Deep-Union Completion

Siddharth Baskar[1*], **Karan Vikyath Veeranna Rupashree**[1*], **Daniel L Pimentel-Alarcón**[1,2]

[1]Wisconsin Institute for Discovery
[2]University of Wisconsin-Madison

Abstract

Large amounts of missing data are becoming increasingly ubiquitous in modern high-dimensional datasets. Unfortunately, classical completion methods like low-rank, high-rank, or deep matrix completion (LRMC/HRMC/DMC) are often unable to handle real data that does not fall under their respective models. Here we propose a novel completion strategy that generalizes all these models. The main idea is to find a Union of Subspaces (UoS) that can fit a non-linear embedding of the original data, and complete the data according to this latent UoS. This embedding is obtained through a novel pseudo-completion layer in a deep architecture, and the UoS structure is identified in closed-form through an intermediate clustering layer. Our design reduces the exponential memory requirements that are typically induced by uneven patterns of missing data. We give exact details of our architecture, model, loss functions, and training strategy. Our experiments on over 10 real datasets show that our method consistently outperforms the state-of-the-art accuracy by more than a staggering 40%.

GitHub Repository —
https://github.com/KaranVikyath/DUC

Introduction

Motivation: missing data. Missing data is a widespread challenge across various fields, including epidemiology, social sciences, finance, clinical research, computer vision and many more (Enders 2022; Baraldi and Enders 2010; Beaulieu-Jones et al. 2018; Gelman and Loken 2016; Little and Rubin 2019; Garcia-Laencina, Sancho-Gómez, and Figueiras-Vidal 2010). For example, missing data is observed in epidemiology due to participant attrition and incomplete responses during health assessments (Beaulieu-Jones et al. 2018), while in social science research, non-response bias in survey-based studies compromises representativeness (Gelman and Loken 2016). Clinical trials also face missing data due to participant dropout (Little and Rubin 2019). In finance and economic analysis, missing data occurs frequently, particularly in time-series data where gaps may arise due to reporting lags, data collection constraints, or economic events affecting data availability (Garcia-Laencina, Sancho-Gómez, and Figueiras-Vidal 2010). Additionally, computer vision encounters missing data when input image files are corrupted.

Prior work and limitations. Over the years, various methods have been developed to address missing data, but each has limitations (Woods et al. 2021). For instance, **(1)** single imputation methods (Zhang 2016) can reduce the variability of the data and introduce bias due to the assumption that one value can adequately replace the missing ones. **(2)** Deletion methods (Baraldi and Enders 2010; Enders 2022) can lead to significant data loss and potential bias if the missing data are not missing completely at random. **(3)** Model-based methods (Enders 2022; Ma and Chen 2018), (4) deep methods (Zheng and Charoenphakdee 2023; Khan, Wang, and Liu 2022), and **(5)** machine learning methods (Khosravi et al. 2020; Hong and Hao 2023) are computationally intensive and rely on the assumed underlying statistical model, which can lead to biased estimates and misleading conclusions. **(6)** Multiple imputation methods (Schafer 1999; White, Royston, and Wood 2011) require complex statistical expertise to implement and interpret results correctly. **(7)** Hybrid methods that include combinations of the aforementioned methods can be difficult to analyze and implement, potentially requiring careful tuning to balance the strengths and weaknesses of combined approaches, and some combinations can probably yield incorrect solutions (Elhamifar 2016). Moreover, none of these methods can handle the large amounts of missing data that are present in modern datasets and that are information-theoretically feasible (Pimentel-Alarcon and Nowak 2016). To summarize, existing methods that address missing data come with significant limitations that can impact the accuracy and reliability of their results. Also, none of these methods are fully capable of handling the large amounts of missing data typical in modern datasets, highlighting the ongoing challenges and the need for more robust approaches. More details about these and other completion methods are discussed at length in Appendix A.

This paper introduces a new, more general completion paradigm that we call *Deep Union Completion* (DUC). The main idea is to find a non-linear embedding of the data where its latent representation lies in a *union of subspaces* (UoS). The goal is to fill the missing values according to this latent UoS. Besides completing the missing values, our model also allows us to cluster the samples according to the latent UoS. To achieve this dual goal (completing and clustering), we

[*]These authors contributed equally.

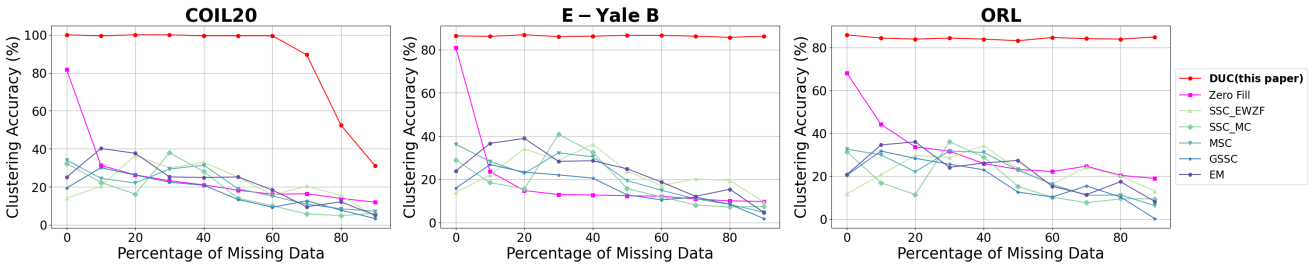

Figure 1: Clustering accuracy for COIL20, Yale B, and ORL dataset respectively. Our architecture outperforms the next best algorithm by a staggering 40% at any interval across all datasets. Notice that our DUC model slightly outperforms the zero-fill model (see subsection Novel pseudo-completion layer) even for 0% missing data. This is attributed to the increase in complexity of the model caused by the extra neural layers in the pseudo-completion layer.

propose a deep learning architecture that consists of three main components: (i) a novel completion layer, in charge of estimating the missing values in the data, (ii) an autoencoder that embeds the data into the latent space, and (iii) a middle clustering layer where we identify the union of subspaces in closed-form. As we will see, there is a myriad of specific applications that fall under our DUC model. Moreover, our methodology can be used as a routinely exploratory procedure (similar to PCA) on datasets of which nothing is known a prior.

