Missing Modalities Imputation via Cascaded Residual Autoencoder

Luan Tran, Xiaoming Liu, Jiayu Zhou
Department of Computer Science and Engineering
Michigan State University, East Lansing MI 48824
{tranluan, liuxm, jiayuz}@msu.edu

Rong Jin
Alibaba Group Holding Limited
Hangzhou, Zhejiang, China
jinrong.jr@alibaba-inc.com

Abstract

Affordable sensors lead to an increasing interest in acquiring and modeling data with multiple modalities. Learning from multiple modalities has shown to significantly improve performance in object recognition. However, in practice it is common that the sensing equipment experiences unforeseeable malfunction or configuration issues, leading to corrupted data with missing modalities. Most existing multi-modal learning algorithms could not handle missing modalities, and would discard either all modalities with missing values or all corrupted data. To leverage the valuable information in the corrupted data, we propose to impute the missing data by leveraging the relatedness among different modalities. Specifically, we propose a novel Cascaded Residual Autoencoder (CRA) to impute missing modalities. By stacking residual autoencoders, CRA grows iteratively to model the residual between the current prediction and original data. Extensive experiments demonstrate the superior performance of CRA on both the data imputation and the object recognition task on imputed data.

1. Introduction

The soaring advances in sensor technologies have greatly reduced the cost of producing sensors for different purposes, which allows researchers to collect data with multiple modalities. The *multi-modal data* describes one sample or datum point from different perspectives, and these perspectives provide complementary information about the datum point [13], but are also closely related because they describe the same sample. Researchers have conducted extensive research on how to combine the useful information from different modalities to better achieve application-specific goals [6, 10, 12, 17–19, 21, 25]. Especially, combining information from multiple modalities is shown to be very effective in various computer vision tasks, such as object detection from aerial videos [19], where we may obtain multiple sensing information, including RGB, LiDAR, multispectral imaging, hyperspectral imaging, GPS, etc.

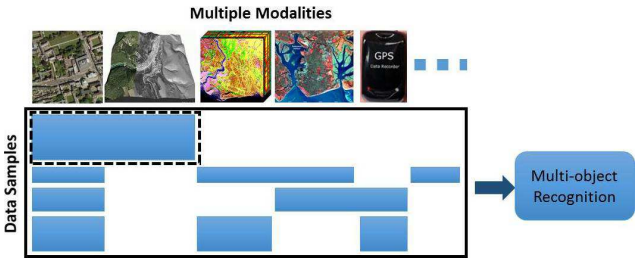


Figure 1: Learning with missing modalities. Given a large collection of sensor data from multiple modalities, data imputation may take advantage of *all* available data to learn multi-object classifiers despite the missing modalities (white area). In contrast, the existing approaches may have to remove some modalities and/or training samples so that all *remaining* training samples are observed in all *remaining* modalities (dashed box).

Most prior work on information fusion of multi-modal data assumes that all modalities are available for every training data point [10, 21]. This assumption can greatly limit applications of multi-modal analysis because in practice data collection process may likely generate data points with missing modalities. E.g., unforeseeable sensor malfunction may fail to retrieve sensing information. Moreover, configuration issues in sensing equipment may lead to incompatible data. Thus, when jointly analyzing data from different agencies, sample coverage may not be entirely the same and some corresponding modalities may be incompatible. We call samples with missing modalities as *corrupted samples*. Such corrupted samples can impose significant challenges to apply multi-modal analysis: we may need to choose from (i) remove corrupted samples from training or (ii) remove the modalities with corrupted samples. We illustrate the problem of missing modalities in Fig. 1. Both options, unfortunately, would eliminate potentially useful information we have collected. Moreover, the multi-modal analysis would fail when a significant portion of training data is corrupted.

The missing modality problem is a special type of missing data problem. Traditionally, by assuming that the missing values of a data matrix are random (i.e., missing-at-

random, MAR), there are many readily usable imputation methods, whose underlying principle is to take advantage of the latent relatedness among matrix elements, and infer the missing values from the observed elements. One well-studied imputation method is matrix completion [4,5]. However, in the multi-modal data, when concatenating features of different modalities into the data matrix, the missing values are no longer randomly distributed, but rather appear in *blocks*: the missing values would simultaneously appear in one modality (Fig. 1). Due to the violation of MAR, traditional methods can no longer establish guarantees in recovering corrupted samples with missing modalities. Also, many matrix completion methods involve iterative Singular Value Decomposition (SVD) of the entire data matrix, rendering it computationally prohibitive for large data.

In this paper, we propose a novel cascaded residual autoencoder (CRA) for imputation with missing modalities, which is composed of a set of stacked residual autoencoders (RAs) that iteratively model the residuals. The conventional autoencoder has been used to impute missing data in many domains such as traffic data [11] and sensor networks [35], where the input layer is the data sample with missing entries and the output layer is the complete data sample. In contrast, for the RA at each layer of the CRA, its desired output is the difference between the input (i.e., incomplete) data sample and the complete data sample. Starting from the first layer, each RA is learned sequentially such that its output pushes the overall imputed data to be closer to the full data. This forward learning paradigm results in a cascaded autoencoder that takes the corrupted data and estimate a function well approximating the complete data. Further, we also develop a joint optimization scheme to simultaneously estimate the parameters of all RAs in CRA such that the overall loss can be further minimized. This optimization is performed similar to the back propagation in Convolutional Neural Network (CNN). Experimental results on benchmark datasets demonstrate the superiority of CRA on both the data imputation and subsequent object recognition based on the imputed data.

