subset validation set was sufficient. When the validation and testing data is microarray (top and middle rows) the performance is nearly identical. When the validation and test data is RNA-seq (bottom row) the test set performance is slightly worse than the validation set, but this effect is mostly likely due to the validation and test set containing expression samples from different sequencing platforms.

Table S1. Information on the SEEK Datasets

GPL (ID)	GPL (Name)	Total Number of Genes in GPL	Number of Samples for GPL in SEEK	Number of Measured Genes Used for Training
GPL5175	[HuEx-1_0-st] Affymetrix Human Exon 1.0 ST Array [transcript (gene) version]	19124	2407	16098
GPL6244	[HuGene-1_0-st] Affymetrix Human Gene 1.0 ST Array [transcript (gene) version]	19105	9384	16507
GPL571	[HG-U133A_2] Affymetrix Human Genome U133A 2.0 Array	11816	6782	11491
GPL8300	[HG_U95Av2] Affymetrix Human Genome U95 Version 2 Array	8188	2340	7878
GPL6480	Agilent-014850 Whole Human Genome Microarray 4x44K G4112F (Probe Name version)	18438	7815	16369
GPL4133	Agilent-014850 Whole Human Genome Microarray 4x44K G4112F (Feature Number version)	18249	6888	16243
GPL1708	Agilent-012391 Whole Human Genome Oligo Microarray G4112A (Feature Number version)	17311	2509	15798
GPL6884	Illumina HumanWG-6 v3.0 expression beadchip	18130	4679	16001
GPL6947	Illumina HumanHT-12 V3.0 expression beadchip	17053	6548	14772
TCGA	TCGA RNASeq V2 collection	15049	5085	13703

Section 1.3: Description of imputation methods

In this section, Figures S3-S7 provide a pictorial representation of how each imputation method was implemented. Following these figures is a description of all the parameters that were used for the *GeneDNN* and *GeneGAN* method. A schematic of the *GeneGAN* method can be seen in the original publication for that method (4).

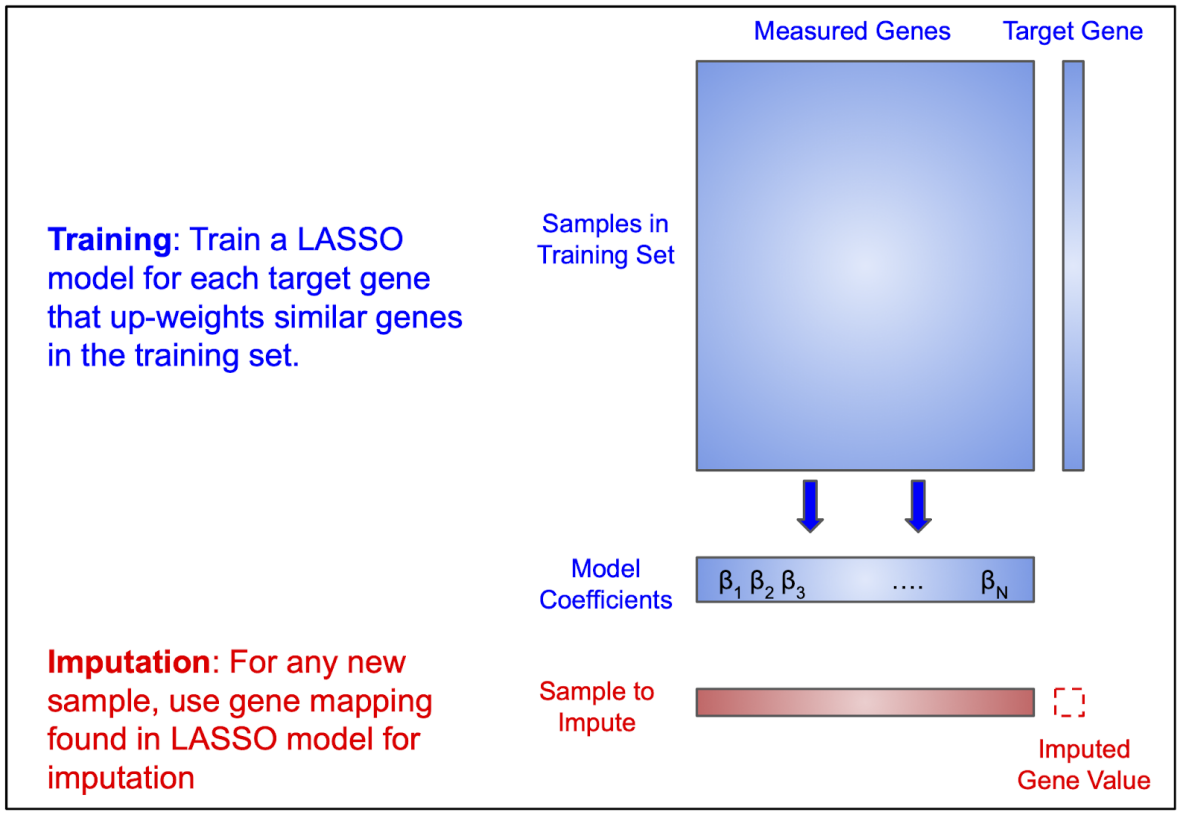

Fig. S3. Schematic of *GeneLASSO*. Blue color denotes training/fitting step and red color denotes imputation step.

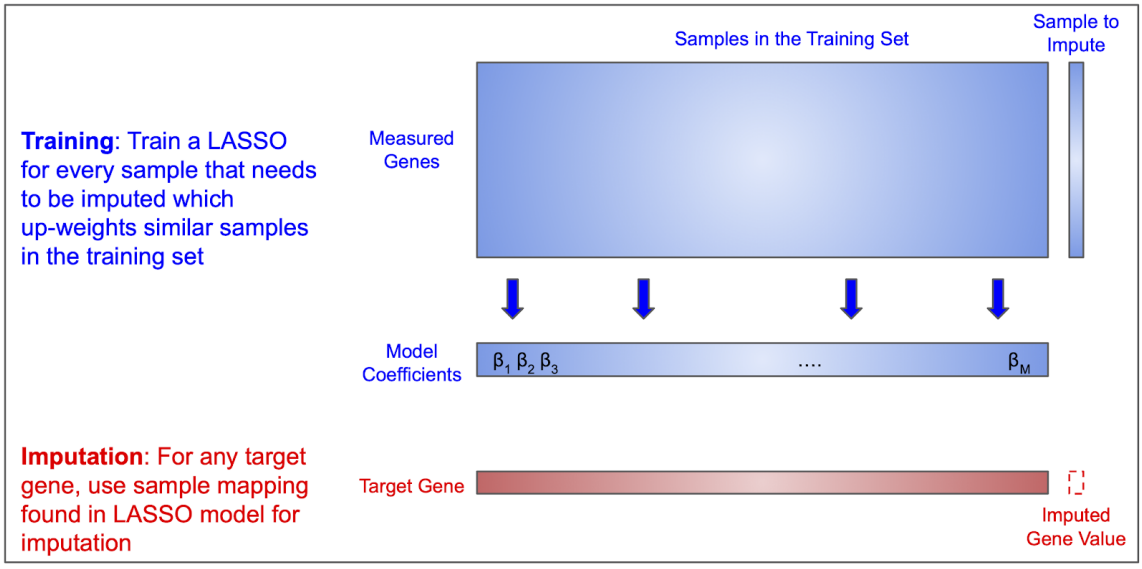

Fig. S4. Schematic of *SampleLASSO*. Blue color denotes training/fitting step and red color denotes imputation step.

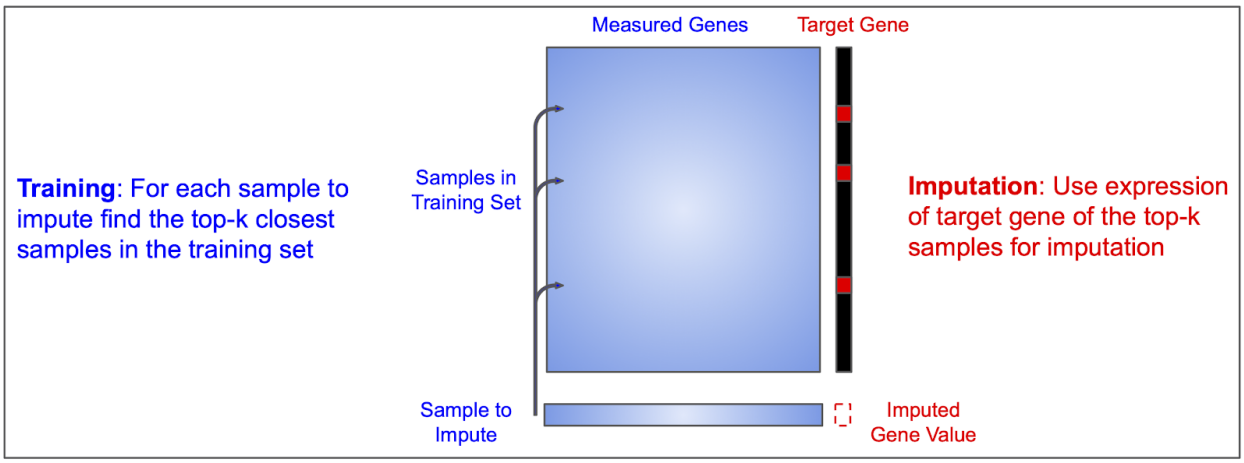

Fig. S5. Schematic of *SampleKNN*. Blue color denotes training/fitting step and red color denotes imputation step.

Fig. S6. Schematic of *GeneKNN*. Blue color denotes training/fitting step and red color denotes imputation step.