Results overview. Our model is capable of achieving exceptional accuracy on real datasets, as evidenced by our experiments on the COIL20(Nene et al. 1996), Extended Yale B (Lee, Ho, and Kriegman 2005), ORL (Samaria and Harter 1994), Boston Housing (Harrison and Rubinfeld 1996), Period Changer (Gül and RAHIM 2022), Heart Disease (Janosi and Detrano 1988), Human Activity Recognition Using Smartphones (Reyes-Ortiz and Parra 2012), The Oxford-IIIT Pet (Parkhi et al. 2012), and 102 Category Flower (Nilsback and Zisserman 2008) datasets. The results of 3 datasets are summarized in Figure 1, where we compare to other state-of-the-art methods. This Figure shows that our architecture outperforms the next best algorithm in clustering accuracy by a staggering 40% at almost any interval of the percentage of missing data across the entire dataset. Figure 2 also demonstrates the extraordinary reconstruction capabilities of our model. This figure includes images observed from different datasets having 80% of its entries missing and the reconstruction images obtained from our model shows the uncanny similarity to the original images from the dataset.

Deep Union Completion Model

In this section we introduce our *deep union completion* (DUC) model, which can be seen as the generalization of *high-rank matrix completion* (HRMC) (Eriksson, Balzano, and Nowak 2012) that incorporates a latent feature representation. As we will see, this will enable us to handle more complex data patterns and larger amounts of missing values.

To define our model, let $\mathbf{X}$ be an $\mathrm{m} \times \mathrm{n}$ data matrix, and let $\mathbf{\Omega}$ be a subset of $\{1, \ldots, \mathrm{m}\} \times \{1, \ldots, \mathrm{n}\}$ indicating a subset of entries in $\mathbf{X}$. Define $\mathbf{X}^{\mathbf{\Omega}}$ as the $\mathrm{m} \times \mathrm{n}$ matrix that is equal to $\mathbf{X}$ in all the entries of $\mathbf{\Omega}$, and has missing

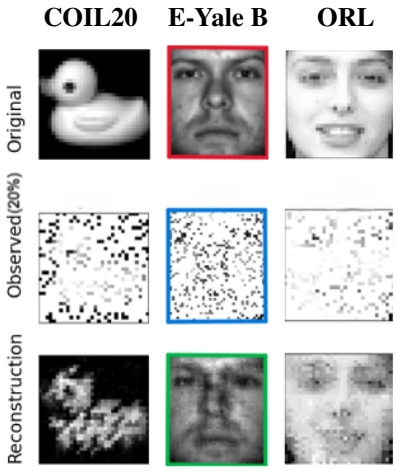

Figure 2: Image reconstruction for COIL20, E-Yale B and ORL datasets with 80% missing data. The reconstruction capabilities of DUC are remarkable. The images at the bottom are reconstructions obtained from the images in the middle, which has 80% of their entries missing. Compared to its originals, at the top.

values elsewhere. Let $\mathbf{x}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{m}}$ denote the i^{th} column of $\mathbf{X}$. Our DUM paradigm assumes that there exists a mapping $\mathcal{M} : \mathbb{R}^{\mathrm{m}} \to \mathbb{R}^{\ell}$, and a collection of *latent* subspaces $\mathcal{U}_1, \ldots, \mathcal{U}_{\mathrm{K}} \subset \mathbb{R}^{\ell}$ such that,

$$\bigcup_{\mathrm{i}=1}^{\mathrm{n}} \mathcal{M}(\mathbf{x}_{\mathrm{i}}) \subset \bigcup_{\mathrm{k}=1}^{\mathrm{K}} \mathcal{U}_{\mathrm{k}}. \tag{1}$$

In words, our DUC model assumes that there is an embedding $\mathcal{M}$ of the data $\mathbf{X}$ that is contained in the union of $\{\mathcal{U}_{\mathrm{k}}\}_{\mathrm{k}=1}^{\mathrm{K}}$. Given $\mathbf{X}^{\mathbf{\Omega}}$, our goals are to (i) estimate the missing values in $\mathbf{X}^{\mathbf{\Omega}}$, i.e., recover $\mathbf{X}$, (ii) learn the mapping $\mathcal{M}$, (iii) learn the latent subspaces $\{\mathcal{U}_{\mathrm{k}}\}_{\mathrm{k}=1}^{\mathrm{K}}$, and (iv) cluster the columns of $\mathbf{X}^{\mathbf{\Omega}}$ according to the latent UoS.

Data following the DUC paradigm. Notice that HRMC is the special case of DUC where the mapping $\mathcal{M} = \mathbf{I}$, the identity operator. If we additionally set $\mathrm{K} = 1$, then DUC becomes low-rank matrix completion (LRMC) (Candes and

Recht 2012; Recht 2011). Alternatively, if $\mathcal{M} = \mathbf{I}$ and no data is missing, DUC reduces to *subspace clustering* (SC) (Elhamifar and Vidal 2013). Finally, the full-data case with $\mathcal{M} = \mathbf{I}$ and $\mathrm{K} = 1$ is equivalent to *principal component analysis* (PCA) (Pearson 1901; Hotelling 1933; Jolliffe 2002). Therefore, as a generalization of these tasks, DUC can be used in all the applications where these methods are used: exploratory analysis (Jain and Shandliya 2013; Xia et al. 2017), dimensionality reduction (Jia et al. 2022; Reddy et al. 2020; Bharadiya 2023), visualization (Eckhardt et al. 2023), recommendations (Li and Murata 2012), computer vision (Peng et al. 2020), object tracking (Javed et al. 2022; Ma and Liu 2018), and the list is endless (Enders 2022; Qu et al. 2023; Fajariah et al. 2023). However, DUC enjoys the additional robustness capabilities brought by the flexibility of a general embedding $\mathcal{M}$ and $\mathrm{K} > 1$ latent subspaces. Thanks to this additional flexibility, DUC can handle modern complex datasets that are not captured by existing methods, and with much higher missing rates. Examples of such datasets arise in computer vision, more specifically in image reconstruction, health data analytics, exploratory analysis across a large number of domains including housing data, heart disease detection and many more. This is verified in our experiments below (see Figures 7-12) in Appendix B.