In summary, this paper makes these contributions: 1) identify the general problem of missing modality in multi-modal data; 2) propose a data imputation method using cascaded residual autoencoder; 3) demonstrate state-of-the-art performance on imputing data with missing modalities.

2. Related Works

Since we propose a novel data imputation method for the missing modality problem, we review prior works on imputation, e.g., matrix completion and autoencoders.

Matrix completion is the task of filling in the missing entries of a partially observed matrix. Matrix completion often assumes the missing entries are related to the observed ones, which is equivalent to a low-rank structure in the completed matrix. This low-rank assumption could be applicable to our missing modality problem due to correlations among different modalities. Some existing approaches are based on the nuclear norm minimization, such as SVT [3] and Soft-Impture [26]. OptSpace [20], on the other hand, formulates from a matrix factorization view based on traditional SVD.

There are two limitations of applying matrix completion to our problem: (i) they usually compute the SVD of the data matrix - computationally expensive with a large data matrix, which is often the case with multiple modalities; (ii) many matrix completion settings assume missing at random (MAR) (e.g., [5]). However, missing continuous blocks in multi-modal data obviously break this assumption. Hence, matrix completion might not be optimal for our problem, as demonstrated by the results in Fig. 8.

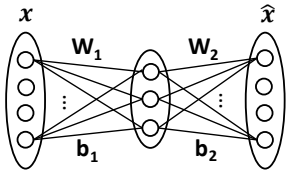


Figure 2: Autoencoder.

Autoencoder (AE) is originally an unsupervised learning method, with the objective of learning a latent (hidden) data representation, from which we may reconstruct original data through a neural network. It encodes an input vector to a hidden representation via a non-linear mapping, and then decodes it back via another mapping (Fig. 2). The autoencoder is trained so that the output ($\hat{\mathbf{x}}$) is as similar to the input ($\mathbf{x}$) as possible. One early AE-based imputation method combines AE with genetic algorithms (GA) [1,28]. Given a trained AE, the complete vector combining the guessed and observed values is fed into the AE as input. GA will select the vector that minimizes the difference between input and output of the AE. Vincent et al. [33] propose a noise-robust variant of AE for feature extraction, called Denoising Autoencoder (DA). DA reconstructs the noise-free signal given its corrupted counterpart as the input. DA is more applicable to imputation than combining AE and GA.

Stacked Autoencoders (sDA) Despite the success of autoencoders in various datasets [1,9,11,28], it is challenging for a single-layer autoencoder to model complex relationship among data in different modalities. Several autoencoders can be stacked to form a deep hierarchy [33]. An AE can be placed inside a previous AE, and receive its input from the previous AE's latent representation (hidden layer) and learn to reconstruct this representation. It is shown in [16] that without pretraining, the deep autoencoder always reconstructs the average of the training data. However, in our application it can be difficult to learn a deep autoencoder with multiple latent layers, given the limited training samples due to the large amount of missing data in multiple modalities, which is validated by Tab. 3.

Autoencoder in Multi-modal Data There are also auten-

coder methods tailored for multi-modal data. Ngiam et al. [30] use deep autoencoder to learn interaction between high-level features of speech and video signals. Wang et al. [34] enforce the correlation in the feature representation of multi-modal data. Since these works focus more on representation learning, they do not necessarily lead to good imputation performance, as will be shown in Tab. 3.

Deep Residual Network Our proposed CRA is inspired by recent achievement in deep CNN for object recognition. He et al. [15] propose a residual learning framework termed ResNet for object recognition. They reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. While CRA and ResNet share similarity in network design, they serve different purposes. CRA is for data imputation and ResNet is for object recognition. Further, CRA adopts layer-by-layer learning followed by joint optimization, while the parameters of ResNet are learned end-to-end. Finally, CRA is able to dynamically determine its depth while ResNet needs a pre-defined depth.

3. Cascaded Residual Autoencoder

The Autoencoder is used in prior work [11, 27, 28, 35] to impute missing data, in the case of missing at random (MAR). For MAR, it is rare to have a continuous large block of missing entries. E.g., even when the data sparsity is 10% (i.e., 90% of data are missing), the probability of having 20 continuous missing entries is merely 12%. For data with a missing modality, the missing entries typically occur in a much *larger* continuous block. This key difference implies that in missing modality the correlation between the available and missing entries is more complicated than that of MAR. This clearly poses a challenge for a single autoencoder to accurately recover the missing modality.

