Latent Union Completion

Karan Vikyath Veeranna Rupashree
Wisconsin Institute for Discovery
Madison, WI, USA
veerannarupa@wisc.edu

Siddharth Baskar
Wisconsin Institute for Discovery
Madison, WI, USA
sbaskar2@wisc.edu

Daniel Pimentel-Alarcón
Dept. of Biostatistics and Medical Informatics
University of Wisconsin-Madison
Madison, WI, USA
pimentelalar@wisc.edu

2025 IEEE International Symposium on Information Theory (ISIT) | 979-8-3315-4399-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISIT63088.2025.11195409

***Abstract*—Large amounts of missing data are becoming increasingly ubiquitous in modern high-dimensional datasets. High-rank matrix completion (HRMC) uses the powerful union of subspace (UoS) model to handle these vast amounts of missing data. However, existing HRMC methods often fail when dealing with real data that does not follow the UoS model exactly. Here we propose a new approach: instead of finding a UoS that fits the observed data directly, we will find a UoS in a latent space that can fit a non-linear embedding of the original data. Embeddings of this kind are typically attained with deep architectures. However, the abundance of missing data impedes the training process, as the coordinates of the observed samples rarely overlap. We overcome this difficulty with a novel pseudo-completion layer (in charge of estimating the missing values) followed by an autoencoder (in charge of finding the embedding) coupled with a self-expressive layer (that clusters data according to a UoS in the latent space). 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 several real datasets show that our method consistently outperforms the state-of-the-art accuracy by more than a staggering 40%.**

***Index Terms*—Subspace clustering, autoencoder, matrix completion, pseudo-completion layer**

I. Introduction

Motivation: Missing Data. Missing data is a widespread challenge in various fields, including epidemiology, social sciences, finance, clinical research, computer vision, and many more [1]–[6]. For example, missing data are observed in epidemiology due to participant attrition and incomplete responses during health assessments [3], while in social science research, nonresponse bias in survey-based studies compromises representativeness [4]. Clinical trials also face missing data due to participant dropout [5]. 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 [6]. In addition, 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 [7]. For instance, **(1)** single imputation methods [8] 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 [1], [9] can lead to significant data loss and potential bias if the missing data are not missing completely at random. **(3)** Model based methods [2], [10] and **(4)** machine learning methods [11] are computationally intensive and rely on the assumed underlying statistical model, which can lead to biased estimates and misleading conclusions. **(5)** Multiple imputation methods [12], [13] require complex statistical expertise to implement and interpret the results correctly. **(6)** 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 [14]. 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 [15].

Instead of being evenly distributed throughout the feature space, high-dimensional data is sometimes seen to display a low-dimensional structure. Through the use of this structure, observable entries can be completed by interpolating missing values through the inference of the underlying structure. Such completion tasks typically make use of linear subspaces, as exemplified by Low-rank Matrix Completion (LRMC) techniques [16].

High-rank Matrix Completion (HRMC) approaches are more effective at approximating modern datasets, which frequently exhibit complexities that exceed the capacity of a single subspace model [17]. HRMC accommodates numerous subspaces by extending LRMC with a Union of Subspaces (UoS) framework. This effectively adapts Subspace Clustering (SC) techniques [18] to handle incomplete data. The goals of HRMC are as follows: (a) determine the ideal UoS that most accurately depicts the incomplete data matrix $\mathbf{X}^{\mathbf{\Omega}}$; (b) cluster the columns of $\mathbf{X}^{\mathbf{\Omega}}$ in accordance with the determined UoS; and (c) fill in the missing values within $\mathbf{X}^{\mathbf{\Omega}}$. Cluster knowledge would make it easier to apply LRMC to each cluster for data completion and subspace learning; on the other hand, missing value detection would help SC cluster the data and find subspaces. These two goals are intertwined. The main challenge lies in completing these objectives at the same time.