Beyond specific applications. Above we give a diverse list of applications that fall under our DUC paradigm. However, in general, data is rarely ever known to follow a specific model. That is in fact one of the main appeals of PCA: it can be used to determine *if* a given data follows a single-subspace model, and if so, to determine such subspace. In general, modern datasets rarely follow the single or multiple-subspace models of existing methods (PCA, SC, LRMC, HRMC), or at least not closely enough for existing methods to identify such models (i.e., there is often an excedent of noise, outliers, missing data, or other deviations). For all these datasets, it is possible that they follow the more general DUC paradigm. Similar to PCA, our methodology will enable us to use DUC as a routinely exploratory procedure to determine *if* a given dataset (of which nothing is known) follows the latent union model, and in that case, find such model. Thanks to its immense flexibility, we expect DUC to be applicable in a myriad of applications beyond the specific ones we mentioned above.

DUC is non trivial. Unsupervised learning is specially challenging in the presence of missing data because estimating the data's structure (e.g. embedding and latent union) typically requires filling in the missing entries, and filling in the missing entries appropriately requires knowledge of the structure (chicken and egg problem). Unfortunately, filling missing entries naively (e.g. with zeros or means) (Yang, Robinson, and Vidal 2015) introduces bias and disrupts the true underlying structure, leading to provably incorrect results (Elhamifar and Vidal 2013). Similarly, eliminating samples, or truncating or sketching features is infeasible when data is missing in large quantities (Pimentel-Alarcón, Balzano, and Nowak 2016). Finally, existing models like HRMC (Eriksson, Balzano, and Nowak 2012) do not account for an embedding, and are therefore inadequate to perform DUC. To summarize, DUC is a nontrivial task that requires special methodology.

Deep Union Completion Architecture

In this section we detail the *deep union completion* (DUC) architecture that we propose to (i) estimate the missing values in $\mathbf{X}^{\mathbf{\Omega}}$, (ii) learn the mapping $\mathcal{M}$, (iii) learn the latent subspaces $\{\mathcal{U}_{\mathrm{k}}\}_{\mathrm{k}=1}^{\mathrm{K}}$, and (iv) cluster the columns of $\mathbf{X}^{\mathbf{\Omega}}$ according to the latent UoS. To achieve these goals we use three main components: (a) a novel pseudo-completion layer that will be in charge of filling in the missing entries, (b) a standard autoencoder, similar to the one used in (Ji et al. 2017) to perform subspace clustering, and (c) a novel clustering layer that learns the latent union of subspaces using a closed-form solution, and clusters the columns in $\mathbf{X}^{\mathbf{\Omega}}$ accordingly. Next we describe each of these components in detail.

Novel Pseudo-Completion Layer

The first component of our architecture is a novel module comprised of 2 partially connected neural layers, which we call the pseudo-completion component. The main purpose of this component is to fill out the missing entries in the data. This is a challenging tasks due to the impossibility of computing pairwise distances between samples with missing entries (in order to cluster them), and filling the missing entries with naive values, like means or zeroes, creates a bias that deviates the model from the truth (more below). To avoid this, we will use a pseudo-completion component that will allow us to fill out the missing entries according to the latent model so that the distances required to cluster the data can be computed in the latent space. To present this layer, recall that the (incomplete-data) input to our model (and to the pseudo-completion layer, which is the first component of our model) is denoted by $\mathbf{X}^{\Omega}$. We will denote the (full-data) output of the pseudo-completion layer as $\mathbf{X}^{\mathrm{c}}$. This output will later be fed into the second component of the architecture. We explain the technical challenges posed by missing data on our architecture and the creation of pseudo-completion in detail in the following subsections.

Technical challenges of missing data. Typically, each neuron of an architecture receives as input, one feature of a single sample at a time. Therefore, when dealing with missing data, one natural approach is to mask the neurons corresponding to the missing entries in each sample at each training step (similar to a dropout strategy). However, in our case, each neuron of our architecture receives as input, a vector with one feature of *all* the samples. This is because the goal is to learn patterns between these columns that reveal the latent UoS. Therefore, under high missing data regimes that we are interested in, masking the neurons that receive any missing entry as input would result in all the neurons being masked at once, leaving no active connections between the input layer and the encoder.

Naive pseudo-completion layer. It is to be noted that our autoencoder needs the complete an $\mathrm{m} \times \mathrm{n}$ matrix to function. Hence, an additional component had to be introduced before the encoder to complete the data. One way to solve the masking problem is by flattening $\mathbf{X}^{\mathbf{\Omega}}$ into a $1 \times \mathrm{mn}$, such that each node in the input layer corresponds to one entry in $\mathbf{X}$. To facilitate the completion of the data and to maintain the shape of the matrix after the masking, the component would

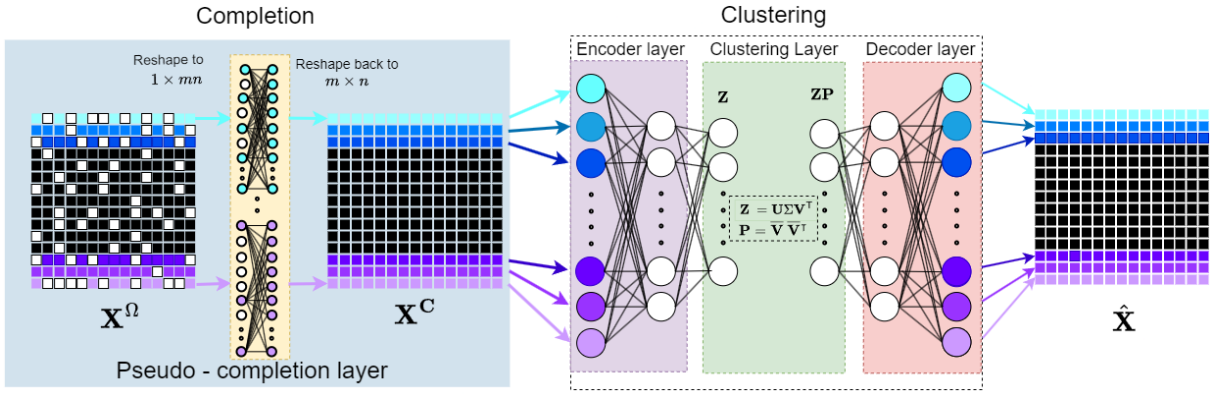

Figure 3: DUC Network.

have two $1 \times \mathrm{mn}$ dimensional layers which would be fully connected to impute the corresponding masked nodes from the input layer with weighted-sum of $\mathbf{X}^{\mathbf{\Omega}}$ in the second layer. This would allow us to mask the missing entries in a dropout style. The output of the second layer would be reshaped to the original input shape, which would be fed to the autoencoder.