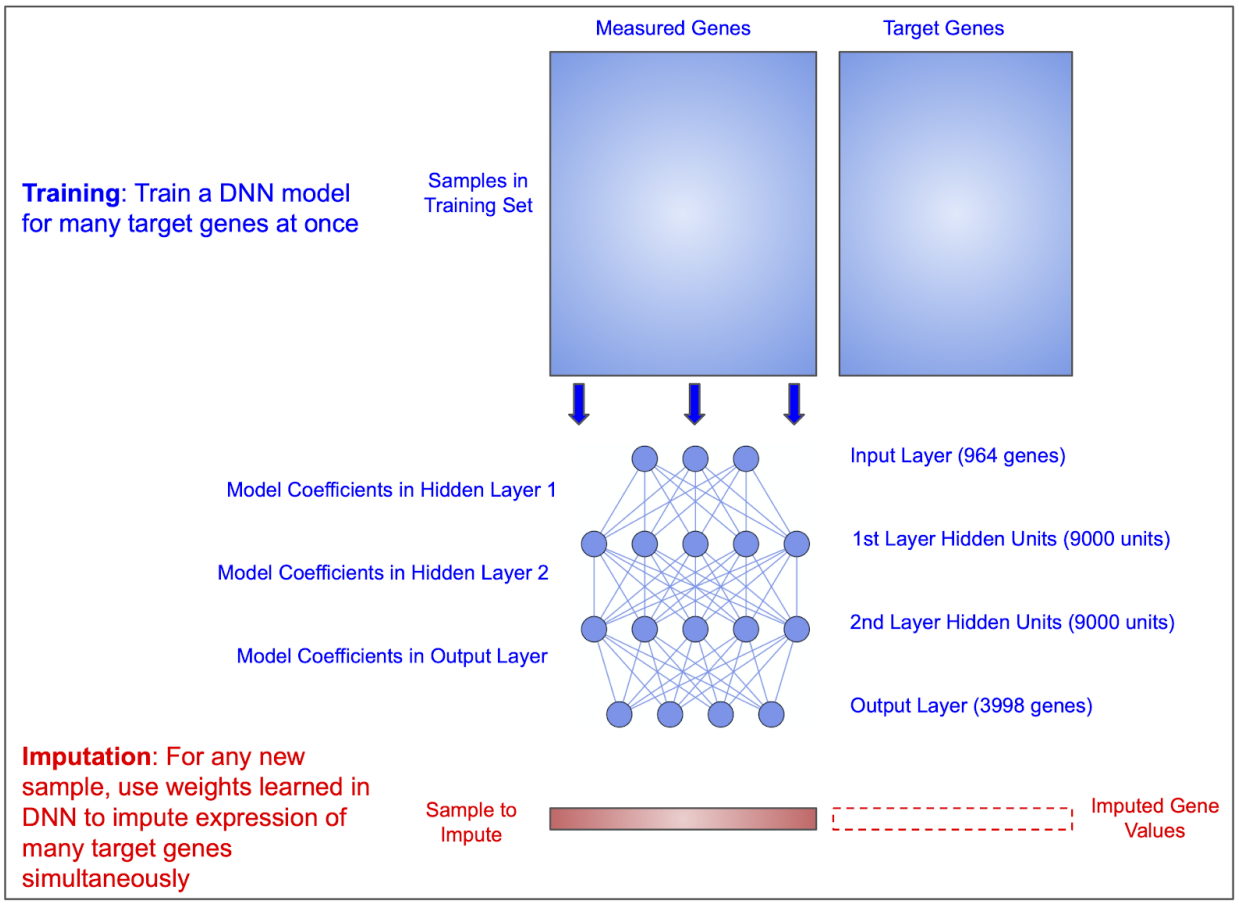

Fig. S7. Schematic of *GeneDNN*. Blue color denotes training/fitting step and red color denotes imputation step. A schematic of the *GeneGAN* method can be seen in Wang et al., 2018.

Parameters for the DNN

The model parameters for the *DNN* used in this work were chosen based on those used by *D-GEX* (3). This includes using a dropout rate of 10%, Xavier Uniform weight initialization (6), a mini-batch size of 200 and 200 training epochs. We chose 2 hidden layers with 9000 units in each layer as this architecture was the best overall for doing same-technology and cross-technology imputation. It was not obvious what exact optimizer was used by (3), so we performed hyperparameter tuning of the optimizer, using Adam (7, 8) and Adadelta (9). We implemented our *DNN* in *Keras* (10) using a *Tensorflow* backend (11).

In the original *D-GEX* paper, due to memory constraints, the target (unmeasured) genes were split into 4 sets, a separate *DNN* was trained for each set, and then the predictions were combined at the end. We implemented both settings: splitting into 4 sets and training 4 models (referred to here as *D-GEX*) as well trained a single *DNN* using all the target (unmeasured) genes in one model (referred to as *GeneDNN*). For all results presented in this work, we use the *GeneDNN* method as: 1) the performance of *D-GEX* and *GeneDNN* are nearly identical [Fig. S8] and 2) using all genes in the target set is how the *GAN* method is implemented. Further, our setup is consistent with growing memory efficiency due to which future implementations of *DNN*s are likely to include as many target genes in one model to increase the performance gained from transfer learning.

Parameters for the GAN

The model parameters for the *GAN* used in this work were chosen based on those used by *GGAN* (4). For the generator, we used a DenseNet architecture with 2 hidden layers with 9000 units each. We note that, in the *GGAN* paper, a three-hidden-layer model performed the best. However, this model would not fit into the memory of our GPUs. For the discriminator, we used a one-hidden-layer network with 3000 hidden units. The loss functions, weight normalization, layer normalization, and hidden layer activation functions are all implemented based on the description in the *GGAN* paper. We did not implement an exponential decay learning rate with Adam as they did in the *GGAN* paper, although one optimizer we did try was Adadelta, which has an adaptive learning rate. We trained each model for 200 epochs or for a maximum time of 48 hours, whichever came first. Since there was no code released with the *GGAN* model, we had to build it from scratch ourselves, and therefore, some differences between our implementation and the original implementation might exist.

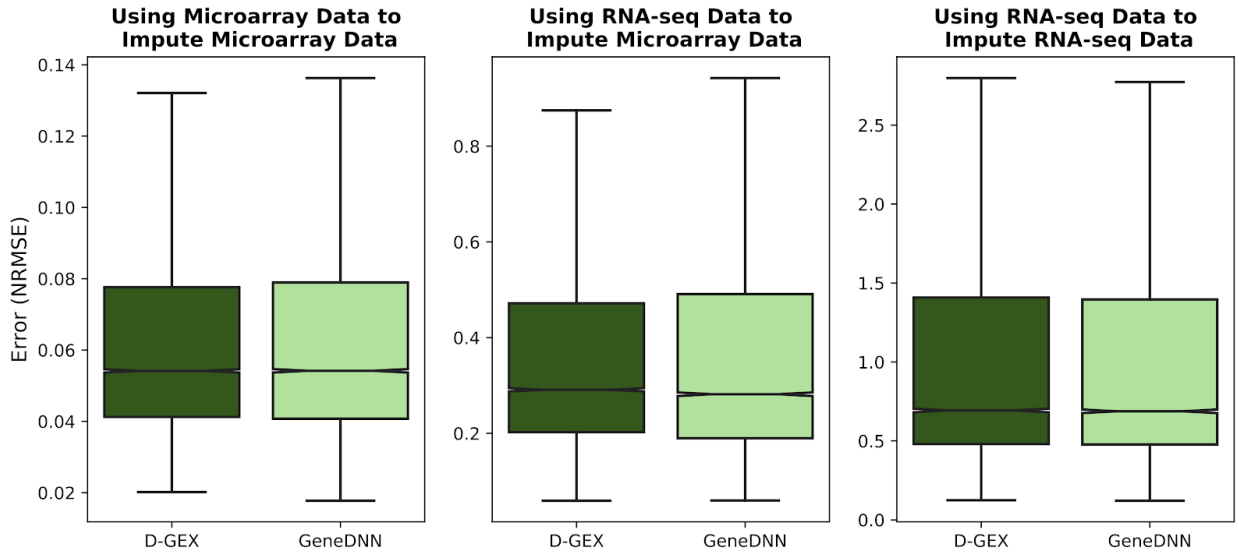

Fig. S8. Performance comparison between *GeneDNN* and *D-GEX*. The performance between *GeneDNN* and *D-GEX* on the LINCS gene split is nearly identical.

Section 1.4: Hyperparameter Tuning

Hyperparameter tuning was performed for all combinations of methods (*SampleLASSO*, *GeneLASSO*, *SampleKNN*, *GeneKNN*, *GeneDNN*, *GeneGAN*, and *D-GEX*), gene subsets (GPL96-570, LINCS) and tasks (Microarray-Microarray, RNAseq-Microarray, RNAseq-RNAseq). In the *LASSO* methods, the hyperparameter that was tuned was the strength of the L1-regularization, referred to as alpha. In the *KNN* methods, the hyperparameter that was tuned was the number of closest samples, referred to a k. In the deep learning methods, the hyperparameter that was tuned was the optimizer and the learning rate (Adadelta does not require a learning rate to be set), and the results show the error using the full validation set after the best model was selected across all epochs. All results are shown for NRMSE. The results of the hyperparameter tuning can be seen in Figs. S9-S15. Table S2 shows the hyperparameter that yielded optimal performance.

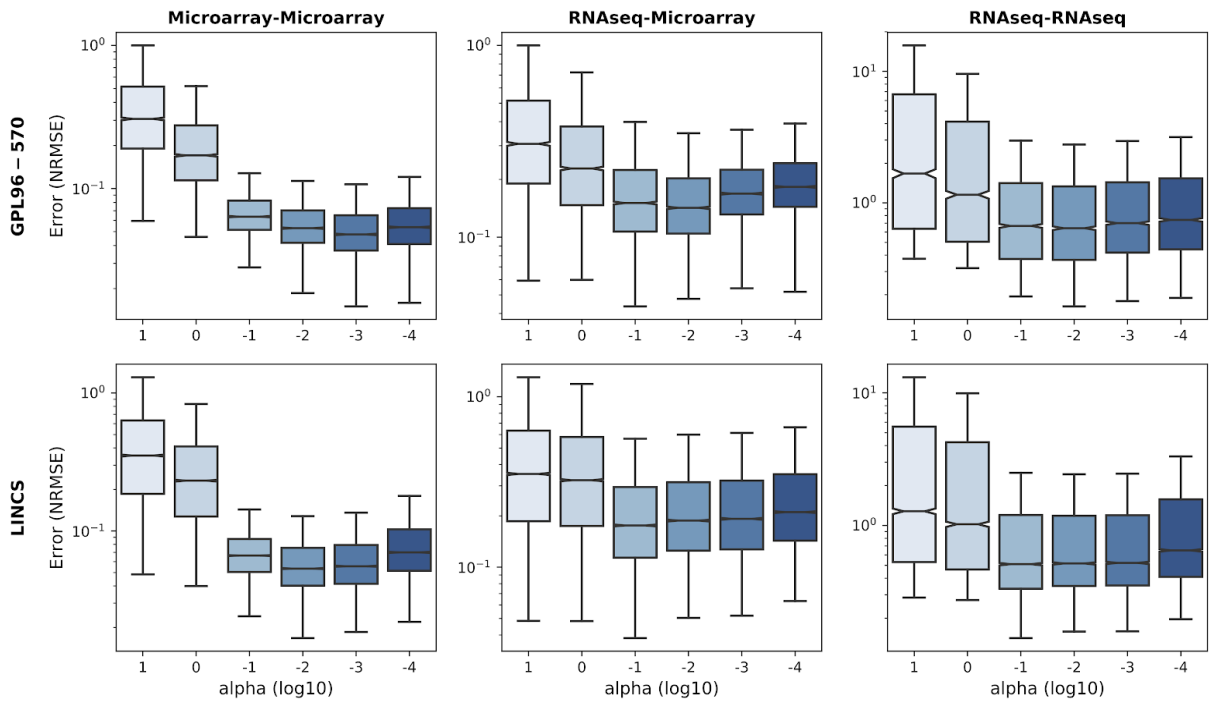

Fig. S9. Hyperparameter tuning for *SampleLASSO*.