In recent years, there has been a proliferation of High-rank Matrix Completion (HRMC) algorithms due to the widespread adoption of the Union of Subspaces (UoS) model. The algorithms demonstrate a wide range of approaches and levels of performance. Among the notable techniques are nearby methods, which use distances between partially seen sites to construct clusters [17]. Studies also describe naïve methods that

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

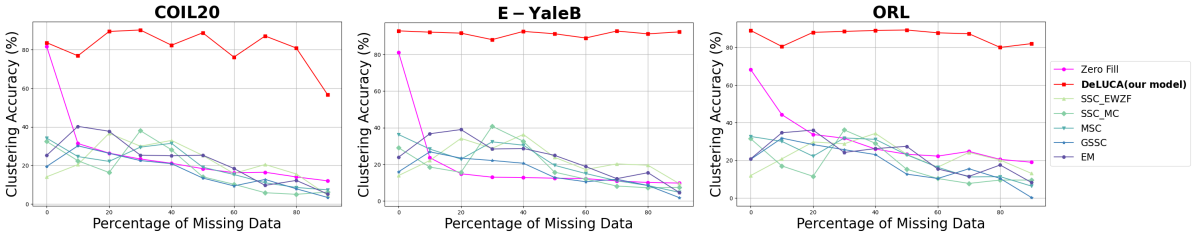

Fig. 1: Clustering accuracy for COIL20, Yale B, and ORL dataset respectively. Our architecture outperforms the next best algorithm by a staggering 40% at any interval across all datasets.

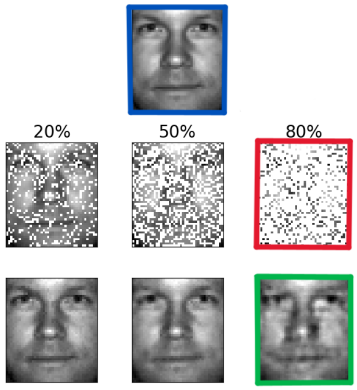

Fig. 2: Reconstruction Images for Yale B dataset. The reconstruction capabilities of our DeLUCA model are extraordinary. For example, the image highlighted in green is a reconstruction obtained from the image highlighted in red, which has 80% of their entries missing. Compare to its original, highlighted in blue.

replace missing items with zeros or means before clustering using a Subspace Clustering (SC) method [19]. As described by [20], additional techniques have also been developed, such as GROUSE [21], which are used in addition to techniques that combine aspects of ridge and lasso regression. Usage of unions as second-order algebraic structures in techniques known as "liftings" is another novel strategy [22]. Furthermore, the incorporation of quantum computing into HRMC represents a breakthrough, where quantum algorithms are utilized for data imputation to boost performance in comparison to conventional techniques [23]. When it comes to neural networks, variational autoencoders [24] and Long Short-Term Memory (LSTM) networks [25] are commonly paired with one another to handle data imputation. In addition, generative models like Denoising Diffusion Probabilistic Models (DDPM) [26] and Generative Adversarial Networks (GANs) [27] are being used more and more in matrix completion tasks.

Unfortunately, there are drawbacks to each of these techniques. For instance, naïve approaches face challenges when working with relatively large datasets since the mere act of filling in missing data disrupts the fundamental Union of Subspaces (UoS) structure [19]. On the other hand, neighborhood approaches are not feasible in many situations since they need

# of Subspaces	1	K	K+embedding
Full	PCA	SC	DSC-net
Incomplete	LRMC	HRMC	**LUC (this paper)**

Fig. 3: LUC and related problems.

a super-polynomial number of samples or a significant number of observations to provide adequate overlaps [17]. Increasing the dimensionality of an already high-dimensional space is the goal of lifting methods [22]. Problems like quantum noise and computational constraints occur in quantum computing, a field within quantum information theory [23]. Furthermore, while many approaches produce excellent results on synthetic data, real-world data performance presents difficulties.

In this paper, we present *Latent Union Completion* (LUC), a novel and broader completion model. The basic goal is to identify a non-linear structure that encapsulates the observed data and may be represented as a UoS in a latent space. The objective is to use this latent UoS (LUoS) to fill in the missing data. Our proposal involves a deep learning architecture with three main components: (i) an autoencoder that embeds the data into the latent space; (ii) an auto-completion layer that estimates the missing values in the input $\mathbf{X}^{\mathbf{\Omega}}$; and (iii) a self-expressive layer that clusters the embedded data based on a UoS model. In this manner, we can concurrently do both goals of clustering and completing with our architecture, which we refer to as *Deep Latent Union Completion Architecture* (DeLUCA). Above all, our experiments on the COIL20 [28], Extended Yale B, [29], and ORL [30] datasets demonstrate that it can achieve extraordinarily high accuracy on real datasets. Figure 1 provides a summary of the findings and allows for comparison with other cutting-edge techniques. This Figure demonstrates how, at every given interval of the proportion of missing data over the whole dataset, our architecture performs 40% better in clustering accuracy than the next best approach. Furthermore highlighting our model's remarkable reconstruction ability is Figure 2. To illustrate an astonishing resemblance to the genuine (original) image, highlighted in blue, the face from the Yale B dataset, indicated in red, was finished from the partial image above, noted in green. The outcomes of the remaining samples are equally striking (see Figures throughout).