Efficient pseudo-completion layer. Fully connected layers create large requirements of computing resources. To reduce this requirement, connections between the two layers were modified to be partially connected layers. Now each $\mathbf{x}_j^{\mathbf{\Omega}} \in \mathbb{R}^{\mathrm{n}}$ nodes in the flattened layer and the second layer were fully-connected. These partially connected layers together were termed as the pseudo-completion layer. We termed this obtained matrix as $\mathbf{X}^{\mathrm{c}}$ with $\mathbf{x}_j^{\mathrm{c}} \in \mathbb{R}^{\mathrm{n}}$ representing the rows of $\mathbf{X}^{\mathrm{c}}$. We implemented PReLu as the activation function for the neural layers in the pseudo-completion layer.

The importance of pseudo-completion is that in general, deep architectures will yield inaccurate results when missing data are filled naively with zeros or means. This can be seen across all datasets in our experiments (Figures 1, 4, 5, 6, and figures 7-12 in Appendix B, and figures 16-29 in Appendix E), where the pink line in the graph plots represents the zero-filled results, consistently diverging from the true model in red. This phenomenon highlights the need for our pseudo-completion layer, and explains its superior performance over naive completion and training strategies. More details in the Experiments section and Appendix E.

Standard Autoencoder

The second main component of our architecture is the autoencoder. It is designed for the encoder to map the functions from the pseudo-completion layer to the latent UoS, where the proposed closed-form solution will perform subspace clustering. The output from this layer is connected to the decoder component of the autoencoder which in turn generates the output matrix $\hat{\mathbf{X}}$.

Loss function. The loss function of the model was curated to accommodate the presence of missing data. Recall that we use $\mathbf{X}^{\mathbf{\Omega}}$ to represent the observed entries of $\mathbf{X}$ indicated by $\mathbf{\Omega}$. Then we compare $\mathbf{X}^{\mathbf{\Omega}}$ with $\hat{\mathbf{X}}^{\mathbf{\Omega}}$, which is $\hat{\mathbf{X}}$ observed on the corresponding entries of $\mathbf{X}^{\mathbf{\Omega}}$. To compare this (complete) output with the (incomplete) input of our network, we present our loss function:

$$L \;=\; \|\mathbf{X}^{\mathbf{\Omega}} - \hat{\mathbf{X}}^{\mathbf{\Omega}}\|_F^2$$

This loss aims to capture the error between the observed input and their corresponding entries in the output. Notice that this loss does not include any clustering term. That is because our network directly clusters the data in closed-form using the novel clustering layer described next.

Novel Closed-Form Clustering Layer

The purpose of this layer is to determine the clustering of the data in closed form. Its input is the embedding, $\mathbf{Z} = \mathcal{M}(\mathbf{X}^{\mathrm{C}})$, and its output is $\mathbf{ZP}$, where $\mathbf{P}$ is the projection operator onto the latent sample space. The key idea is that the clusters are encoded in the support of $\mathbf{P}$, and this choice of input-output reveals the embedding. The reason is that any embedding $\mathbf{Z}$ following a UoS model can be represented as a column permutation of

$$[\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_{\mathrm{K}}] = [\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{\mathrm{K}}]\mathbf{V}^{\mathsf{T}} + \mathbf{W}.$$

Here, $\mathbf{Z}_{\mathrm{k}}$ is the matrix containing the columns of $\mathbf{Z}$ corresponding to the k^{th} subspace, spanned by $\mathbf{U}_{\mathrm{k}}$, $\mathbf{W}$ is a noise matrix, and $\mathbf{V}$ has the following group-sparse structure:

$$\mathbf{V}^{\mathsf{T}} = \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_{\mathrm{K}} \end{bmatrix},$$

where the columns of $\mathbf{V}_{\mathrm{k}}$ contain the coefficients of $\mathbf{Z}_{\mathrm{k}}$ with respect to $\mathbf{U}_{\mathrm{k}}$. We can assume without loss of generality that each $\mathbf{V}_{\mathrm{k}}$ is orthonormal (otherwise, we can find a basis rotation $\mathbf{V}'_{\mathrm{k}} = \mathbf{\Gamma}\mathbf{V}_{\mathrm{k}}$, and apply the inverse rotation to obtain the alternative basis $\mathbf{U}'_{\mathrm{k}} = \mathbf{U}_{\mathrm{k}}\mathbf{\Gamma}^{-1}$ so that the new bases also yield the same product $\mathbf{Z}_{\mathrm{k}} = \mathbf{U}'_{\mathrm{k}}\mathbf{V}'_{\mathrm{k}}$. The key observation is that the projection operator onto $\mathrm{span}(\mathbf{V})$ encodes the clustering, because $\mathbf{P} = \mathbf{V}\mathbf{V}^{\mathsf{T}}$ has a block-diagonal form stemmed from the block-diagonal form of $\mathbf{V}$. Therefore, if the $(\mathrm{i}, \mathrm{j})^{\mathrm{th}}$ entry of $\mathbf{P}$ is nonzero, then the i^{th} and j^{th} columns of $\mathbf{Z}$ correspond to the same subspace cluster. Hence, the clusters can be determined as the distinct supports (nonzero

patterns) in the columns of $\mathbf{P}$. Naturally, the specific basis $\mathbf{V}$ is generally unidentifiable. However, projector operators are unique, so we can (a) estimate $\mathbf{P}$ as $\hat{\mathbf{P}} = \hat{\mathbf{V}}\hat{\mathbf{V}}^{\mathsf{T}}$, where $\hat{\mathbf{V}}$ contains the leading right singular vectors of $\mathbf{Z}$, and (b) apply any mechanism to determine the distinct supports of $\hat{\mathbf{P}}$, for example (i) a hierarchical approach that agglomerates samples (i, j) according to the magnitude of the entries of $\hat{\mathbf{P}}$, (ii) a direct application of a clustering algorithm on $\hat{\mathbf{P}}$, such as Lloyd's algorithm (Lloyd 1982), K-means++ (Arthur and Vassilvitskii 2007) or spectral clustering (Ng, Jordan, and Weiss 2001), or (iii) an application on $\hat{\mathbf{P}}$ of any method that minimizes distortion. In our implementation we simply use spectral clustering on $\hat{\mathbf{P}}$.