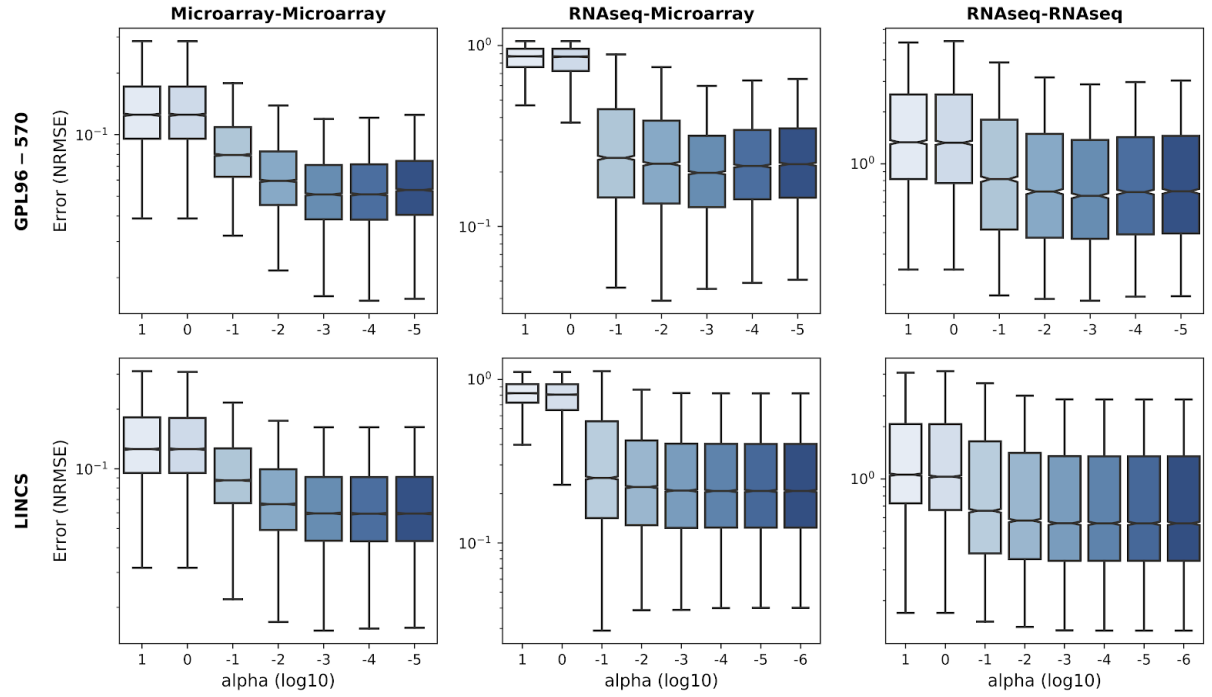

Fig. S10. Hyperparameter tuning for *GeneLASSO*.

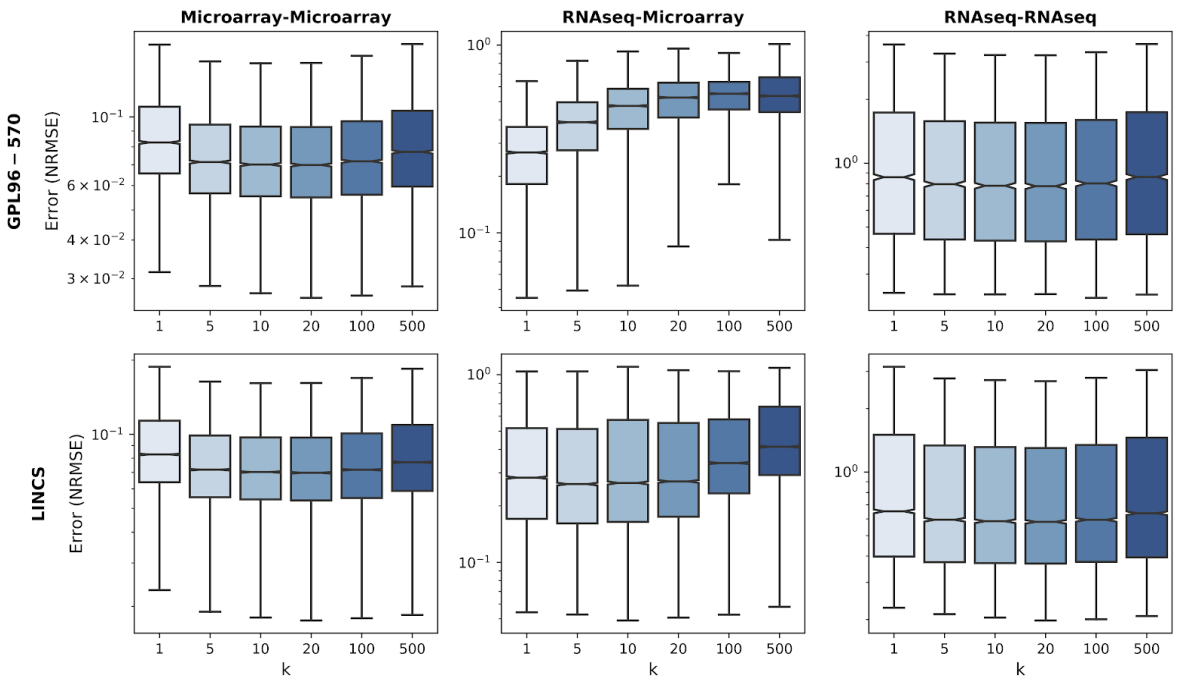

Fig. S11. Hyperparameter tuning for *SampleKNN*.

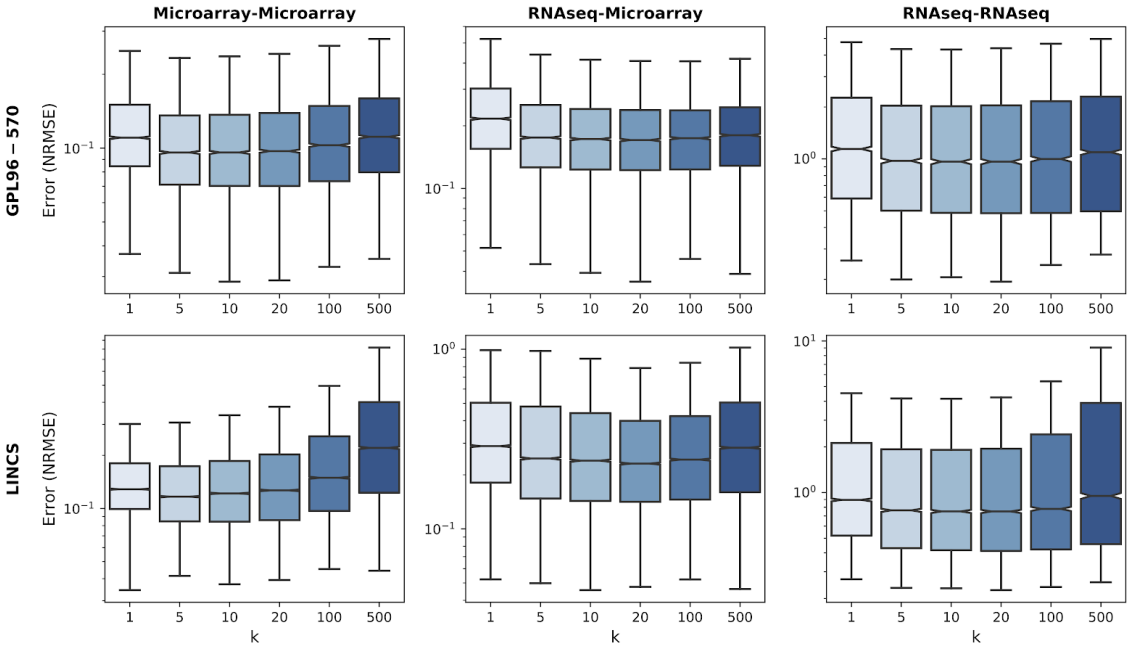

Fig. S12. Hyperparameter tuning for *GeneKNN*.

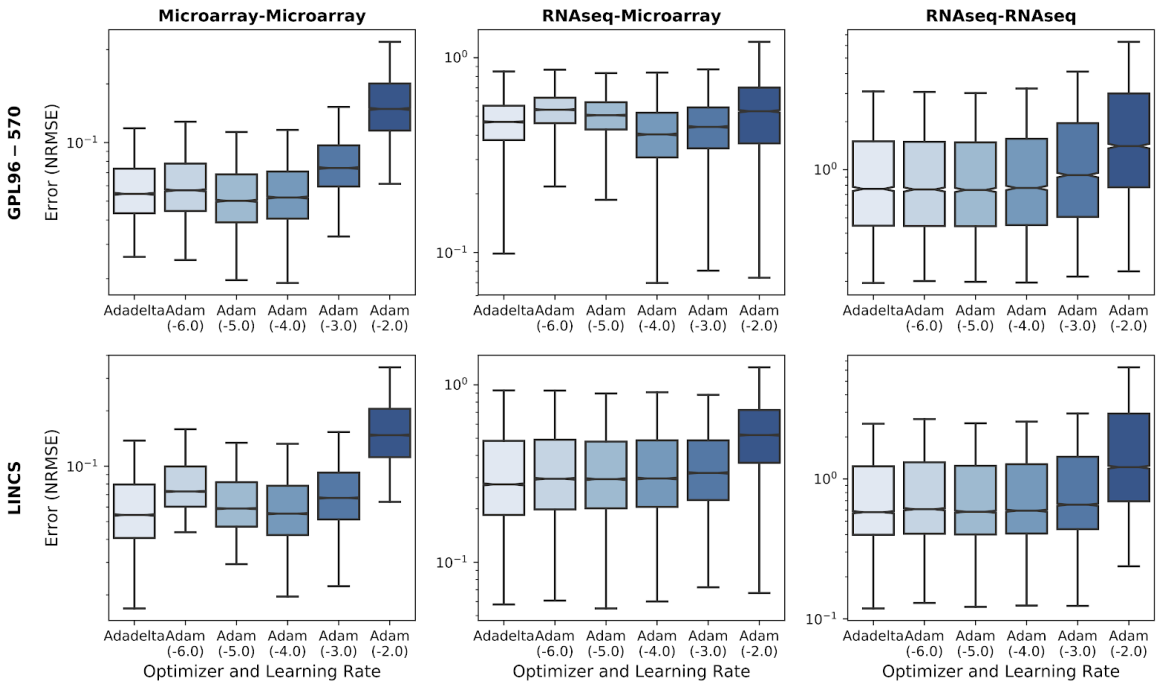

Fig. S13. Hyperparameter tuning for *GeneDNN*.