Architecture Novelty. Another model that has shown success in the related problem of Subspace Clustering (SC) serves as the basis for our design, DeLUCA. Remember that when data is completely observed, SC can be thought of as the particular case of HRMC, where the objective is just to cluster the data based on a UoS. More specifically, we modify the DSC-net architecture [31] by introducing a unique pseudo-completion layer made up of two partially connected layers as a first component. The main innovation is that the missing elements will be imputed from the normalized entries of the observed data after the data has gone through the pseudo-completion layer. This makes it possible for us to enter data that is incomplete into a clustering network that would not operate otherwise. By including this pseudo-completion layer, our design can now smoothly handle clustering and finishing at the same time.

II. LUoS Model and DeLUCA Network

This section contains a detailed presentation of our LUoS model, along with an explanation of the difficulties a deep learning architecture faces in the event of missing data and how we overcome them to create our DeLUCA network. Hereafter we will use $\mathbf{X}$ to denote a full-data matrix of size $m \times n$, and $\mathbf{X}^{\mathbf{\Omega}}$ to denote the incomplete version of $\mathbf{X}$ that is only observed in the entries of $\mathbf{\Omega} \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$.

HRMC assumes that the rows of $\mathbf{X}$ lie near the union of K subspaces denoted by $\mathcal{U}_1, \ldots, \mathcal{U}_{\mathrm{K}}$. Given $\mathbf{X}^{\mathbf{\Omega}}$, the goals of HRMC is (a) to infer the underlying subspaces $\mathcal{U}_1, \ldots, \mathcal{U}_{\mathrm{K}}$, (b) to cluster the columns of $\mathbf{X}^{\mathbf{\Omega}}$ according to their closest subspace, and (c) to complete the missing values in $\mathbf{X}^{\mathbf{\Omega}}$. LUC. Unfortunately, many datasets do not lie near a UoS. However, any data is more likely to lie near a non-linear structure that can be represented as a UoS in a latent space. That is because UoSs are the special case of this latent model where the embedding is simply the identity map. Hence, we will assume that there exists an embedding $\mathbf{Z} \in \mathbb{R}^{m \times r}$ of $\mathbf{X}$ where the rows of $\mathbf{Z}$ lie near the union of K subspaces denoted by $\mathcal{V}_1, \ldots, \mathcal{V}_{\mathrm{K}}$. We make no assumptions about the number of subspaces or their dimensions. Given $\mathbf{X}^{\mathbf{\Omega}}$, the goals of LUC are (a) to find the embedding $\mathbf{Z}$ and infer the latent subspaces $\mathcal{V}_1, \ldots, \mathcal{V}_{\mathrm{K}}$, (b) to cluster the columns of $\mathbf{Z}$ (and by correspondence, the columns of $\mathbf{X}^{\mathbf{\Omega}}$) according to their closest latent subspace, and (c) to complete the missing values in $\mathbf{X}^{\mathbf{\Omega}}$ according to the inverse embedding and the latent UoS. To achieve these goals, we will use a deep network architecture that is detailed below.

A. Architecture

Three major parts make up our DeLUCA network (Figure 4): (i) a self-expressive layer, (ii) an autoencoder, and (iii) a pseudo-completion layer. The pseudo-completion layer enables the handling of missing data differently than the traditional imputation methods like zero-fill or mean imputation. This is by no means a simple task, as each sample contains uneven patterns of observed entries that disturb the data to the point where it can no longer be fed directly into the autoencoder, as is usually the case with full-data methods such as DSC-net [31] (for more information, see Section II-C).