Notice that our loss function does not explicitly enforce clustering, as would, for example, an alternative loss like

$$L' \;=\; \|\mathbf{X}^{\mathbf{\Omega}} - \hat{\mathbf{X}}^{\mathbf{\Omega}}\|_F^2 \;+\; \lambda\|\mathbf{Z} - \mathbf{Z}\mathbf{Q}\|_*,$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\mathbf{Q}$ is another parameter to be learnt, and $\lambda > 0$ is a regularization parameter. This type of explicit clustering (i) requires designing tailored (non-general) clustering penalties that may only be suited for specific cases, (ii) its performance has high variability, depending on the compatibility of penalty and the data at hand, (iii) it requires further optimization to identify the additional parameters (e.g., $\mathbf{Q}$), and (iv) its performance heavily depends on the careful tuning of the regularization parameters, which can be difficult and computationally onerous. In contrast, in our architecture, clustering is implicitly attained without the need for additional regularization terms or parameters, by requiring that the decoder recovers $\mathbf{X}$ from $\mathbf{ZP}$, and not from the encoder's embedding $\mathbf{Z}$ directly. This forces the learnt embedding to satisfy the condition $\mathbf{Z} = \mathbf{ZP}$, which ensures that $\mathbf{Z}$ lies near a UoS. This is because an embedding follows the group-sparse form above if and only if $\mathbf{ZP} = (\mathbf{U}\mathbf{V}^{\mathsf{T}})(\mathbf{V}\mathbf{V}^{\mathsf{T}}) = \mathbf{U}(\mathbf{V}^{\mathsf{T}}\mathbf{V})\mathbf{V}^{\mathsf{T}} = \mathbf{U}\mathbf{V}^{\mathsf{T}} = \mathbf{Z}$.

Implementation and Training

Pretraining. The clustering layer in DUC requires receiving the entire dataset at once (rather than one sample at a time). This is required because the closed-form clustering layer must compute similarities between all samples to learn the patterns in the observed data that will reveal the UoS. Since we are dealing with missing data, it is not possible to pretrain the autoencoder, and hence, we do not pretrain our model. It also does not require initialization of the model with a preprocessed dataset for performing subspace clustering unlike other SC models. It is also to be noted that for training we do not have a set epoch value for termination but rather the termination happens when the learning rate reaches a value of the original learning rate/10.

Output, finding K and dimensions. For the closed-form solution we have to determine the rank of the matrix $\mathbf{Z}$ so that we can reduce $\mathbf{V}$ to the rank of $\mathbf{Z}$. This rank is determined by the product of two factors: K, representing the number of subspaces, and d, that indicates the dimensionality of each subspace. Subsequently, this reduced matrix $\mathbf{V}$ is used to determine the projection matrix $\mathbf{P}$ as explained above. We use the product $\mathbf{ZP}$ to give the resultant matrix that is then fed into the decoder section of the autoencoder. In the context of classification datasets, the value of K is assumed to be the number of distinct classes. Another approach to discern the rank of $\mathbf{Z}$ involves employing Singular Value Decomposition (SVD) on the dataset. The resultant singular values are plotted, which helps in understanding the dataset's structure and identifying significant features or patterns. We set a threshold of 95%-99% and for the number of singular values and that fall below this threshold is considered to be the rank of $\mathbf{Z}$. This method was used to calculate the rank for regression dataset where K and d are non existent.

Experiments

The following section provides an overview of the baseline models used for comparison, the experimental setup, and a discussion of the results and improvements observed in our paper. It is essential to note that while our DUC network has shown remarkable performance on real datasets, the results obtained for synthetic data are only comparable to the baseline models. This can be attributed to several factors considered for conducting the synthetic data experiments.

Initially, the synthetic datasets were formulated to conform to conditions optimal for performance of the baseline models. These conditions had perfectly separated clusters, nearly orthogonal subspaces, small dimensions, uniform distribution among subspaces, and isotropic data distribution within each subspace. Furthermore, specific parameters were meticulously selected to enhance the baseline models' performance, establishing an environment tailored to their strengths. Despite the advantageous conditions favoring the baseline models, our network still delivered competitive results. Notably, our model was not explicitly engineered or fine-tuned for such idealized scenarios. Instead, it showcased its resilience and versatility by demonstrating commendable performance even within this highly controlled environment. This underscores its potential for broader applicability across diverse datasets and scenarios. In summary, while the baseline models may excel under controlled conditions specifically tailored to their strengths, our network's ability to achieve competitive results in such idealized settings highlights its versatility and robustness in real-world applications.

Comparative baselines. For the comparative analysis, synthetic and real data were provided to 10 different methods to undergo data completion and data clustering. These methods can be categorized into two types: the first type includes methods exclusively for data completion, while the second type encompasses methods capable of both completion and clustering. Among the completion-only algorithms are the following: (1) SimpleFill: This technique involves filling missing values in a dataset with the most recent non-missing value. (2) K-Nearest Neighbors: This method utilizes the similarity between data points to estimate missing values (Troyanskaya et al. 2001). (3) Iterative Imputer: An advanced imputation technique that iteratively predicts missing values by modeling each feature with missing data as a function of other features (Van Buuren and Groothuis-Oudshoorn 2011). (4) SoftImpute: Missing points are recovered by minimizing the rank of the completed matrix while incorporating a

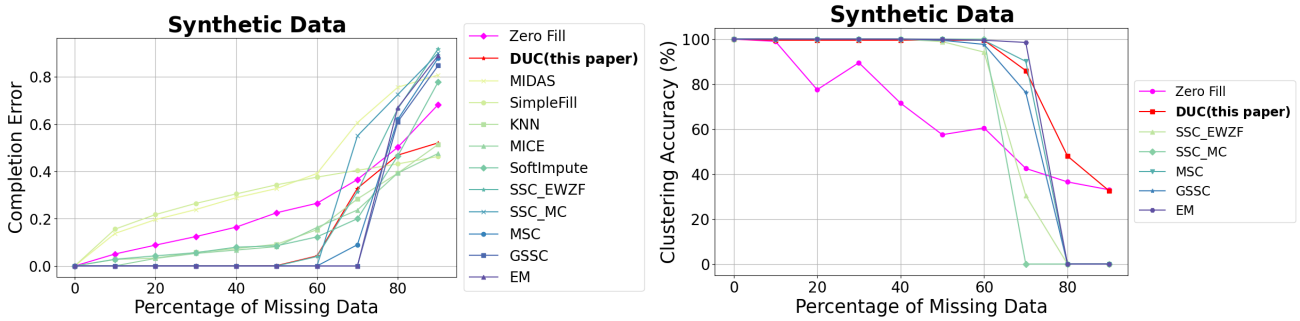

Figure 4: Completion Error and Clustering Accuracy for Synthetic Data.

penalty term (Mazumder, Hastie, and Tibshirani 2010). (5) MIDAS: This approach employs denoising autoencoders to corrupt and reconstruct data (Lall and Robinson 2022). The remaining 5 models fall under the second category, capable of both reconstruction and subspace clustering. These models include SSC-MC, SSC-EWFZ, GSSC, EM, and MSC, as discussed in the related work section. Additionally as mentioned in our DUC architecture section we also generated results for the zero fill model, where we impute the missing entries with 0 instead of using our pseudo-completion layer.