An intuitive solution for this problem is to add extra hidden layers inside an autoencoder to create a deep neural network, e.g., sDA. However, our experiments reveal that in our applications the performance of sDA is not better than a denoising autoencoder. We hypothesize that the deep autoencoder is more difficult to train, especially with limited training samples (see Tab. 3) - a typical scenario due to missing modalities. To this end, we propose a novel cascaded residual autoencoder (CRA) framework for data imputation. In each cascade layer, a residual autoencoder is trained to approximate the residual between the input data, which is the currently recovered data, and the desired uncorrupted data. With multiple cascade layers, CRA progressively refines its estimation toward the uncorrupted data.

3.1. Autoencoder

An autoencoder aims to learn the latent representation of data, which can reconstruct original data. It maps a d-dim input vector $\mathbf{x}$ to a d'-dim hidden representation $\mathbf{z} = f(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$, where $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times d'}$, $\mathbf{b}^{(1)} \in \mathbb{R}^{d'}$ and f is a non-linear function of ReLU. The hidden representation is then mapped to the output $\hat{\mathbf{x}} = f(\mathbf{W}^{(2)}\mathbf{z} + \mathbf{b}^{(\mathbf{2})})$, where $\mathbf{W}^{(2)} \in \mathbb{R}^{d' \times d}$ and $\mathbf{b}^{(\mathbf{2})}, \hat{\mathbf{x}} \in \mathbb{R}^{d}$. By generalizing to a deep autoencoder with L layers, and denoting the value at the l-th layer as $\mathbf{a}^{(l)}$, we have $\mathbf{a}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{a}^{(l)} + \mathbf{b}^{(l)})$.

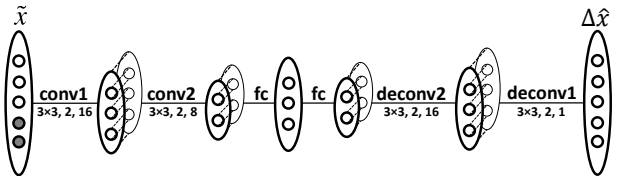


Figure 3: Convolutional residual autoencoder. The format of the convolution parameter is: filter size, stride, filter number.

To this end, an autoencoder maps the input to output via:

$$\theta_{\mathbf{W},\mathbf{b}}(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^d : \theta_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \mathbf{a}^{(L)}$$
$$\theta_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = f\left(\mathbf{W}^{(L)} f(...f(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})...) + \mathbf{b}^{(L)}\right) \quad (1)$$

The parameters $\{\mathbf{W}, \mathbf{b}\}$ are obtained by minimizing the L_2 loss between $\mathbf{x}$ and $\hat{\mathbf{x}}$ over all training data:

$$\mathbb{L} = \tfrac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \tfrac{1}{2}\|\mathbf{x} - \theta_{\mathbf{W},\mathbf{b}}(\mathbf{x})\|_2^2. \quad (2)$$

With the same architecture as the autoencoder, a denoising autoencoder (DA) reconstructs a complete output from a partially observed input.

3.2. Residual Autoencoder

The basic building block of our proposed CRA is a variant of autoencoder called Residual Autoencoder (RA). An RA has the same structure as the conventional autoencoder or DA, including the input layer, latent layer(s) and output layer. Both RA and DA take the corrupted data as the input layer. In the output layer, DA produces the completed data, while RA generates the difference between the input data and the completed data. This seemingly minor difference has a significant impact: it enables us to stack a set of RA in a cascaded architecture, to refine the estimation after each autoencoder. Mathematically, for a single RA, we transform incomplete input data $\tilde{\mathbf{x}}$ into a hidden layer, and then to the output layer, both through nonlinear mappings as in autoencoder. The desired output is defined as $\Delta\mathbf{x} = \mathbf{x} - \tilde{\mathbf{x}}$. RA aims to make the estimated output, $\Delta\hat{\mathbf{x}} = \theta_{\mathbf{W},\mathbf{b}}(\tilde{\mathbf{x}})$, to be as close to the desired output as possible, in terms of least squares. This leads to the following loss function:

$$\mathbb{L} = \tfrac{1}{2}\|\Delta\mathbf{x} - \Delta\hat{\mathbf{x}}\|_2^2 = \tfrac{1}{2}\|(\mathbf{x} - \tilde{\mathbf{x}}) - \theta_{\mathbf{W},\mathbf{b}}(\tilde{\mathbf{x}})\|_2^2. \quad (3)$$

3.3. Convolutional Residual Autoencoder

When input data are 2D images, we extend RA to convolutional residual autoencoder, as shown in Fig. 3. This

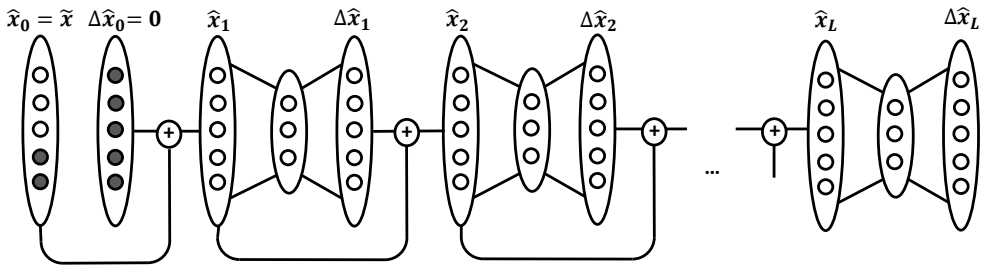


Figure 4: The architecture of an L-layer CRA, where each layer is an RA. The black dots are missing modalities.