Pseudo-completion Layer. This consists of two flattened, partially linked layers. In order to create a completed data matrix $\mathbf{X}^{\mathrm{c}}$ with placeholder imputations, these layers normalize all the observed elements in $\mathbf{X}^{\mathbf{\Omega}}$. These normalized values then replace the missing entries. In this manner, the matrix $\mathbf{X}^{\mathrm{c}}$ can be smoothly fed into our architecture's autoencoder component. See Section II-C for more information.

Autoencoder. The second key element of our architecture is the autoencoder. The autoencoder performs the task of mapping the imputed data from the pseudo-completion layer into an embedding where the self-expressive layer can locate the latent UoS structure which will then cluster the given data. This autoencoder, as usual, consists of two parts. The encoder is the first component whose goals is to capture important features and patterns in the input data and compress the data into a reduced dimensional space until it generates a compact representation in the latent space. The second part being the decoder, which returns the data to its original space, is the second part of the autoencoder. Completing the data in accordance with the latent UoS is its goal.

Self-expressive Layer. The self-expressive layer is our model's final essential element. The autoencoder contains this layer in the center, in between the encoder and the decoder. Its objective is to use Sparse Subspace Clustering (SSC) to determine the UoS [32]. Sparsity in data representations is used in SSC, a technique for exposing UoS structures in high dimensional datasets. The method is to express each sample as a linear combination of the remaining data, so creating a sparse representation of each sample. Subspaces are then revealed by clustering the data according to the coefficients of these combinations. These three layers achieve their respective tasks through a coupled loss function, detailed below.

B. Loss Function

The design of the loss function required careful consideration in order to accommodate the presence of missing data, and the disparity between the incomplete input and the complete output. Recall that we use $\mathbf{X}^{\mathbf{\Omega}}$ to represent the incomplete version of $\mathbf{X}$ observed on the entries indicated in $\mathbf{\Omega}$. Here $\mathbf{X}^{\mathbf{\Omega}}$ is the input of our network. We use $\mathbf{Z}$ to denote the latent representation of $\mathbf{X}$, which corresponds to the output of the encoder and the input to the self-expressive layer, whose weights we represent with the coefficient matrix $\mathbf{\Theta} \in \mathbb{R}^{m \times m}$. Notice that besides $\mathbf{\Theta}$, the output of our network depends on all the other parameters of the network (weights of the pseudo-completion and autoencoder), which we denote as $\mathbf{\Phi}$. To emphasize this dependency, we use $\hat{\mathbf{X}}_{\mathbf{\Phi}}$ to denote the output of our network. Finally, to compare the (complete) output with the (incomplete) input of our network, we define $\hat{\mathbf{X}}_{\mathbf{\Phi}}^{\mathbf{\Omega}}$ as the incomplete version of $\hat{\mathbf{X}}_{\mathbf{\Phi}}$ observed on the entries indicated in $\mathbf{\Omega}$. With this, we are ready to present our loss function:

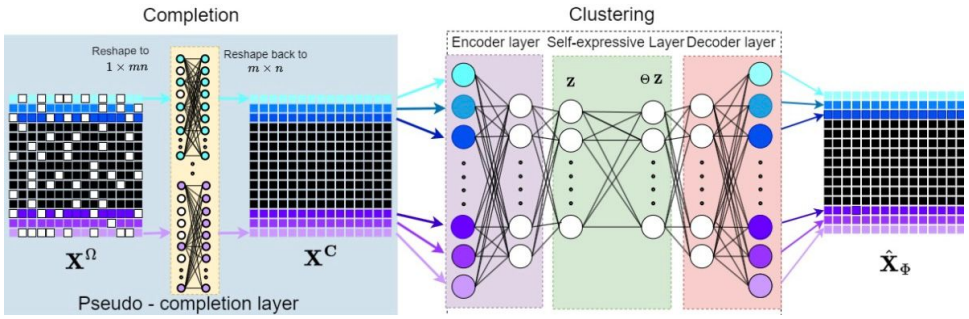

Fig. 4: DeLUCA Network.

$$L(\mathbf{\Theta}, \mathbf{\Phi}) \;=\; \frac{1}{2}\|\mathbf{X}^{\mathbf{\Omega}} - \hat{\mathbf{X}}^{\mathbf{\Omega}}_{\mathbf{\Phi}}\|_F^2 + \frac{\lambda_1}{2}\|\mathbf{Z} - \mathbf{\Theta}\mathbf{Z}\|_F^2 \;+\; \lambda_2\|\mathbf{\Theta}\|_2$$