Setup for synthetic data. We constructed a synthetic dataset to approximate a union of subspaces as described below. Initially, we randomly generated K d-dimensional subspaces within $\mathbb{R}^{\mathrm{n}}$. For each subspace, we created a matrix $\mathbf{U}_{\mathrm{k}} \in \mathbb{R}^{\mathrm{d}\times\mathrm{n}}$ with entries drawn from a standard Gaussian distribution, which we subsequently orthogonalized. Data for each subspace, represented by $\mathbf{X}_{\mathrm{k}} \in \mathbb{R}^{\mathrm{m_k}\times\mathrm{n}}$, was then generated as:

$$\mathbf{X}_{\mathrm{k}} = \mathbf{V}_{\mathrm{k}}\mathbf{U}_{\mathrm{k}} + \mathbf{E}_{\mathrm{k}},$$

where $\mathbf{V}_{\mathrm{k}} \in \mathbb{R}^{\mathrm{m_k}\times\mathrm{d}}$ has entries following a normal distribution $\mathcal{N}(0, 1/\mathrm{d})$, and $\mathbf{E}_{\mathrm{k}} \in \mathbb{R}^{\mathrm{m_k}\times\mathrm{n}}$ consists of noise with entries distributed according to $\mathcal{N}(0, \sigma^2/\mathrm{d})$. We assembled these $\mathbf{X}_{\mathrm{k}}$ matrices vertically to compile the complete data matrix $\mathbf{X}$, resulting in dimensions $\mathrm{n} \times \mathrm{m}$. It's important to note that each row in our dataset represents a feature. To simulate the observation matrix $\mathbf{\Omega}$, we randomly sampled exactly $\ell > 0$ observed entries across the dataset.

Parameters. For the purposes of our experiments, the synthetic dataset dimensions were set to 200×50 ($\mathrm{m} \times \mathrm{n}$), with parameters set as $\mathrm{m_k} = 50$, $\mathrm{K} = 4$, $\mathrm{d} = 3$, and $\mathrm{n} = 50$, including a minor noise factor $\sigma = 0.01$. The dataset was designed with $\mathrm{m_k}$ denoting the number of samples per subspace, K the number of subspaces, d the dimensionality of each subspace, and n the ambient space dimension.

Accuracy metrics. We also introduced missing entries across $\mathbf{X}$ to mimic real-world scenarios where data might be incompletely observed due to various factors. This dataset was then used to evaluate our model, specifically examining how the performance, depicted in Figure 4, varied with different levels of missing data. For this experiment, we consider two types of accuracy for our final result $\hat{\mathbf{X}}$. To measure completion error we take the normalized Frobenius norm of the difference between $\hat{\mathbf{X}}$ and $\mathbf{X}$, i.e., $\|\hat{\mathbf{X}} - \mathbf{X}\|_F / \|\mathbf{X}\|$. We additionally measure clustering accuracy in which we quantify the proportion of correctly assigned data points to their respective clusters compared to the ground truth. The choice for opting this accuracy is due to clustering being effective for segmenting images into meaningful regions or for compressing images while preserving essential details. Clustering accuracy is also necessary for preserving quality and integrity of reconstructed features. Further experiments are performed on varying noise factor to determine the robustness of our model and these experiments can be found in Appendix E.

The importance of pseudo-completion. Figure 6 shows that without a pseudo-completion strategy, both completion error and clustering accuracy deteriorates across most setups (most nontrivial cases of of K and d) by an average of 50% for clustering and 30% for completion accuracy. We performed the same experiment on real datasets with similar results, detailed in Appendix E.

Real data. The true performance of any model can be determined from the results obtained when real data is fed into the model. For this purpose, we chose a wide range of real datasets to perform our experiment. Similar to what we performed with synthetic data, we randomly replaced data entries present with differing amounts of missing entries in these datasets. For this experiment, we ran our model on all datasets with 20%, 50%, and 80% missing entries in each dataset. It is observed from Figure 1 and other comparative graphs in Figure 5 that, the clustering accuracy and completion error of the baseline models can generate outstanding results in synthetic data but are unable to replicate the desired results on real data. In contrast, our model performs better with real data than what was initially obtained from the synthetic data. Here onwards, we discuss all the datasets that were used for the experiments. The results for the rest of these extensive experiments can be found in Appendix B.

COIL20 (Nene et al. 1996). The first real dataset used in this project is COIL20. This dataset consists of 1440 grayscale 128×128 images of $\mathrm{K} = 20$ objects with 72 poses each. These images were reshaped to 32×32 pixels for computation feasibility.

Extended Yale B (Lee, Ho, and Kriegman 2005). We then used the Extended Yale B which contains 2414 images of 38 human subjects with 64 images per person, where all the images are manually aligned, cropped, and then re-sized to 192×168 images. From this, we used only 20 human subjects

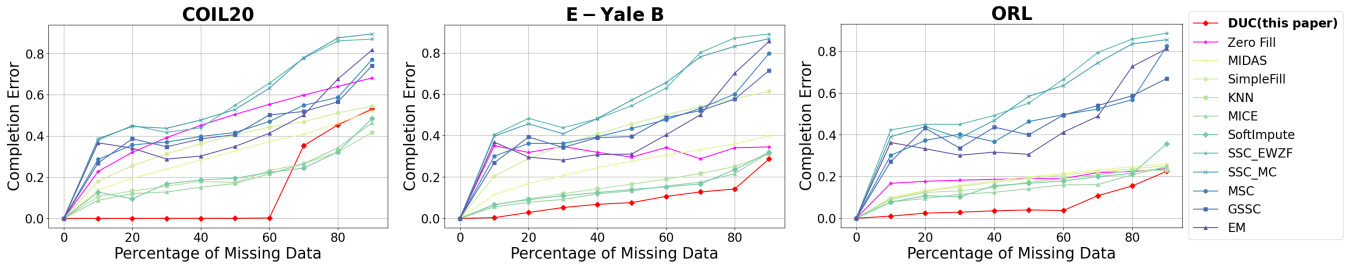

Figure 5: Completion error for COIL20 (Nene et al. 1996), Extended Yale B (Lee, Ho, and Kriegman 2005), and ORL Dataset (Samaria and Harter 1994).

with a total of 1280 images which were reshaped to 48×42 pixels and this dataset had $\mathrm{K} = 20$.