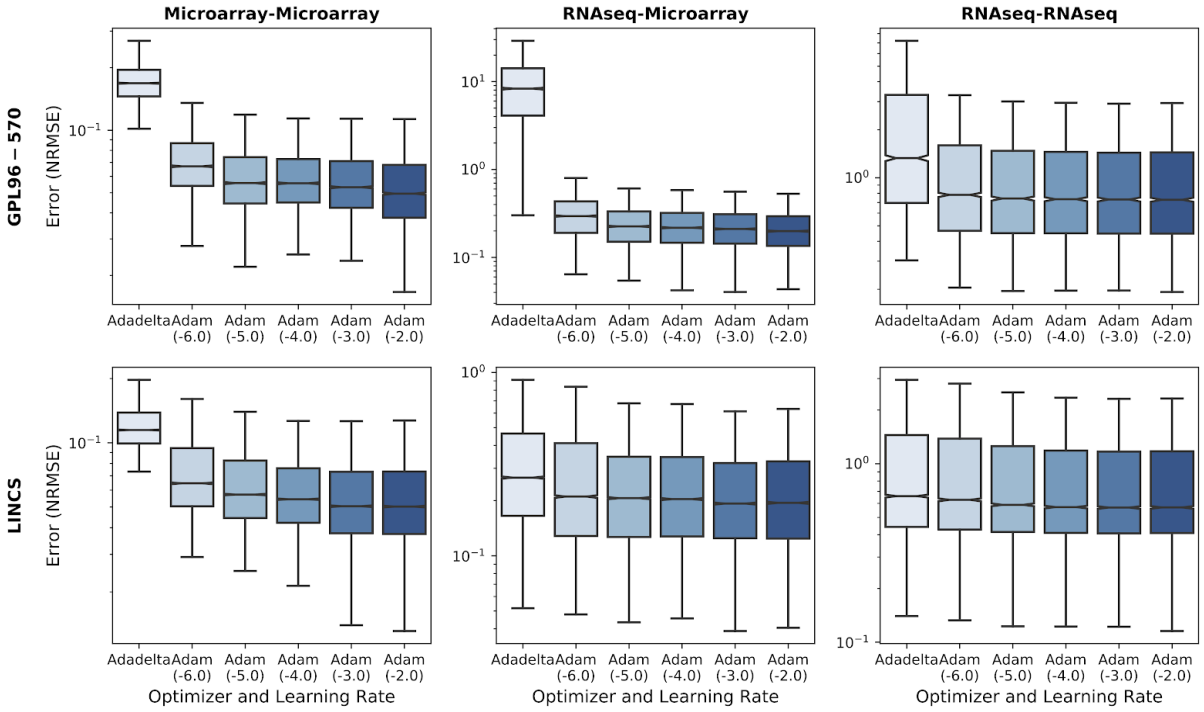

Fig. S14. Hyperparameter tuning for *GeneGAN*.

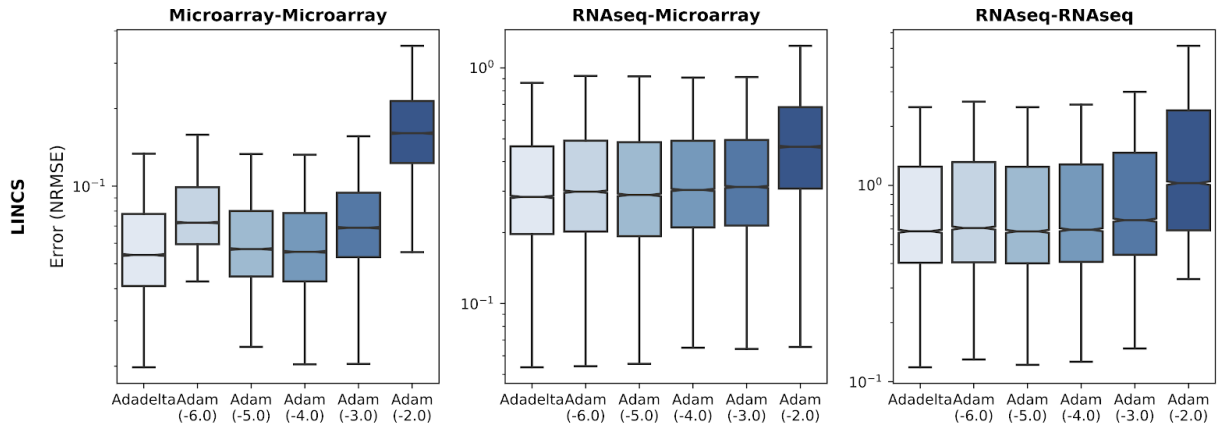

Fig. S15. Hyperparameter tuning for *D-GEX*.

Table S2: Best Models Settings from Hyperparameter Tuning. The NRMSE value is the median NRMSE value across all genes for that hyperparameter set.

		Method	Model Setting	Optimal Choice	NRMSE
Microarray-Microarray	GPL96-570	GeneDNN	optimizer	Adam (1e-05)	0.0502
	GPL96-570	GeneGAN	optimizer	Adam (0.01)	0.0495
	GPL96-570	GeneKNN	k	5.0	0.0961
	GPL96-570	GeneLASSO	alpha	0.001	0.0510
	GPL96-570	SampleKNN	k	20.0	0.0699
	GPL96-570	SampleLASSO	alpha	0.001	0.0478
	LINCS	D-GEX	optimizer	Adadelta	0.0540
	LINCS	GeneDNN	optimizer	Adadelta	0.0543
	LINCS	GeneGAN	optimizer	Adam (0.01)	0.0502
	LINCS	GeneKNN	k	5.0	0.1167
	LINCS	GeneLASSO	alpha	0.0001	0.0593
	LINCS	SampleKNN	k	20.0	0.0698
	LINCS	SampleLASSO	alpha	0.01	0.0534
RNAseq-Microarray	GPL96-570	GeneDNN	optimizer	Adam (0.0001)	0.4033
	GPL96-570	GeneGAN	optimizer	Adam (0.01)	0.1990
	GPL96-570	GeneKNN	k	20.0	0.1708
	GPL96-570	GeneLASSO	alpha	0.001	0.1980
	GPL96-570	SampleKNN	k	1.0	0.2677
	GPL96-570	SampleLASSO	alpha	0.01	0.1422
	LINCS	D-GEX	optimizer	Adadelta	0.2837
	LINCS	GeneDNN	optimizer	Adadelta	0.2748
	LINCS	GeneGAN	optimizer	Adam (0.001)	0.1927
	LINCS	GeneKNN	k	20.0	0.2309
	LINCS	GeneLASSO	alpha	0.0001	0.2078
	LINCS	SampleKNN	k	5.0	0.2612
	LINCS	SampleLASSO	alpha	0.1	0.1757
RNAseq-RNAseq	GPL96-570	GeneDNN	optimizer	Adam (1e-05)	0.7468
	GPL96-570	GeneGAN	optimizer	Adam (0.01)	0.7281
	GPL96-570	GeneKNN	k	20.0	0.9632
	GPL96-570	GeneLASSO	alpha	0.001	0.6533
	GPL96-570	SampleKNN	k	20.0	0.7792
	GPL96-570	SampleLASSO	alpha	0.01	0.6410
	LINCS	D-GEX	optimizer	Adam (1e-05)	0.5833
	LINCS	GeneDNN	optimizer	Adadelta	0.5788
	LINCS	GeneGAN	optimizer	Adam (0.001)	0.5681
	LINCS	GeneKNN	k	20.0	0.7506
	LINCS	GeneLASSO	alpha	1e-05	0.5560
	LINCS	SampleKNN	k	20.0	0.5811
	LINCS	SampleLASSO	alpha	0.1	0.5107

Table S3: Detailed information on comparing the methods. Median: the median NRMSE value across all genes for that method. Percent SL Better: the percentage of times *SampleLASSO* is better than the other method. Log2 Effect Size: the log2 increase of *SampleLASSO* over the other method considering just the median value (a positive value is when *SampleLASSO* is the better performing method). P-Value: the significance between *SampleLASSO* and the other method based on a Wilcoxon rank-sum test.

		Median	Percent SL Better	Log2 Effect Size	P-Value
Microarray-Microarray-GPL96-570	**SampleLASSO**	0.047	N/A	N/A	N/A
	GeneGAN	0.050	0.92	0.073	0.00e+00
	GeneLASSO	0.050	0.91	0.088	0.00e+00
	GeneDNN	0.050	0.95	0.101	0.00e+00
	SampleKNN	0.069	1.00	0.542	0.00e+00
	GeneKNN	0.096	1.00	1.024	0.00e+00
Microarray-Microarray-LINCS	**GeneGAN**	0.050	0.25	-0.073	0.00e+00
	SampleLASSO	0.053	N/A	N/A	N/A
	GeneDNN	0.054	0.77	0.032	0.00e+00
	GeneLASSO	0.058	0.91	0.137	0.00e+00
	SampleKNN	0.069	0.98	0.373	0.00e+00
	GeneKNN	0.113	1.00	1.094	0.00e+00
RNAseq-Microarray-GPL96-570	**SampleLASSO**	0.144	N/A	N/A	N/A
	GeneKNN	0.172	0.70	0.254	4.38e-223
	GeneLASSO	0.199	0.71	0.467	2.21e-299
	GeneGAN	0.200	0.71	0.474	2.13e-261
	SampleKNN	0.265	0.80	0.881	0.00e+00
	GeneDNN	0.417	0.91	1.536	0.00e+00
RNAseq-Microarray-LINCS	**SampleLASSO**	0.175	N/A	N/A	N/A
	GeneGAN	0.192	0.56	0.134	5.28e-91
	GeneLASSO	0.208	0.62	0.251	0.00e+00
	GeneKNN	0.228	0.84	0.384	0.00e+00
	SampleKNN	0.263	0.76	0.589	0.00e+00
	GeneDNN	0.282	0.76	0.688	0.00e+00
RNAseq-RNAseq-GPL96-570	**GeneGAN**	0.854	0.40	-0.019	2.90e-26
	GeneDNN	0.864	0.39	-0.002	3.93e-02
	SampleLASSO	0.866	N/A	N/A	N/A
	GeneLASSO	0.871	0.41	0.009	6.96e-03
	SampleKNN	0.993	0.93	0.198	0.00e+00
	GeneKNN	1.133	0.98	0.388	0.00e+00
RNAseq-RNAseq-LINCS	**GeneDNN**	0.688	0.13	-0.032	0.00e+00
	GeneGAN	0.688	0.31	-0.032	0.00e+00
	SampleLASSO	0.703	N/A	N/A	N/A
	GeneLASSO	0.744	0.62	0.082	0.00e+00
	SampleKNN	0.777	0.84	0.145	0.00e+00
	GeneKNN	0.914	1.00	0.379	0.00e+00

Section 2: Supplemental Results

Section 2.1: Detailed Information on Method Comparisons

Table S3 shows detailed information between the comparison of all the methods.

Section 2.2: Results for Spearman and MAE metrics

In this section, we present the results in terms of the spearman correlation and the mean absolute error (MAE). The spearman correlation for a gene, (g_i), is given by

$$Spearman(g_i) = \frac{cov(\hat{x}_R, x_R)}{\sigma_{\hat{x}_R}\sigma_{x_R}} \qquad \text{eqn. S1}$$

where $\hat{x}_R$ and x_R are the imputed and real values for all the samples converted to ranks, respectively, cov is the covariance and σ is the standard deviation. The mean absolute error for a gene, (g_i), is given by

$$MAE(g_i) = \sum_{j=1}^{S} \left| \hat{g}_{i,j} - g_{i,j} \right| / S \qquad \text{eqn. S2}$$

Where S is the number of samples, and $\hat{g}_{i,j}$, $g_{i,j}$ are the imputed and real expression values, respectively, for the i^{th} gene in the j^{th} sample.