has the benefits of leveraging the 2D image structure, utilizing learned features instead of the raw pixels, and reducing the number of parameters of RA. Due to limited ranges of convolution receptive fields, two fully connected (fc) layers are used to enable the long-range interactions among modalities. Similar to RA, we estimate the residual from the corrupted data sample. Depending on the type of layers, the relation between consecutive layers is represented as: $\mathbf{a}^{(l+1)} = f(\mathbf{a}^{(l)} * \mathbf{W}^{(l)} + \mathbf{b}^{(l)})$ ($*$ is a convolution operator) or $\mathbf{a}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{a}^{(l)} + \mathbf{b}^{(l)})$.

In order to learn a convolutional RA, we first train a convolutional and deconvolutional network that maps from an input $\mathbf{x}$ to itself, by estimating the four convolutional filters. We then construct a convolutional RA by adding two fully connected layers and learning their weights, so as to enable interaction among modalities. Finally, both the filters and weights within one convolutional RA are jointly optimized by minimizing the loss in Eqn. 3.

3.4. Cascaded Residual Autoencoder

The proposed cascaded residual autoencoder is constructed by connecting a series of RAs, as shown in Fig. 4. For the first RA, the input is the corrupted data, i.e., $\hat{\mathbf{x}}_0 = \tilde{\mathbf{x}}$. For the remaining RAs, the input is the summation of the input of the last RA and the output of the last RA. Specifically, let $\Delta\hat{\mathbf{x}}_0 = \mathbf{0}$. The input of k-th RA can be represented as $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \Delta\hat{\mathbf{x}}_{k-1}$, where $\Delta\hat{\mathbf{x}}_{k-1}$ is the output of the last RA. Each RA will be learned to minimize the different between current estimation and the complete data, which leads to the loss function of the k-th RA:

$$\mathbb{L}_k = \tfrac{1}{2}\|\Delta\mathbf{x}_k - \Delta\hat{\mathbf{x}}_k\|_2^2 = \tfrac{1}{2}\|(\mathbf{x} - \hat{\mathbf{x}}_k) - \theta^{(k)}_{\mathbf{W},\mathbf{b}}(\hat{\mathbf{x}}_k)\|_2^2. \quad (4)$$

During training, CRA dynamically determines its *optimal depth* by iteratively learning RA and adding it to the current CRA, until the loss function stabilizes. Finally, an L-layer CRA is obtained with L sets of parameters $\mathbf{W}$ and $\mathbf{b}$. To this end, we expect the reconstruction error reduces as increasing layers. Our experiment in Sec. 4.3 shows that, a deeper CRA outperforms a shallow one. When convolutional RAs are used instead of RAs, we call the resultant CRA as convolutional CRA.

3.5. Joint Optimization

As described, a CRA is trained in a forward and layer-wise fashion. Each additional RA is trained to further minimize the reconstruction error of the current CRA stack. Although each single RA added has arrived at a local minimum, we might fine-tune the CRA by jointly considering the stacked RAs. To this end, we develop a joint learning scheme to simultaneously estimate the parameters of all RAs in an CRA, such that the overall reconstruction error can be minimized. Specifically, combining the outputs of all RAs $\Delta\hat{\mathbf{x}}_i$, the estimation is given by:

$$\begin{aligned}\hat{\mathbf{x}} &= \hat{\mathbf{x}}_L + \Delta\hat{\mathbf{x}}_L = (\hat{\mathbf{x}}_{L-1} + \Delta\hat{\mathbf{x}}_{L-1}) + \Delta\hat{\mathbf{x}}_L = ... \\ &= \hat{\mathbf{x}}_0 + \sum\nolimits_{i=1}^{L} \Delta\hat{\mathbf{x}}_i. \end{aligned} \quad (5)$$

The joint loss function for a CRA is defined as:

$$\mathbb{L} = \tfrac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \tfrac{1}{2}\|\mathbf{x} - (\hat{\mathbf{x}}_0 + \sum\nolimits_{i=1}^{L} \Delta\hat{\mathbf{x}}_i)\|_2^2. \quad (6)$$

3.5.1 Optimization via Back-propagation

The loss function of Eqn. 6 depends on the outputs of all RAs, which are controlled by the parameters $\mathbf{W}$ and $\mathbf{b}$. All L sets of parameters $\mathbf{W}$ and $\mathbf{b}$ are learned simultaneously when minimizing this loss. Motivated by the CNN learning, we adopt the commonly used back-propagation scheme to minimize this loss, which relies on the recursively computed derivatives of L w.r.t. each parameter. The following derivation is for CRA, the derivation for convolutional CRA is similar and omitted due to the limited space.