Coefficients and Parameter Tuning. In the first term of the loss function, we minimize the error between the observed input and their corresponding entries in the output. The second term aims to express each row in $\mathbf{Z}$ as a linear combination of the remaining rows of $\mathbf{Z}$. These coefficients are given by $\mathbf{\Theta}$. The intuition is that $\mathbf{\Theta}$ will reveal the UoS structure because larger coefficients will indicate belonging to the same subspace. The last term aims to regularize $\mathbf{\Theta}$, so that it produces a stable result with minimum norm (as there could be arbitrarily many solutions). We show that SSC generally uses an ℓ_1 norm for this last term, in order to favor sparse solutions. Our choice to use the ℓ_2 norm instead was driven by recent results showing improved performance [33]. In this work it was also concluded that the typical constraint $diag(\mathbf{\Theta}) = \mathbf{0}$ is not a strict requirement when the ℓ_2 regularizer is used. Finally, λ_1 and λ_2 are regularization parameters. These regularization parameters were determined by an iterative refinement approach where parameters were tuned based on their impact on model performance, leading to the identification of optimal values. Our training goal is to find the parameters $\mathbf{\Theta}$ and $\mathbf{\Phi}$ that minimize this loss. We describe our strategy to attain this goal below.

C. Training Strategy

The self-expressive layer requires receiving the entire dataset at once (rather than one sample at a time). This is required because the 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 pre-processed 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. Furthermore, for the loss function, we determine the values of $\mathbf{\Theta}$ and $\mathbf{\Phi}$ by iterative tuning where multiple configurations were explored, and the final values were selected based on their performance in minimizing the loss.

The additional challenge is that, contrary to what is frequently accomplished by other approaches [34], the missing entries cannot be naively supplied with zeros or mean values since this type of imputation introduces bias and distorts the true underlying low-dimensional structure [14]. In a similar vein, we are unable to truncate or sketch (keep only a few characteristics) due to the possibility of bias, information loss, and decreased generalization ability caused by missing data in every column [35].

It was decided to mask the missing entries so that their absence would not impact the weights assigned to them by the network. But it needed to be done cautiously as the rationale behind it is that every neuron encodes a single complete feature. However, every feature contains a significant number of missing entries in the high missing data regimes that we are interested in. Therefore, masking any neuron with missing values would mask all the neurons, meaning that there would be no active connections between the encoder and the input layer.

Creation of Pseudo Completion Layer. We solved the masking problem by flattening $\mathbf{X}^{\mathbf{\Omega}}$ into a $1 \times \ell$ dimensional vector, where $\ell = m \times n$. Now each node contains one entry and all of the missing entries were masked. An additional layer of $1 \times \ell$ dimension was introduced in the model before encoder. For the earlier version of the architecture these layers were fully connected. These layers also served the purpose of preserving the original shape after the masking procedure. The entries in the second layer were then reshaped from $1 \times \ell$ into its original $m \times n$. We termed this obtained matrix as $\mathbf{X}^{\mathrm{c}}$ with $\mathbf{x}^{\mathrm{c}}_j \in \mathbb{R}^m$ representing the feature vectors of $\mathbf{X}^{\mathrm{c}}$. For activation, RELU activation function was implemented in the pseudo-completion layer.

Refining Pseudo Completion Layer. But fully connected layers created large requirements of computing resources. To reduce this requirement, connections between initial layers were modified to be partially connected layers. Now each $\mathbf{x}^{\mathbf{\Omega}}_j$ nodes at m intervals in the flattened layer and the additional layer were interconnected. These partially connected layers together were termed as the pseudo-completion layer.