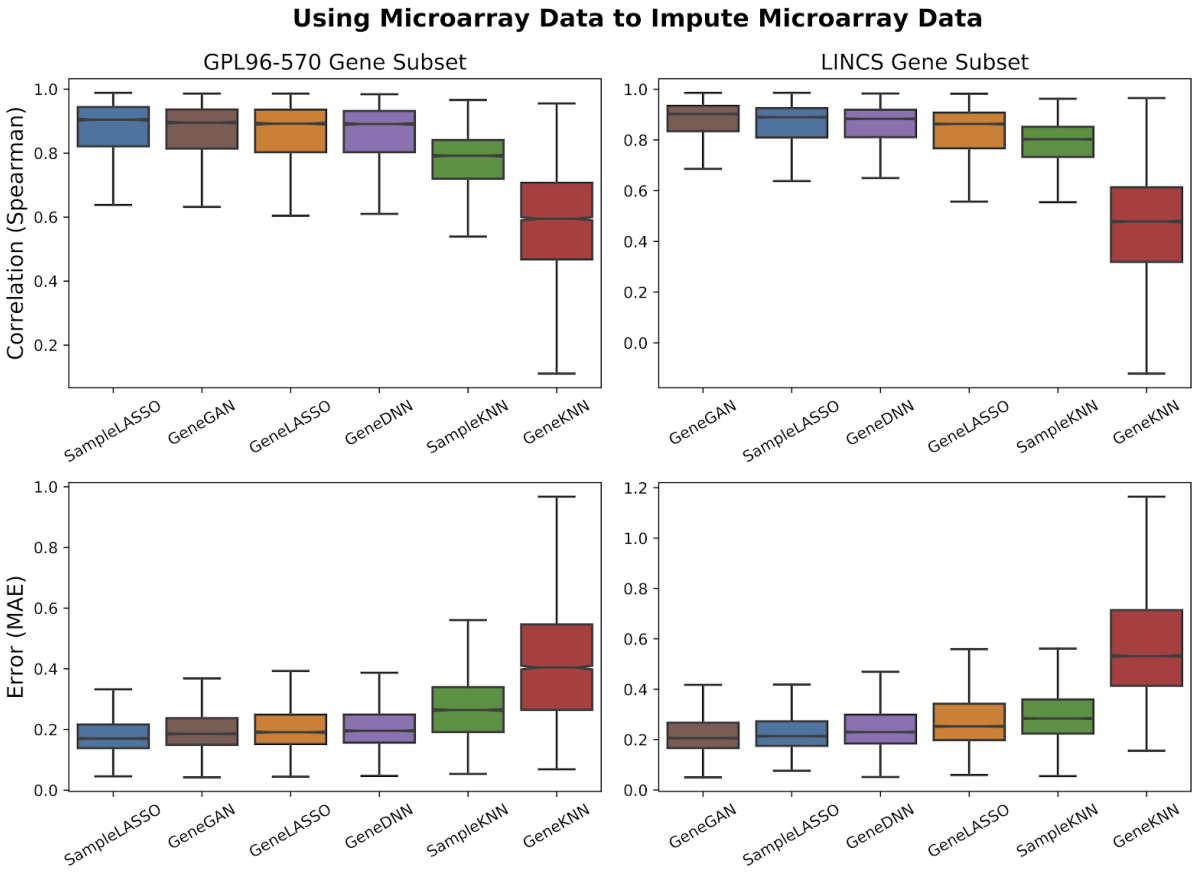

Fig S16. Performance of imputation models on microarray data with Spearman and MAE metrics. The performance of the six methods imputation models (*SampleLASSO*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*, *GeneGAN*) are compared for using microarray data to impute microarray data.

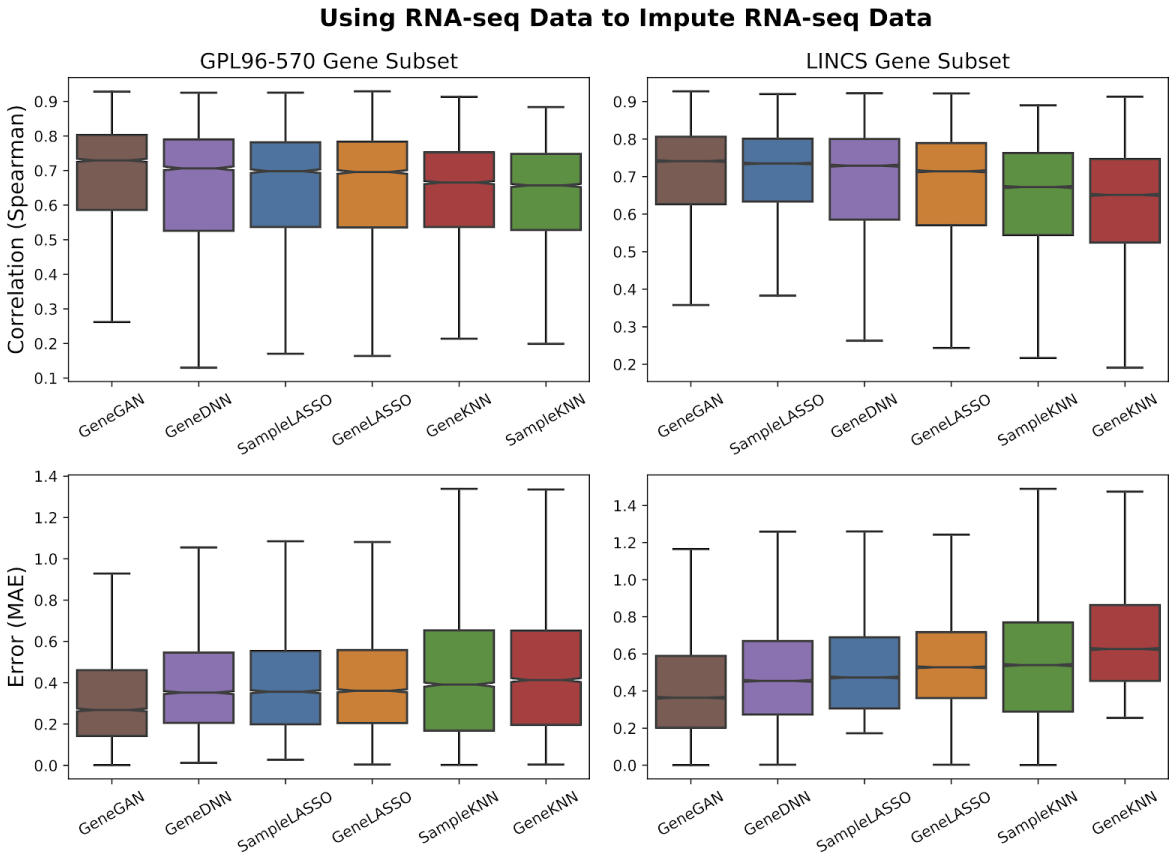

Fig S17. Performance of imputation models on RNA-seq data with Spearman and MAE metrics. The performance of the six methods imputation models (*SampleLASSO*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*, *GeneGAN*)) are compared for using RNA-seq data to impute RNA-seq data.

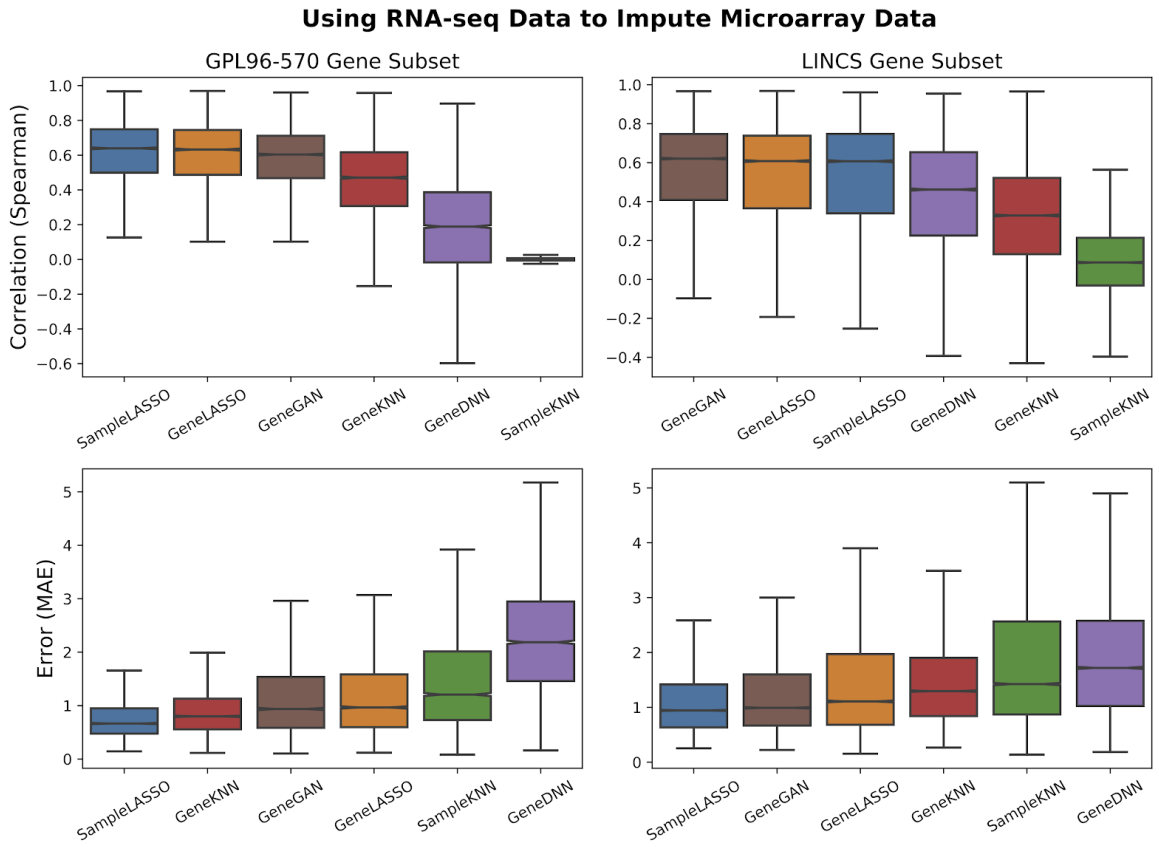

Fig S18. Performance of imputation methods for cross-technology imputation with Spearman and MAE metrics. The performance of the six methods imputation models (*SampleLASSO*, *GeneDNN*, *GeneLASSO*, *SampleKNN*, *GeneKNN*, *GeneGAN*) are compared for using RNA-seq data to impute microarray data.

Section 2.3: Effects of normalization on imputation

To assess how applying a basic normalization technique can improve imputation accuracy, we jointly normalized the data used for the task of using RNA-seq data to impute microarray data. For this analysis, we first quantile normalized the microarray test set data and then transformed the RNA-seq training data to this space. We then compared using this normalization method to only considering quantile normalizing the microarray test set data. It was necessary to perform the quantile normalization on the microarray test set data in the “No Normalization” setting to allow for the imputed values to be equivalent between both imputation settings. The imputation was done for 1000 random samples in the test set for the LINCS gene subset. We see that performing a joint quantile normalization between the RNA-seq and microarray data increases the performance substantially [Fig. S19]. We note that even for the “No Normalization” case, the errors are higher than those observed in Fig. 3. This could be due to the fact the test data used in this analysis and that used in the main manuscript are on different scales due to the quantile normalization being performed on the test set in this analysis.

Although this type of joint normalization has been done in all recent imputation studies involving data from different platforms and technologies, we note that this normalization procedure cannot be applied to real-world imputation in a straightforward manner. The reason for this is that, in a real imputation case, the unmeasured genes in the samples to be imputed are actually not known. Whereas in an evaluation study, we create mock ‘unmeasured genes’ and use their known ground truth values of the unmeasured genes to evaluate the imputation methods. In addition to being able incorporate this distribution of expression values from the unmeasured genes into normalization methods, an interesting avenue of future exploration will be in designing normalization methods that could either transform the samples to be imputed into the space of the training set samples, or vice versa.