ORL Dataset (Samaria and Harter 1994). The ORL Database of Faces contains 400 images from 40 distinct subjects. The size of each image is 92×112 pixels, which was reshaped to 32×32 pixels, with 256 gray levels per pixel.

Boston Housing (Harrison and Rubinfeld 1996). This is a non-image, data driven regression dataset. We chose this dataset to inspect our model's performance on regression datasets and how it performs without K and d. It contains data regarding various housing and neighborhood details. This dataset contains 507 samples with 14 features.

Period Changer (Gül and RAHIM 2022). This dataset contains information about non-toxic molecules designed for functional domain of a core clock protein, CRY1. The goal is to classify these molecules between two classes: changer and no changer. This dataset contains 90 instances and 1177 features with $\mathrm{K} = 2$.

Heart disease (Janosi and Detrano 1988). The goal of this dataset is to classify if a patient has heart disease or not. This is a classification dataset that contains 303 samples with 14 features and $\mathrm{K} = 5$. It comprises of Categorical, Integer, Real values.

Human Activity Recognition Using Smartphones (Reyes-Ortiz and Parra 2012). This is a classification dataset and we chose this dataset due to its large size. This dataset classifies human activities into walking, sitting, standing, and laying. It contains 10299 samples with 561 features and $\mathrm{K} = 6$

The Oxford-IIIT Pet Dataset (Parkhi et al. 2012). This is an image based dataset of pet animals with images having images have a large variations in scale, pose and lighting. All images have an associated ground truth annotation of breed, head ROI, and pixel level trimap segmentation. This dataset contains 200 sample images and 37 classes with $\mathrm{K} = 37$.

102 Category Flower Dataset (Nilsback and Zisserman 2008). This was final dataset that was used for the experiments and contains images that are to be classified into different flower species. This dataset has 102 flower categories and each class consists of 40 to 258 images with $\mathrm{K} = 102$.

Reproducibility. Table 1 in Appendix C gives exact details of each of these datasets, along with the exact parameters used for our implementation and experiments.

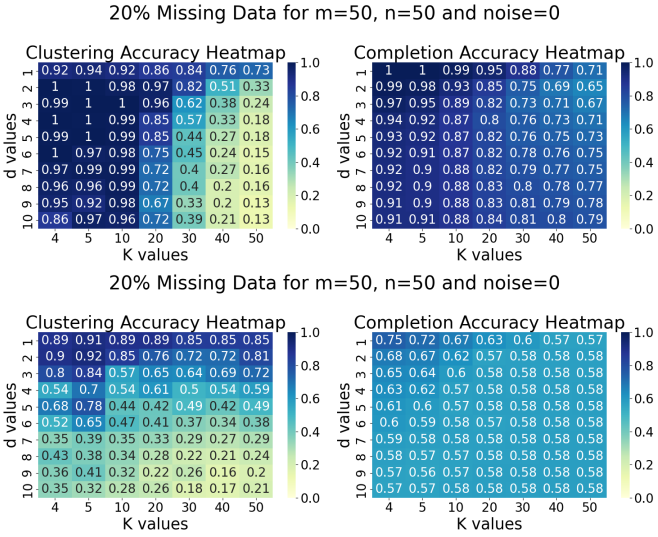

Figure 6: Heat maps depicting clustering and completion accuracy for ORL dataset with and without pseudo-completion layer (Noise = 0, missing data = 20%). The two heat maps in the top row are generated by the model with pseudo-completion layer and the bottom ones are generated without pseudo-completion layer.

Results conclusions. Our results in Figure 5 show that DUC performed better than all other models and even generated 0 error for completion with up to 60% missing data (see COIL20 dataset). For the Extended Yale - B and ORL dataset, the DUC model uniformly outperformed all other models. Figure 2 shows the remarkable image reconstruction for a randomly sampled image from the three aforementioned datasets at 80% of missing data. Figures 7-12 in Appendix B depict the results for the remaining datasets, in which the DUC outperforms all other models for a wide range of datasets which include multiple non-image classification dataset, a couple of colored image dataset, and a regression dataset. Moreover, Figures 13-15 in Appendix C demonstrate the image reconstruction ability of our DUC model which also includes colored images. Also, Figures 16-29 in Appendix E show the resiliency of our model to noise while also justifying the importance of the pseudo-completion layer.

Acknowledgments

This work was partially supported by NSF's CAREER Award #2239479.

References

Arthur, D.; and Vassilvitskii, S. 2007. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035.

Baraldi, A. N.; and Enders, C. K. 2010. An introduction to modern missing data analyses. *Journal of school psychology*, 48(1): 5–37.

Beaulieu-Jones, B. K.; Lavage, D. R.; Snyder, J. W.; Moore, J. H.; Pendergrass, S. A.; and Bauer, C. R. 2018. Characterizing and managing missing structured data in electronic health records: data analysis. *JMIR medical informatics*, 6(1): e8960.

Bharadiya, J. P. 2023. A tutorial on principal component analysis for dimensionality reduction in machine learning. *International Journal of Innovative Science and Research Technology*, 8(5): 2028–2032.

Candes, E.; and Recht, B. 2012. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6): 111–119.

Eckhardt, C. M.; Madjarova, S. J.; Williams, R. J.; Ollivier, M.; Karlsson, J.; Pareek, A.; and Nwachukwu, B. U. 2023. Unsupervised machine learning methods and emerging applications in healthcare. *Knee Surgery, Sports Traumatology, Arthroscopy*, 31(2): 376–381.

Elhamifar, E. 2016. High-rank matrix completion and clustering under self-expressive models. In *Advances in Neural Information Processing Systems*, 73–81.

Elhamifar, E.; and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11): 2765–2781.