At the output of the pseudo-completion layer, the data $\mathbf{X}^{\mathrm{c}}$ is now compatible with the autoencoder. This activates the autoencoder which then performs its function of mapping $\mathbf{X}^{\mathrm{c}}$

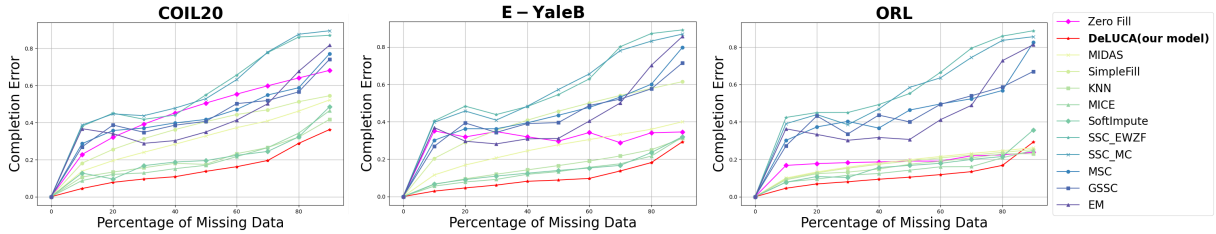

Fig. 5: Completion Errors for COIL20, Extended Yale B, and ORL Dataset.

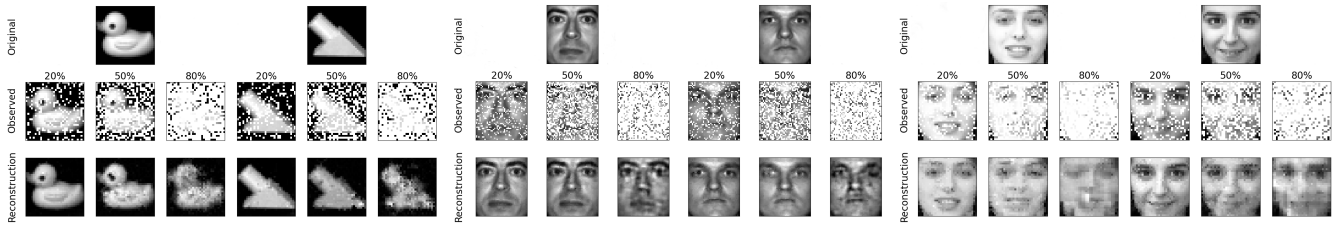

Fig. 6: Reconstruction results for different missing data percentages for COIL20, E-YaleB and ORL Datasets.

into an embedding where the self-expressive layer can find the LUoS structure that clusters the given data. Following the clustering process, the decoder reverses the embedding with the final layer containing $\hat{\mathbf{x}}_{\mathbf{\Phi}_j} \in \mathbb{R}^m$ feature vectors. This is then resolved into an output matrix; $\hat{\mathbf{X}}_{\mathbf{\Phi}}$ that comprises of predicted values.

III. Experiments

Comparative Baselines. The 10 methods that we used for comparative analysis 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. The following are examples of completion-only algorithms: (1) SimpleFill: This method uses the most recent non-missing value to fill in any missing values in a dataset. (2) K-Nearest Neighbors: This technique estimates missing values using the similarity between data points [36]. (3) Iterative Imputer: Using a sophisticated imputation method, each feature with missing data is modeled as a function of other characteristics, allowing the iterative prediction of missing values [37]. (4) SoftImpute: A penalty term is incorporated while the rank of the finished matrix is minimized to recover missing points [38]. (5) MIDAS: To corrupt and recover data, this method uses denoising autoencoders [39]. The remaining five models, which can perform both subspace clustering and reconstruction, belong to the second group. These models, which are covered in the related work section, include SSC-MC, SSC-EWFZ, GSSC, EM, and MSC. Additionally, as mentioned in Section 2, we also generated results for the zero-fill model, where we impute the missing entries with 0 instead of using our pseudo-completion layer.

COIL20. 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. It was observed from Figure 5 that DeLUCA performed better than all other models in terms of completion error when plotted against the percentage of missing data. It can also be noted from Figure 1 that, the model performs significantly better than all other methods in terms of clustering accuracy and that it outperforms the next best method by more than 40%.

As seen from Figure 6, in a set of 2 random images sampled from the COIL20 dataset the image reconstruction is performed by DeLUCA for three levels of missing entries starting from 20% to 50% and then finally at 80%.

Extended Yale B. We then used 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 with a total of 1280 images which were reshaped to 48×42 pixels and this dataset had $\mathrm{K} = 20$. Similar to COIL20, it is again observed from Figure 5 that the completion error of DeLUCA outperforms uniformly over all the other methods for any amount of missing data in the dataset. Also worth noting that for this dataset, even at 80% missing entries, the model performs at a clustering accuracy of more than 80%.