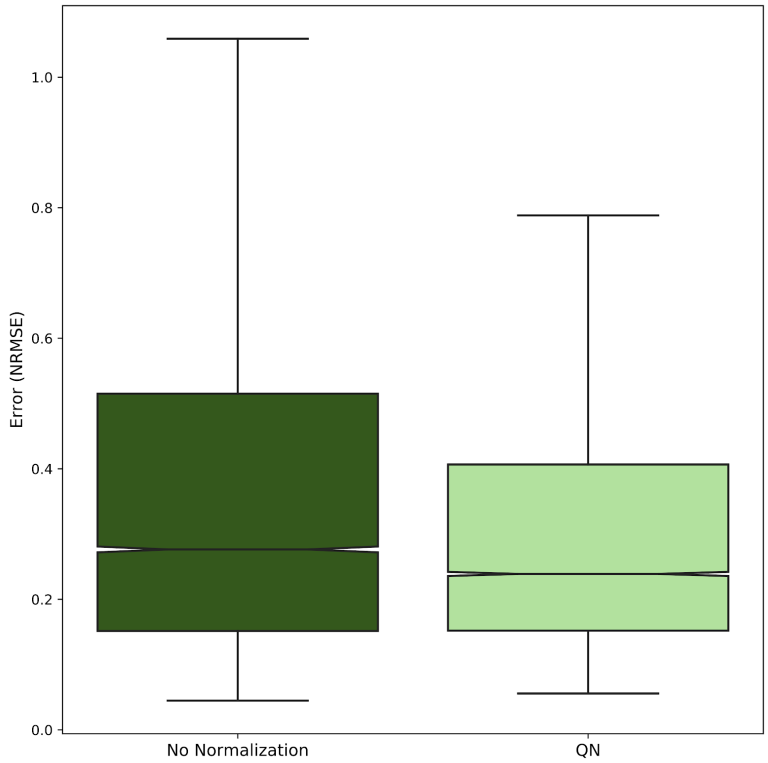

Fig S19. Performance comparison for using normalization and no normalization. The task is using RNA-seq data to predict microarray data and the gene subset is LINCS. QN (light green) refers to the case of performing joint quantile normalization across the RNA-seq and microarray datasets.

Section 2.4: Evaluations considering the expression levels and variance

In this section, we analyze how the performance of the methods changes for two gene properties; the mean expression of a gene and the variance of the expression. Genes were split up into low, medium and high bins for each property, and each panel in a figure [Figs. S20-S25] is the intersection of genes included in the two bins. See the plotting notebook in the associated

GitHub repo (https://github.com/krishnanlab/Expresto) for the breakdown of mean and variance values in each bin as well as the number of points in each boxplot.

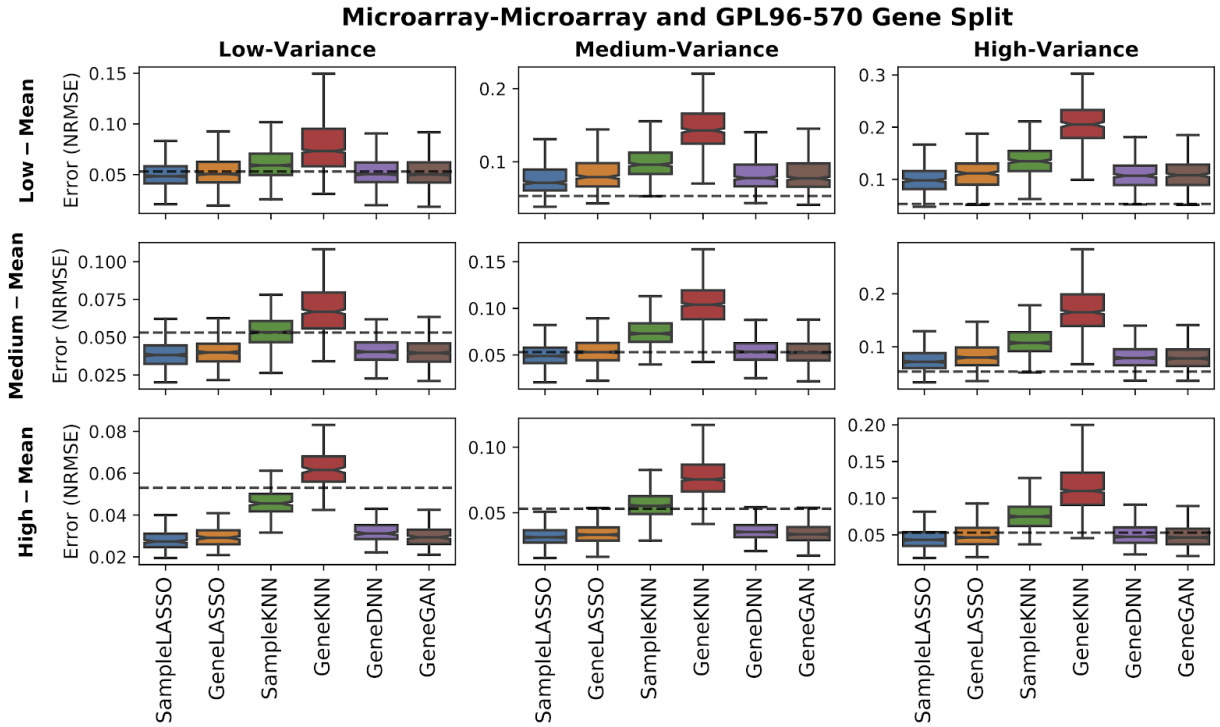

Fig. S20. Results broken up by mean and variance of gene expression for using microarray data to impute microarray data for the GPL96-570 gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

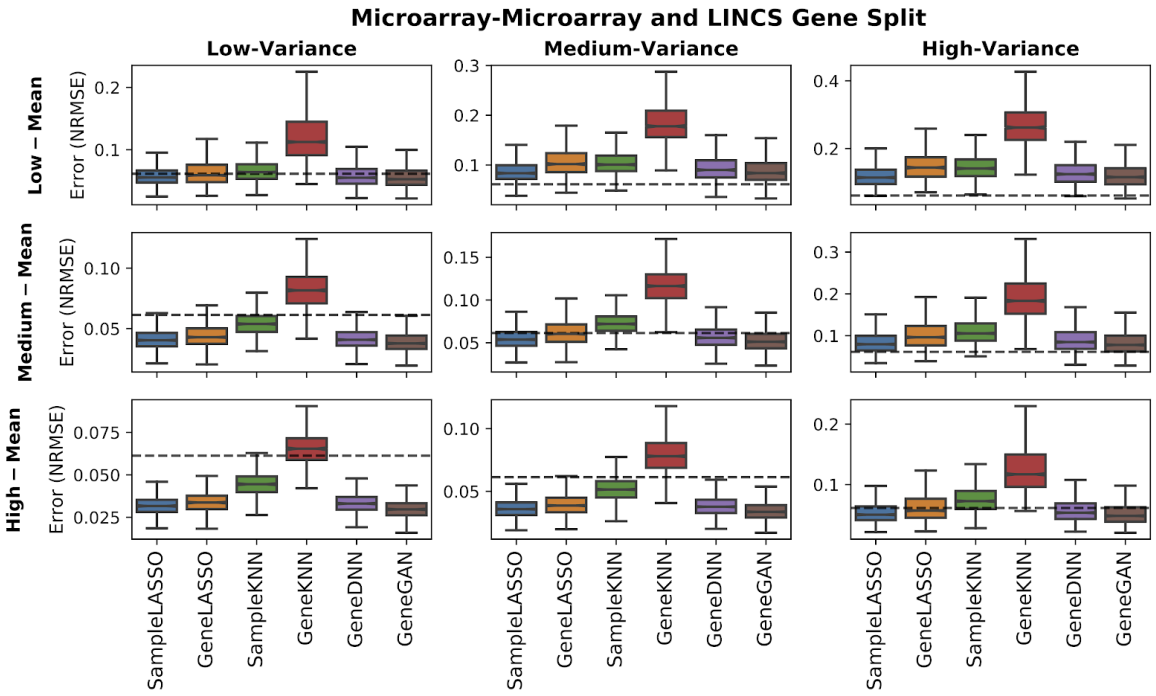

Fig. S21. Results broken up by mean and variance of gene expression for using microarray data to impute microarray data for the LINCS gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

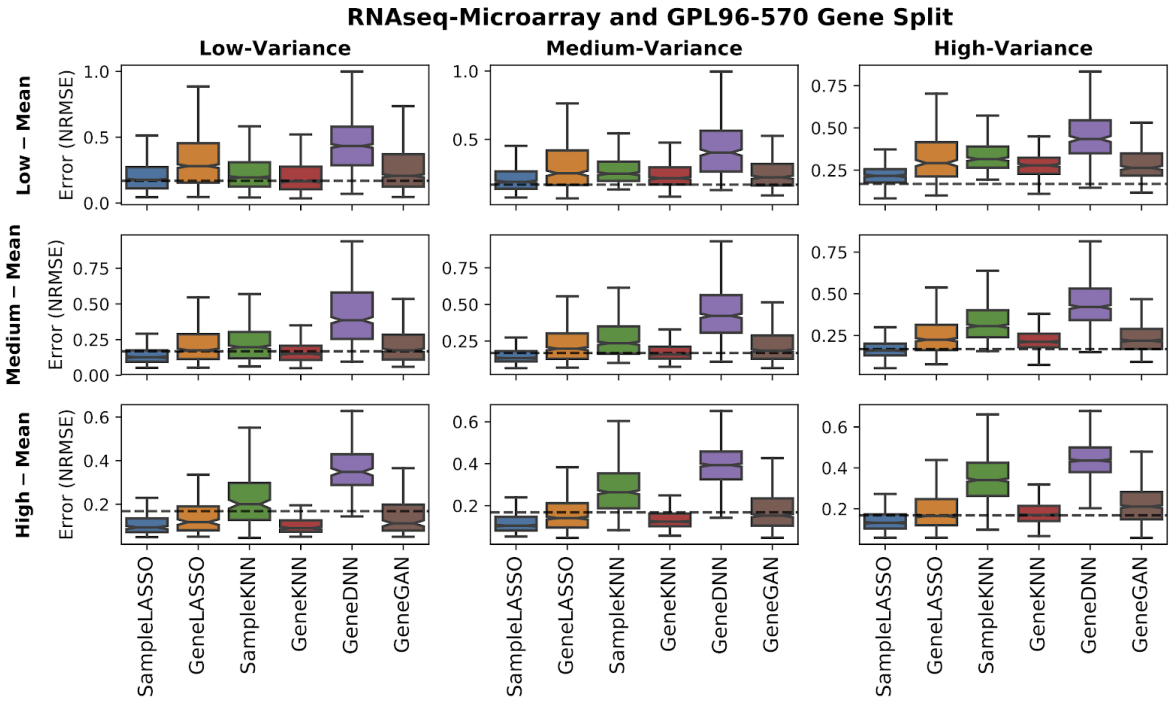

Fig. S22. Results broken up by mean and variance of gene expression for using RNA-seq data to impute microarray data for the GPL96-570 gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