We denote each RA has n_l layers (i.e., the first and last layers corresponding to the input and output respectively), and the value at the l-th layer of k-th RA as $\mathbf{a}_k^{(l)}$. The loss function w.r.t. a single training sample is:

$$\begin{aligned}\mathbb{L} =& \tfrac{1}{2}\|\mathbf{x} - (\hat{\mathbf{x}}_0 + \sum\nolimits_{i=1}^{L} \Delta\hat{\mathbf{x}}_i)\|_2^2 \\ =& \tfrac{1}{2}\|\mathbf{x} - (\tilde{\mathbf{x}} + \sum\nolimits_{i=1}^{L} \mathbf{a}_i^{(n_l)})\|_2^2. \end{aligned} \quad (7)$$

We note that the derivatives w.r.t. parameters (e.g., $\frac{\partial\mathbb{L}}{\partial\mathbf{W}_k^{(l)}}$) can be directly computed from derivatives w.r.t. their associated intermediate layer ($\frac{\partial\mathbb{L}}{\partial\mathbf{a}_k^{(l)}}$). Hence, we first show

Algorithm 1: Computation of gradients

input : Sample $\mathbf{x}$, $\tilde{\mathbf{x}}$, CRA values $\mathbf{a}_k^{(l)}$ with $\tilde{\mathbf{x}}$ as input
output: Derivatives w.r.t. parameters $\nabla_{\mathbf{W}_k^{(l)}}\mathbb{L}, \nabla_{\mathbf{b}_k^{(l)}}\mathbb{L}$

for $k \leftarrow L$ **to** 1 **do**
  // Compute the derivative w.r.t. the output layer of the k-th RA
  $\delta_k^{(n_l)} \leftarrow -\left(\mathbf{x} - \tilde{\mathbf{x}} - \sum_{i=1}^{L} \mathbf{a}_i^{(n_l)}\right)^\intercal$;
  if $k \neq L$ **then**
    // $\delta_k^{(n_l)}$ are calculated using Eqn. 9
    $\delta_k^{(n_l)} \leftarrow \delta_k^{(n_l)} + \delta_{k+1}^{(1)}$;
  // Compute remaining desired derivatives
  for $l \leftarrow n_l - 1$ **to** 1 **do**
    // Compute derivatives w.r.t. each layer
    $\delta_k^{(l)} \leftarrow (\mathbf{W}_k^{(l)}\delta_k^{(l+1)}) \circ f'(\mathbf{W}_k^{(l)}\mathbf{a}_k^{(l)} + \mathbf{b}_k^{(l)})$;
    // Compute derivatives w.r.t. parameters
    $\nabla_{\mathbf{W}_k^{(l)}}\mathbb{L} \leftarrow \delta_k^{(l+1)} f'(\mathbf{W}_k^{(l)}\mathbf{a}_k^{(l)} + \mathbf{b}_k^{(l)}).(\mathbf{a}_k^{(l)})^\intercal$;
    $\nabla_{\mathbf{b}_k^{(l)}}\mathbb{L} \leftarrow \delta_k^{(l+1)} f'(\mathbf{W}_k^{(l)}\mathbf{a}_k^{(l)} + \mathbf{b}_k^{(l)})$;

how to compute derivatives of the loss function w.r.t. the value at autoencoders' l-th layer in a recursive fashion. Let $\delta_k^{(l)} = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_k^{(l)}}$, the relations of $\delta_k^{(l)}$ between layers within the k-th RA is expressed as [29]:

$$\delta_k^{(l)} = (\mathbf{W}_k^{(l)}\delta_k^{(l+1)}) \circ f'(\mathbf{W}_k^{(l)}\mathbf{a}_k^{(l)} + \mathbf{b}_k^{(l)}), \tag{8}$$

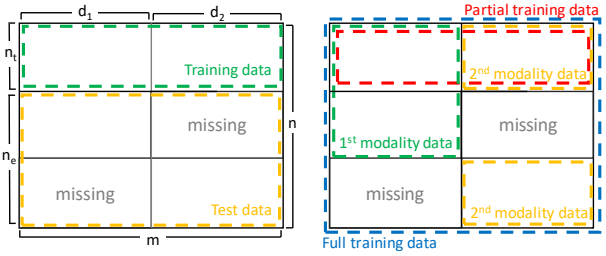
where $\circ$ is element-wise product. The derivatives w.r.t. the output layer $\mathbf{a}^{(n_l)}$ of the k-th RA ($k = 1, 2..., L-1$) are:

$$\begin{aligned}
\delta_k^{(n_l)} &= -\left(\mathbf{x} - \tilde{\mathbf{x}} - \sum_{i=1}^{L} \mathbf{a}_i^{(n_l)}\right)^\intercal \left(\sum_{i=1}^{L} \frac{\partial \mathbf{a}_i^{(n_l)}}{\partial \mathbf{a}_k^{(n_l)}}\right) \\
&= -\left(\mathbf{x} - \tilde{\mathbf{x}} - \sum_{i=1}^{L} \mathbf{a}_i^{(n_l)}\right)^\intercal - \dots \\
&\quad -\left(\mathbf{x} - \tilde{\mathbf{x}} - \sum_{i=1}^{L} \mathbf{a}_i^{(n_l)}\right)^\intercal \left(\sum_{i=k+1}^{L} \frac{\partial \mathbf{a}_i^{(n_l)}}{\partial \mathbf{a}_{k+1}^{(1)}}\right) \\
&= -\left(\mathbf{x} - \tilde{\mathbf{x}} - \sum_{i=1}^{L} \mathbf{a}_i^{(n_l)}\right)^\intercal + \delta_{k+1}^{(1)}.
\end{aligned} \tag{9}$$

Due to the limited space, the full derivation of Eqn. 9 is omitted here. Equations 8 and 9 allows us to recursively compute the derivatives w.r.t. the network parameters. The detailed algorithm is described in Algorithm 1.

3.5.2 Learning Strategy

There are two different learning strategies in training a CRA. One is called *one-shot CRA*, where each RA is trained sequentially, and added to the CRA until the reconstruction error cannot be further reduced. After that, the joint optimization updates the parameters of all RAs simultaneously. The other is called *aggressive CRA*, where the joint optimization of the entire network is performed after each RA being added into the CRA. In other words, this allows the overall loss to be further reduced before adding the next RA, which may have the potential benefit of faster convergence. We compare these two strategies in the experiments.



(a) (b)

Figure 5: (a) Data partition for imputation experiments; (b) Various types of training data for recognition experiments.

4. Experimental Results

The objectives of experiments are to evaluate different methods on data imputation for missing modalities, as well as on using imputed data for object recognition. Thus, we have two main experiments: imputation experiments and recognition experiments, each serving one objective. We implement CRA with Caffe and share the code here.

4.1. Experimental Protocols

Since imputation experiments require uncorrupted data for evaluation, we cannot utilize databases with "real-world" missing modalities. Instead, given a multi-modal database, we synthesize the missing modality by removing some modalities for part of training samples. Fig. 5a shows the partition of training and test data for imputation experiments. Note that the training data is used by autoencoder-based, not matrix completion methods. In general, one might expect these training data have no missing modalities. In practices, we can leverage any data samples as long as they have at least *two* observed modalities, among which we may remove one or more modalities and learn to recover them via the rest observed modalities. For these data samples, missing portion would not be evaluated in the loss function. Hence, the test and training portions may overlap for data with over two modalities. It is the strength of CRA to even leverage data with missing modality for imputation training. After different imputation methods recover missing modalities, the recognition experiments utilize classifiers trained on four types of data as shown in Fig. 5b: (1) partial training data with samples that are available for all modalities, (2) available portion of each single modality, (3) full training data without missing modality, and (4) training data recovered using different imputation methods.

Evaluation Metrics For a comprehensive evaluation of imputation experiments, we use two metrics, each from a different perspective. Normalized Mean Squared Error

Table 1: Properties of four datasets, in the number of modalities (m), data dimensions (d), number of training samples (n_t) and test samples (n_e) for imputation, missing ratio (r), number of classes (c), training samples per class (n), and test samples per class (k) in recognition experiments. The missing ration is the ratio between the number of missing elements and total number of elements.

Dataset	m	d	n_t	n_e	$r(\%)$	c	n	k
GRSS	2	[111, 37]	~ 20	~ 180	45	15	~ 200	~ 1000
RGB-D	2	[2500, 2500]	~ 170	~ 513	40	51	~ 683	~ 138
MTPIE	5	[1024,...,1024]	1529	729	50	137	1529	781
HSFD	24	[625, ..., 625]	76	38	40	38	76	$2-5$

(NMSE) is a generic metric, NMSE = $\frac{\|\mathbf{X}-\hat{\mathbf{X}}\|_F}{\|\mathbf{X}\|_F}$, where $\mathbf{X}, \hat{\mathbf{X}}$ are the original and recovered data matrices and $\|.\|_F$ is the Frobenious norm. The Peak Signal to Noise Ratio (PSNR) is commonly used to quantify image compression and reconstruction. In our problem, since data samples are normalized to the range of $[0, 1]$, we have PSNR $= 10\log_{10}\frac{d}{(\mathbf{x}-\hat{\mathbf{x}})^T(\mathbf{x}-\hat{\mathbf{x}})}$. For recognition experiments, we use the recognition rate on the test set as the metric.

4.2. Datasets

We use four benchmark datasets: the 2013 GRSS Data Fusion Contest Dataset (GRSS), the RGB-D Object Dataset (RGB-D), Multi-PIE (MTPIE), and the hyperspectral face dataset from Hong Kong Polytechnic University (HSFD). Table 1 summarizes main properties of four datasets.