ORL Dataset. And finally, we used the ORL Database of Faces that contains 400 images from 40 distinct subjects. The size of each image is 92×112 pixels which were then reshaped to 32×32 pixels, with 256 grey levels per pixel.

Missing Data and Performance Metrics. To emulate missing data we artificially removed entries from each dataset uniformly at random with replacement. To measure completion error we computed the normalized Frobenius norm of the difference between $\hat{\mathbf{X}}_{\mathbf{\Phi}}$ and $\mathbf{X}$, i.e., $\|\hat{\mathbf{X}}_{\mathbf{\Phi}} - \mathbf{X}\|_F / \|\mathbf{X}\|$. We measured clustering accuracy as the proportion of correctly assigned data points to their respective clusters after using the Hungarian algorithm to find the best matching labeling.

REFERENCES

[1] A. N. Baraldi and C. K. Enders, "An introduction to modern missing data analyses," *Journal of school psychology*, vol. 48, no. 1, pp. 5–37, 2010.

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

[3] B. K. Beaulieu-Jones, D. R. Lavage, J. W. Snyder, J. H. Moore, S. A. Pendergrass, and C. R. Bauer, "Characterizing and managing missing structured data in electronic health records: data analysis," *JMIR medical informatics*, vol. 6, no. 1, p. e8960, 2018.

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

[5] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2019, vol. 793.

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

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

[8] Z. Zhang, "Missing data imputation: focusing on single imputation," *Annals of translational medicine*, vol. 4, no. 1, 2016.

[9] D. A. Newman, "Missing data: Five practical guidelines," *Organizational Research Methods*, vol. 17, no. 4, pp. 372–411, 2014.

[10] Z. Ma and G. Chen, "Bayesian methods for dealing with missing data problems," *Journal of the Korean Statistical Society*, vol. 47, pp. 297–313, 2018.

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

[12] J. L. Schafer, "Multiple imputation: a primer," *Statistical methods in medical research*, vol. 8, no. 1, pp. 3–15, 1999.

[13] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: issues and guidance for practice," *Statistics in medicine*, vol. 30, no. 4, pp. 377–399, 2011.

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

[15] D. Pimentel-Alarcon and R. Nowak, "The information-theoretic requirements of subspace clustering with missing data," in *International Conference on Machine Learning*. PMLR, 2016, pp. 802–810.

[16] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Communications of the ACM*, vol. 55, no. 6, pp. 111–119, 2012.

[17] B. Eriksson, L. Balzano, and R. Nowak, "High-rank matrix completion," in *Artificial Intelligence and Statistics*. PMLR, 2012, pp. 373–381.

[18] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *Acm sigkdd explorations newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[19] S. Van Buuren, *Flexible imputation of missing data*. CRC press, 2018.

[20] D. Pimentel-Alarcón, L. Balzano, R. Marcia, R. Nowak, and R. Willett, "Group-sparse subspace clustering with missing data," in *2016 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2016, pp. 1–5.

[21] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *2010 48th Annual allerton conference on communication, control, and computing (Allerton)*. IEEE, 2010, pp. 704–711.

[22] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[23] S. Kazdaghli, I. Kerenidis, J. Kieckbusch, and P. Teare, "Improved clinical data imputation via classical and quantum determinantal point processes," *arXiv preprint arXiv:2303.17893*, 2023.

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," 2022.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[28] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-20)," 1996.

[29] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[30] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE workshop on applications of computer vision*. IEEE, 1994, pp. 138–142.

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

[32] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[33] P. Ji, M. Salzmann, and H. Li, "Efficient dense subspace clustering," in *IEEE Winter conference on applications of computer vision*. IEEE, 2014, pp. 461–468.

[34] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2463–2472.

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

[36] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[37] S. Van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, vol. 45, pp. 1–67, 2011.

[38] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *The Journal of Machine Learning Research*, vol. 11, pp. 2287–2322, 2010.

[39] R. Lall and T. Robinson, "The midas touch: accurate and scalable missing-data imputation with deep learning," *Political Analysis*, vol. 30, no. 2, pp. 179–196, 2022.