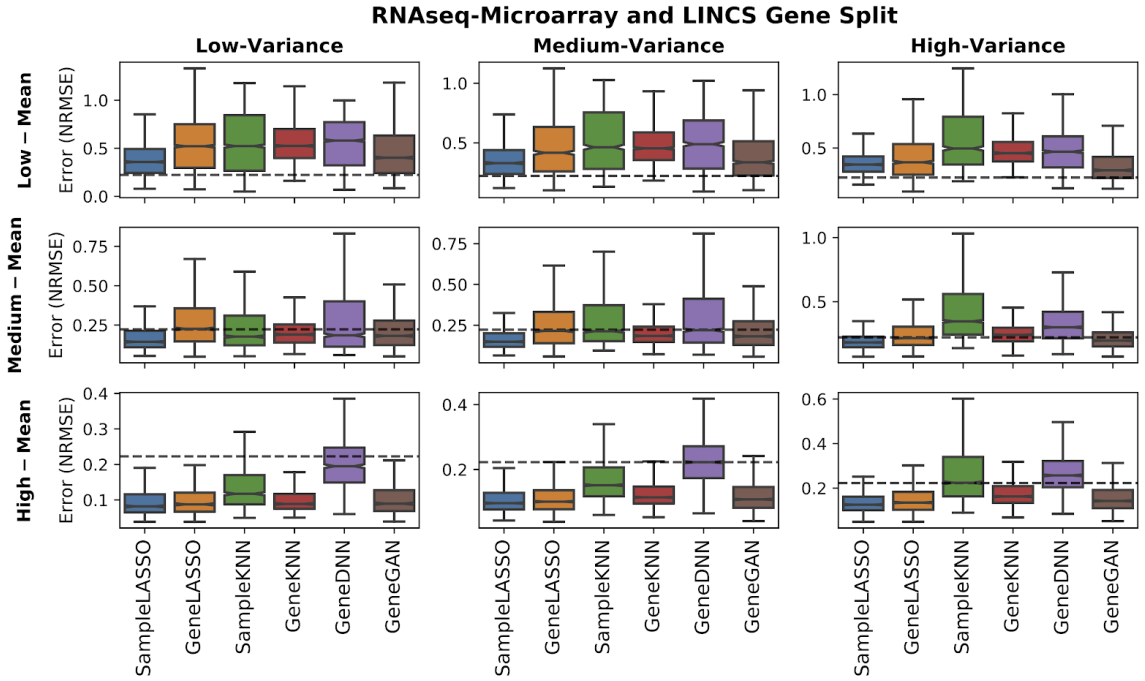

Fig. S23. Results broken up by mean and variance of gene expression for using RNA-seq data to impute microarray data for the LINCS gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

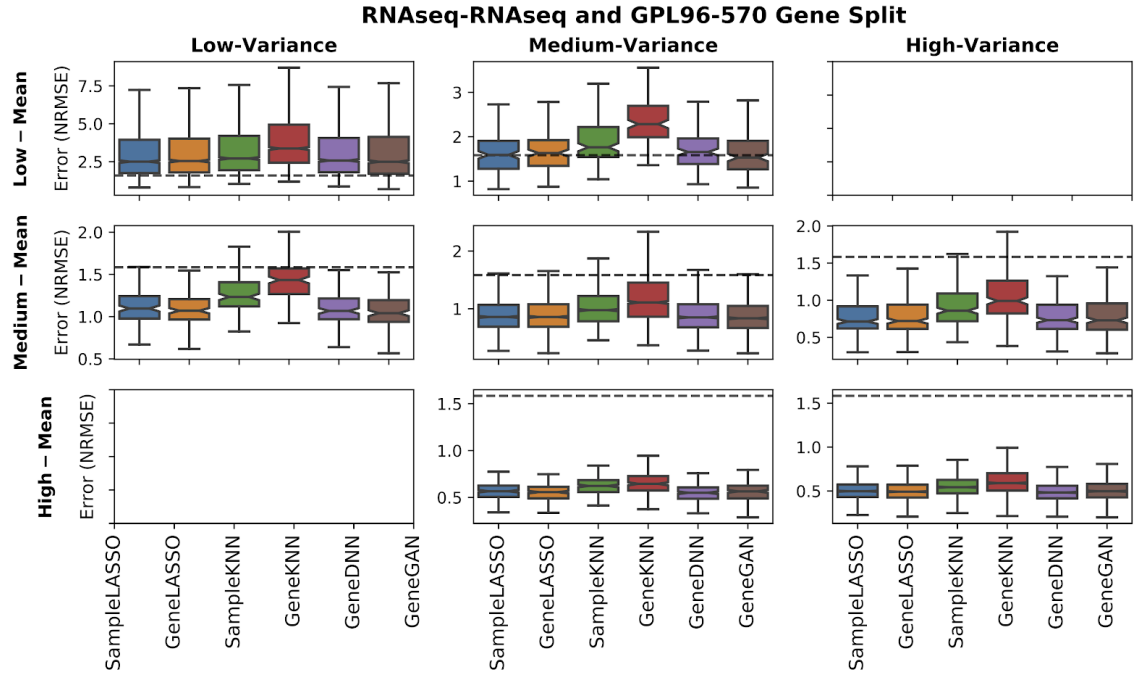

Fig. S24. Results broken up by mean and variance of gene expression for using RNA-seq data to impute RNA-seq data for the GPL96-570 gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

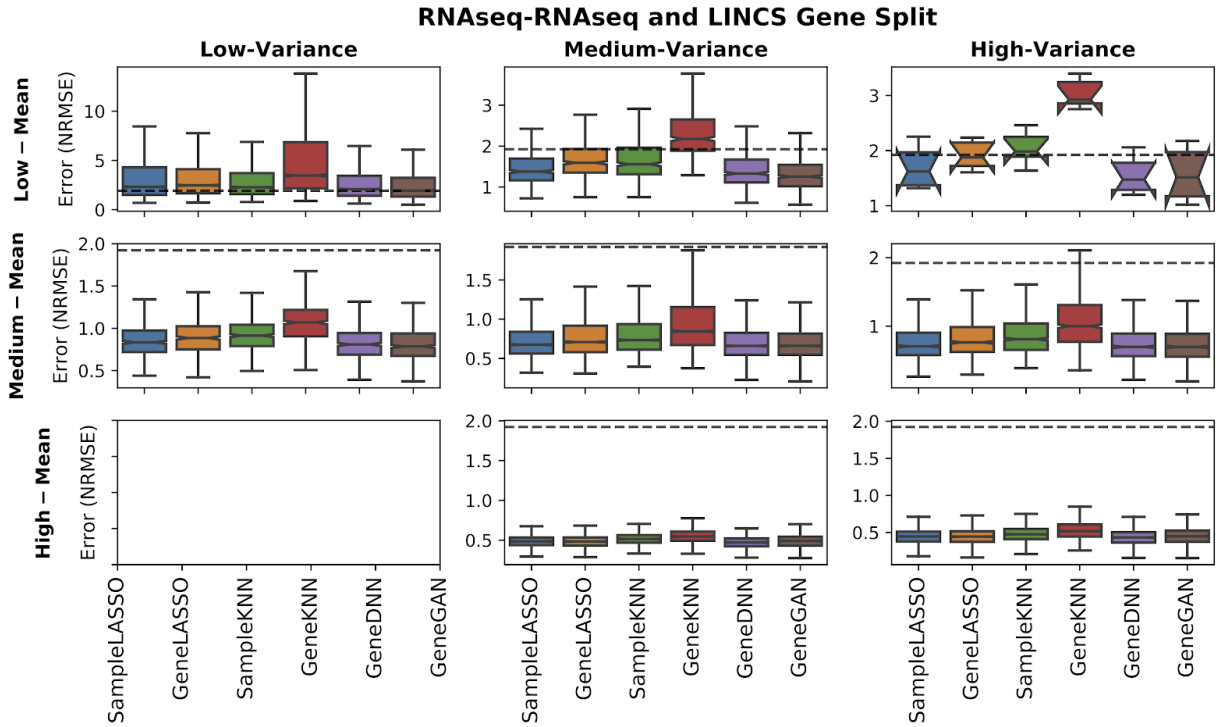

Fig. S25. Results broken up by mean and variance of gene expression for using RNA-seq data to impute RNA-seq data for the LINCS gene subset. The dotted line is the median value when considering all genes for *SampleLASSO* (this is to help compare performances across the panels).

Section 2.5: Supplemental Material for SEEK Analysis

To illustrate how much information is lost if we were to only consider the common genes when jointly analyzing data from two different platforms, we found the number of genes that would be thrown away when combining any two platforms in the SEEK database (i.e. the union minus the intersection of the genes in the two platforms). As can be seen in Figure S26, this usually results in losing information corresponding to thousands of genes.

As it was unfeasible to perform hyperparameter tuning for all methods for all ten platforms, we chose a single set of parameters to use for each method on all ten platforms. The hyperparameters were chosen by looking at all the results from the hyperparameter tuning analysis described in Section 1.4, and selecting the hyperparameter(s) that performed the best across all tasks and gene subsets for each method. This resulted in *k* being 10 for both *SampleKNN* and *GeneKNN*, alpha being 0.01 and 0.001 for *SampleLASSO* and *GeneLASSO*, respectively. For both deep learning methods the optimizer was set to Adam with a learning rate of 10^{-5} and 10^{-3} for *GeneDNN* and *GeneGAN*, respectively.