GRSS The GRSS dataset [8] includes a hyperspectral image (HSI), a LiDAR-derived digital surface model of a university campus. There are 15 classes including both natural and man-made objects. For the classification task, we follow the framework in [21]. Specifically, the spatial features in the hyperspectral and LiDAR data are extracted via the morphological attribute profile [7]. Then, the MLRsub classifier [23] is used for classification. Markov random fields [24] is also employed for spatial regularization to promote spatial smoothness in the final classification results.

RGB-D RGB-D dataset [22] contains $41,877$ RGB-D images of 300 physically distinct everyday objects, organized into 51 classes. This dataset contains both textured and texture-less objects, with large lighting variations. We impute the missing data in either raw depth or grayscale images. To create a data matrix, we place all images at the center of a 200×200 empty image and downsample to 50×50. For classification, we use hierarchical matching pursuit [2] to extract features, and then employ linear SVM. Following the experimental setting in [22], we leave one object instance out from each category for testing, and train on the remaining $300 - 51 = 249$ objects at each trial.

MTPIE The Multi-PIE dataset [14] consists of $754,200$ face images for 337 subjects with variations in time frames, poses, expressions, and illuminations. In our experiments, we only use faces with frontal illumination and neutral expression. Each sample consists of five different poses in

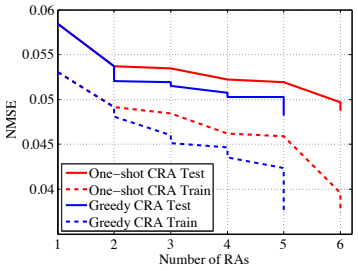


Figure 6: Learning strategy comparison. Vertical lines, occurring after adding each RA to aggressive CRA and once in one-shot CRA, show improvements due to joint optimization.

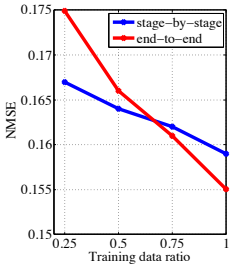


Figure 7: Imputation performance of two learning approaches with different amount/ratio of training data.

ranges of $[0^o, 60^o]$ corespondent to five modalities (images with negative poses is flipped to double data samples). For classification, we use the first 200 subjects for training and remaining 137 subjects for testing. Each test subject has a frontal image as a gallery; we match a face image with an arbitrary pose to a frontal gallery face. Features are extracted using a simple CNN (three conv and two fc layers).

HSFD HSFD [10] includes hyperspectral face images of 48 subjects. Each of the first 25 subjects has four to seven cubes while the rest 23 subjects only have one cube per subject. Each cube contains 33 bands (i.e., modalities) covering the 400 to 720 nm spectral range. Following the experimental protocol of [10], our experiments use the first 25 subjects comprising 113 total cubes. The first six and the last three bands are very noisy and discarded as suggested by [10]. For each subject, two cubes are randomly selected as gallery and the remaining 63 cubes as probes. Faces are cropped using eye coordinates and resized to 25×25. For each subject, one of two gallery cubes is corrupted in multiple bands. Recovered face cubes are fused into 2D face images using the spatiospectral fusion method [32], and the recognition is performed by collaborative representation [36].

4.3. Results of Imputation Experiments

Learning strategy Fig. 6 compares the imputation performance of one GRSS class during the training and test stages, for two learning strategies of CRA. This experiment shows that both strategies lead to similar final imputation performance, while the aggressive CRA requires less number of RAs (3.4 RAs vs 5.6 RAs of one-shot CRA, averaged over 15 classes). This is expected since the aggressive CRA performs joint optimization more often, and hence converges faster. For the rest experiments, we use the aggressive CRA since it is more efficient due to its smaller depth.

We also compare our proposed layer-by-layer learning approach with end-to-end learning. In all four datasets, we observe that the former always achieves better performance than the latter. We hypothesize this is because the number of training samples is not sufficiently large, which is usually

Table 2: Affects of CRA depth on HSFD performance.

Depth	PSNR	NMSE	Recognition rate (%)
1	26.27	0.233	76.42
2	27.42	0.223	77.79
3	27.69	0.218	77.88
4	27.91	0.213	77.95
5	28.01	0.209	78.03

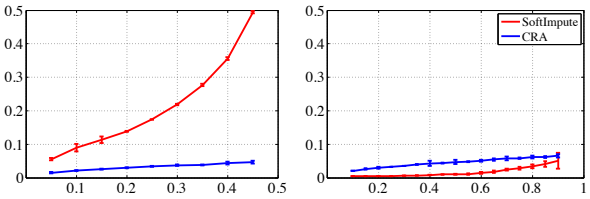


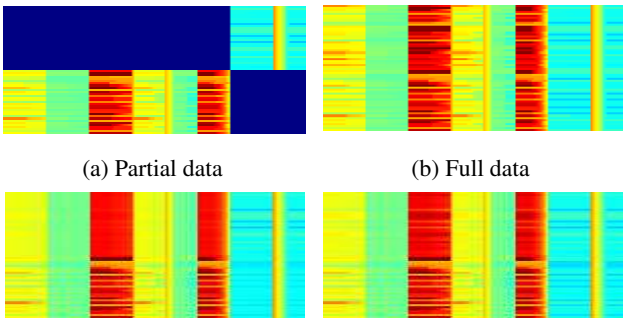
Figure 8: Compare blockwise (left) and elementwise (right) missing data imputation with different missing ratios on GRSS class of soil. In the blockwise corruption, we have to keep at least 1 of 2 modalities, hence the missing ratio must be less than 0.5.