Enders, C. K. 2022. *Applied missing data analysis*. Guilford Publications.

Eriksson, B.; Balzano, L.; and Nowak, R. 2012. High-rank matrix completion. In *Artificial Intelligence and Statistics*, 373–381. PMLR.

Fajariah, F.; Saragih, H.; Dharmawan, D.; Judijanto, L.; and Munizu, M. 2023. Application of Principal Component Analysis and Maximum Likelihood Estimation Method to Identify the Determinant Factors Intention to Use of Paylater in E-Commerce. *Jurnal Informasi dan Teknologi*, 118–123.

Garcia-Laencina, P. J.; Sancho-Gómez, J.-L.; and Figueiras-Vidal, A. R. 2010. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19: 263–282.

Gelman, A.; and Loken, E. 2016. The statistical crisis in science. *The best writing on mathematics (Pitici M, ed)*, 305–318.

Gül, ; and RAHIM, F. 2022. Period Changer. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5B31D.

Harrison, D.; and Rubinfeld, D. L. 1996. Housing Data Set. http://lib.stat.cmu.edu/datasets/boston. Accessed: 2024-05-20.

Hong, X.; and Hao, S. 2023. Imputation of Missing Values in Training Data using Variational Autoencoder. In *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)*, 49–54.

Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6): 417.

Jain, P. M.; and Shandliya, V. 2013. A survey paper on comparative study between principal component analysis (PCA) and exploratory factor analysis (EFA). *International Journal of Managment, IT and Engineering*, 3(6): 415–424.

Janosi, S. W. P. M., Andras; and Detrano, R. 1988. Heart Disease. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52P4X.

Javed, S.; Mahmood, A.; Ullah, I.; Bouwmans, T.; Khonji, M.; Dias, J. M. M.; and Werghi, N. 2022. A novel algorithm based on a common subspace fusion for visual object tracking. *IEEE Access*, 10: 24690–24703.

Ji, P.; Zhang, T.; Li, H.; Salzmann, M.; and Reid, I. 2017. Deep subspace clustering networks. *Advances in neural information processing systems*, 30.

Jia, W.; Sun, M.; Lian, J.; and Hou, S. 2022. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3): 2663–2693.

Jolliffe, I. T. 2002. *Principal component analysis for special types of data*. Springer.

Khan, H.; Wang, X.; and Liu, H. 2022. Handling missing data through deep convolutional neural network. *Information Sciences*, 595: 278–293.

Khosravi, P.; Vergari, A.; Choi, Y.; Liang, Y.; and Broeck, G. V. d. 2020. Handling missing data in decision trees: A probabilistic approach. *arXiv preprint arXiv:2006.16341*.

Lall, R.; and Robinson, T. 2022. The MIDAS touch: accurate and scalable missing-data imputation with deep learning. *Political Analysis*, 30(2): 179–196.

Lee, K.-C.; Ho, J.; and Kriegman, D. J. 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, 27(5): 684–698.

Li, X.; and Murata, T. 2012. Using multidimensional clustering based collaborative filtering approach improving recommendation diversity. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, 169–174. IEEE.

Little, R. J.; and Rubin, D. B. 2019. *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2): 129–137.

Ma, L.; and Liu, Z. 2018. Hybrid Sparse Subspace Clustering for Visual Tracking. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 1737–1742. IEEE.

Ma, Z.; and Chen, G. 2018. Bayesian methods for dealing with missing data problems. *Journal of the Korean Statistical Society*, 47: 297–313.

Mazumder, R.; Hastie, T.; and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11: 2287–2322.

Nene, S. A.; Nayar, S. K.; Murase, H.; et al. 1996. Columbia object image library (coil-20).

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated Flower Classification over a Large Number of Classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V. 2012. Cats and Dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pearson, K. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572.

Peng, X.; Feng, J.; Zhou, J. T.; Lei, Y.; and Yan, S. 2020. Deep subspace clustering. *IEEE transactions on neural networks and learning systems*, 31(12): 5509–5521.

Pimentel-Alarcón, D.; Balzano, L.; and Nowak, R. 2016. Necessary and sufficient conditions for sketched subspace clustering. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 1335–1343. IEEE.

Pimentel-Alarcon, D.; and Nowak, R. 2016. The information-theoretic requirements of subspace clustering with missing data. In *International Conference on Machine Learning*, 802–810. PMLR.

Qu, W.; Xiu, X.; Chen, H.; and Kong, L. 2023. A survey on high-dimensional subspace clustering. *Mathematics*, 11(2): 436.

Recht, B. 2011. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12).

Reddy, G. T.; Reddy, M. P. K.; Lakshmanna, K.; Kaluri, R.; Rajput, D. S.; Srivastava, G.; and Baker, T. 2020. Analysis of dimensionality reduction techniques on big data. *Ieee Access*, 8: 54776–54788.

Reyes-Ortiz, A. D. G. A. O. L., Jorge; and Parra, X. 2012. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K.

Samaria, F. S.; and Harter, A. C. 1994. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision*, 138–142. IEEE.

Schafer, J. L. 1999. Multiple imputation: a primer. *Statistical methods in medical research*, 8(1): 3–15.

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(6): 520–525.

Van Buuren, S.; and Groothuis-Oudshoorn, K. 2011. mice: Multivariate imputation by chained equations in R. *Journal of statistical software*, 45: 1–67.

White, I. R.; Royston, P.; and Wood, A. M. 2011. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine*, 30(4): 377–399.

Woods, A. D.; Davis-Kean, P.; Halvorson, M. A.; King, K.; Logan, J.; Xu, M.; Bainter, S.; Brown, D.; Clay, J. M.; Cruz, R. A.; et al. 2021. Missing data and multiple imputation decision tree.

Xia, J.; Ye, F.; Chen, W.; Wang, Y.; Chen, W.; Ma, Y.; and Tung, A. K. 2017. LDSScanner: Exploratory analysis of low-dimensional structures in high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, 24(1): 236–245.

Yang, C.; Robinson, D.; and Vidal, R. 2015. Sparse subspace clustering with missing entries. In *International Conference on Machine Learning*, 2463–2472. PMLR.

Zhang, Z. 2016. Missing data imputation: focusing on single imputation. *Annals of translational medicine*, 4(1).

Zheng, S.; and Charoenphakdee, N. 2023. Diffusion models for missing value imputation in tabular data. arXiv:2210.17128.