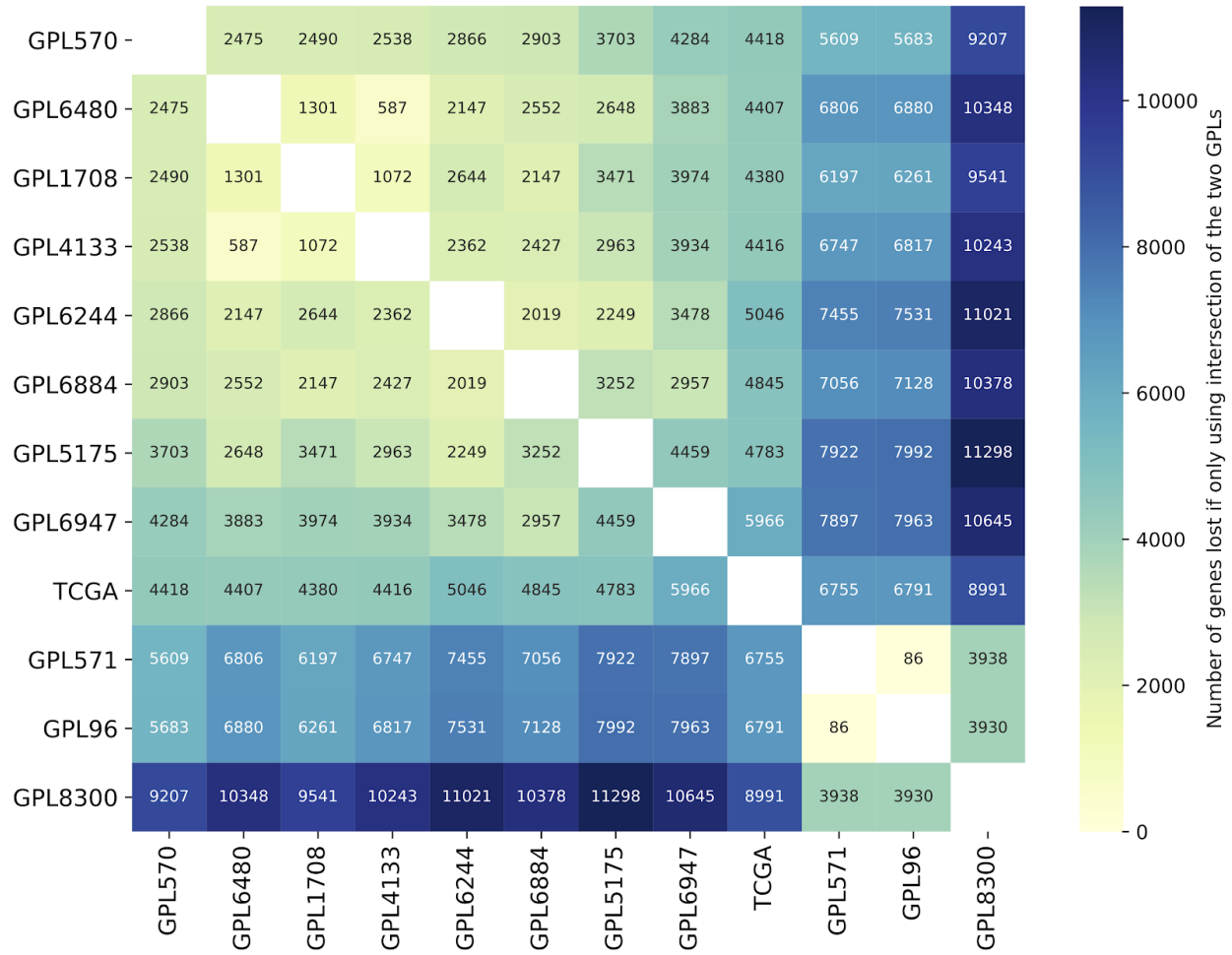

Fig. S26. Information lost when restricting analysis to genes common between two platforms. The values of the heatmap represent the union minus the intersection of the genes contained in a given pair of expression platforms from the SEEK database.

Section 2.6: Supplemental Material for Beta Analysis

The breakdown of the number of how many samples for each tissue were used in the beta-coefficient analysis can be found in Table S4. We additionally analyzed the beta-coefficients by grouping the z-scores of all non-target tissue samples together into a non-target group, as well as grouping the z-scores for all target tissue samples together. This was done for each tissue separately [Fig. S27]. An example of information returned by the Expresto software (https://github.com/krishnanlab/Expresto) can be seen in Table S5.

Table S4. Statistics of Data Used in Beta-Coefficient Analysis

	Test Set		Training Set	
Tissue	**Number of GSMs**	**Number of GSEs**	**Number of GSMs**	**Number of GSEs**
Blood	63	5	1197	33
Liver	40	6	776	27
Breast	40	6	770	25
Brain	39	3	752	19
Lung	29	6	662	10
Kidney	11	3	240	6

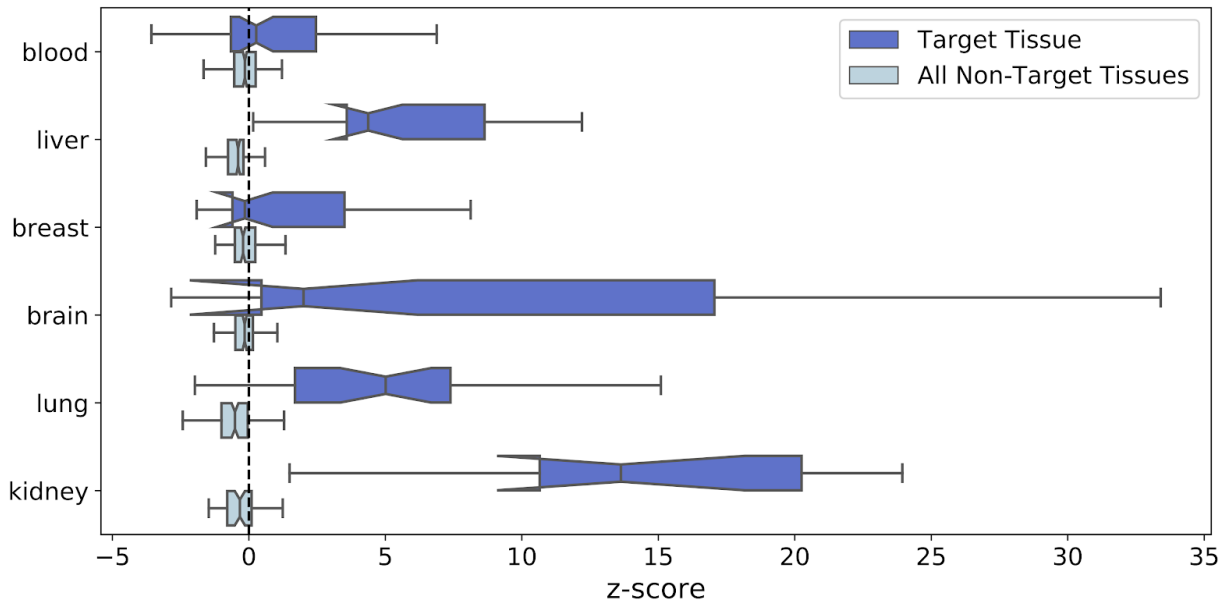

Fig. S27. Model interpretability of the target tissue versus rest of tissues combined. For each test sample labeled for a given tissue, we found the z-score for all six tissues considered. The boxplots show the distribution of z-scores for the target tissue, as well as all non-target tissues.

Table S5. Example of information that can be obtained from the user_function in the *Expresto* software released with this work

Target sample being imputed		Training samples with the three highest model coefficients			
			First	Second	Third
Sample **Study** **Annotation**	GSM478457 GSE19279/GSE19281 Title: Normal pancreas, 3 (U133A)	**Sample** **Study** **Beta-coeff** **Annotation**	GSM175950 GSE7307 0.34 Title: Pancreas SG1 Normal	GSM388108 GSE15471 0.15 Source name: pancreas	GSM388111 GSE15471 0.14 Source name: pancreas
Sample **Study** **Annotation**	GSM664048 GSE26971 Source name: primary breast cancer sample, fresh-frozen	**Sample** **Study** **Beta-coeff** **Annotation**	GSM102499 GSE2109 0.15 Title: Breast - 129692	GSM151315 GSE6532 0.11 Source name: breast	GSM687049 GSE27830/GSE54219 0.09 Title: primary breast cancer, sample_1927
Sample **Study** **Annotation**	GSM4005 GSE475 Source name: Human diaphragm	**Sample** **Study** **Beta-coeff** **Annotation**	GSM342677 GSE13070 0.15 Source name: skeletal muscle (vastus lateralis)	GSM42736 GSE2328 0.13 Source name: skeletal muscle	GSM342884 GSE13070 0.12 Source name: skeletal muscle (vastus lateralis)

Section 2.7: Loss Curves for *GeneDNN*

In this section, we present the loss curves for the *GeneDNN* [Fig. S28]. The data was generated using the *csv_logger* callback in *Keras* using the mean absolute error across all examples of the output of the mode. For all plots the training error displays the expected trends. Two interesting observations are for using RNA-seq data to impute RNA-seq data, the validation data has a lower loss than the training data. When using RNA-seq data to impute microarray data, the validation loss is much higher than the training loss, suggesting that the model is overfitting to the training data. We note that we do not show the loss curves for *GeneGAN* as, when we were writing the code for the *GeneGAN* method, we recorded these training loss curves incorrectly.

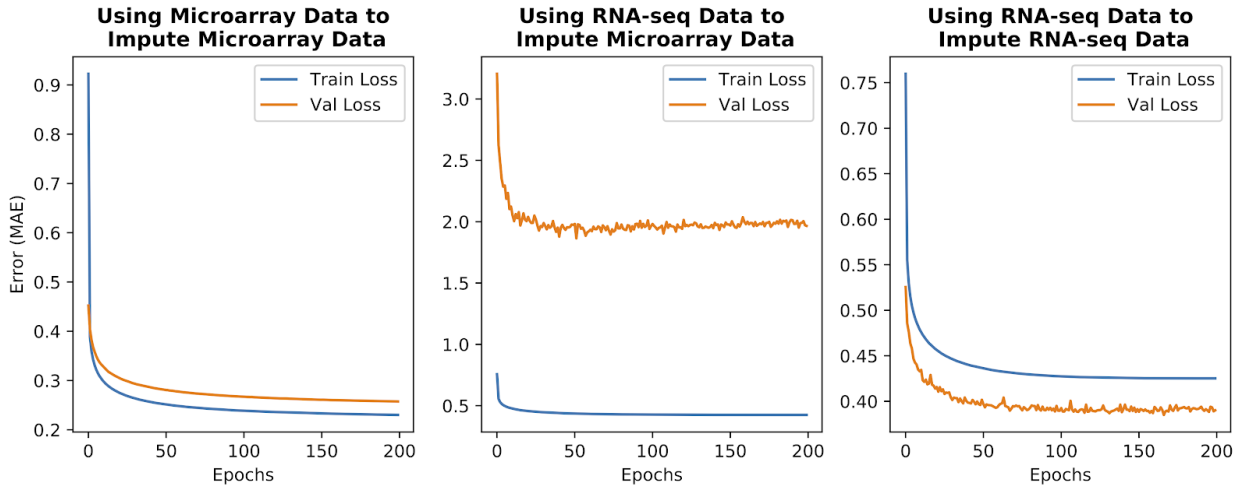

Fig S28. Loss curves the *GeneDNN* method on the LINCS gene subset.

References

1. McCall,M.N., Bolstad,B.M. and Irizarry,R.A. (2010) Frozen robust multiarray analysis (fRMA). *Biostatistics*, **11**, 242–253.
2. Dai,M., Wang,P., Boyd,A.D., Kostov,G., Athey,B., Jones,E.G., Bunney,W.E., Myers,R.M., Speed,T.P., Akil,H., *et al.* (2005) Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data. *Nucleic Acids Res.*, **33**, e175–e175.
3. Chen,Y., Li,Y., Narayan,R., Subramanian,A. and Xie,X. (2016) Gene expression inference with deep learning. *Bioinformatics*, **32**, 1832–1839.
4. Wang,X., Ghasedi Dizaji,K. and Huang,H. (2018) Conditional generative adversarial network for gene expression inference. *Bioinformatics*, **34**, i603–i611.
5. Zhu,Q., Wong,A.K., Krishnan,A., Aure,M.R., Tadych,A., Zhang,R., Corney,D.C., Greene,C.S., Bongo,L.A., Kristensen,V.N., *et al.* (2015) Targeted exploration and analysis of large cross-platform human transcriptomic compendia. *Nat. Methods*, **12**, 211–214.
6. Glorot,X. and Bengio,Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.pp. 249–256.
7. Kingma,D.P. and Ba,J. (2017) Adam: A Method for Stochastic Optimization. *ArXiv14126980 Cs*.
8. Reddi,S.J., Kale,S. and Kumar,S. (2019) On the Convergence of Adam and Beyond. *ArXiv190409237 Cs Math Stat*.
9. Zeiler,M.D. (2012) ADADELTA: An Adaptive Learning Rate Method. *ArXiv12125701 Cs*.
10. Chollet,F. (2015) Keras.
11. Abadi,M., Agarwal,A., Barham,P., Brevdo,E., Chen,Z., Citro,C., Corrado,G.S., Davis,A., Dean,J., Devin,M., *et al.* (2016) TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *ArXiv160304467 Cs*.