the case for multi-modal datasets. To validate this hypothesis, we modify the experiment setting of MTPIE by using face images with all illuminations and expressions. Samples from 200 subjects are used for training and remaining 137 subjects are used to evaluate imputation, which results in $62,767$ and $32,167$ training and testing samples, respectively. Fig. 7 shows that layer-by-layer learning is more suitable when there are limited training samples (e.g., 25%, and 50% of the full training set).

Affect of CRA depth We explore how the number of RAs in CRA (i.e., CRA depth) affects the performance. Specifically, from optimized convolutional CRAs, different versions of CRA are generated based on its depth (the number of RAs). On HSFD, their performances are reported in Tab. 2. The table shows that stacking RAs to build a deep architecture improves both imputation and object recognition, and deeper CRA outperforms shallow ones.

MAR vs missing modality Low-rank matrix completion methods work well by assuming MAR. However, missing continuous blocks in multi-modal data can break these assumptions. In this experiment, we compare a matrix completion method (SoftImpute) with CRA in both types of corruption. Fig. 8 shows the imputation error of these two methods with different missing ratios. SoftImpute, as expected, works well when corruption is elementwise random. However, when data is missing in blocks, SoftImpute peforms significantly worse than CRA, e.g., the imputation error grows exponentially when the missing ratio increases. From Fig. 8, we also observe that CRA produces similar errors regardless the type of corruption.

Imputation results on benchmark datasets This experiment evaluates different imputation methods in recovering the missing modalities. Among eights baseline methods, three are classic matrix completion methods, including SVT [3], SoftImpute [26] and OptSpace [20], all using the authors' implementation. The other five are autoencoder-based methods, including Genetic Algorithm (EA&GA) [1], Denoising Autoencoder (DA) [33], stacked Denoising Autoencoder (sDA) [33], multi-modal autoencoder (Mul-mod AE) [30], deep canonically correlated autoencoders (DCCAE) [34], all using our own implementation. The CRA dynamically determines optimal depth by itself. CRA depths vary from 3 to 6 depending on specific dataset/class, with an average of 4.6 across four datasets. To have fair comparisons and consider that benchmark networks have pre-defined depths, we fix their depths to be 5. With cross validation, we find that the hidden layer size being half of the input size works well for all methods. All convolutional CRAs use 3×3 filters with stride of 2 (Fig. 3). For GRSS and RGB-D, the number of autoencoder models is the same as the number of classes (c). For face datasets (MTPIE, HSFD), we train only one model for all subjects.



(a) Partial data (b) Full data

(c) sDA (d) CRA

Figure 9: Color-coded visualization of imputed test matrices on GRSS dataset. Each row is a 137-d data sample of HIS (left) and LiDAR feature (right). Color from blue to red corresponds to a value in the range of $[0, 1]$ with 0 (blue) indicates missing entries.

Table 3 shows the imputation error of different methods in four datasets. Our proposed CRA consistently outperforms baseline methods in all four datasets. The CRA with joint optimization is slightly better than the one without, and convolutional CRA further improves the performance. It is worth noting that PSNR increased by 1 is considered substantial improvement in image compression [31]. For NMSE, even though the absolute margins between the top two results appear small, the relative margins show our improvement. Further, despite numerically DA and sDA seem to have imputation errors close to CRA, they actually recover data closed to the mean of training samples. Fig. 9 visualizes the imputed data of the soil class in GRSS. The CRA can clearly recover more individual characteristics than sDA. The contrast between Fig. 9 and the relative margin of NMSE shows the limitation of NMSE in quantifying the recovery of detailed information. Fig. 10 shows examples of imputed grayscale and depth images from the

Table 3: Comparision of imputation errors and recognition rates in four datasets. "-" denotes an unconverged solution. Empty space denotes inapplicable value, e.g., DCCAE can not be applied to datasets with more than two modalities, or convolutional CRA can not be used on GRSS samples since they are not 2D images. The best and second best results as well as upper and lower bounds are labeled in boldface and italic, respectively. The relative margin is the ratio (%) of the difference between the top two results to the second best result, or to the difference between the upper and lower bounds.

		PSNR				NMSE				Recognition rate (%)			
		GRSS	RGB-D	MTPIE	HSFD	GRSS	RGB-D	MTPIE	HSFD	GRSS	RGB-D	MTPIE	HSFD
Methods	SVT [3]	-	21.51	17.02	14.47	-	0.548	0.307	0.633	-	21.17	38.73	74.24
	SoftImpute [26]	17.27	22.39	19.86	*26.33*	0.271	0.401	0.241	*0.232*	86.22	23.38	40.52	*76.36*
	OptSpace [20]	12.88	20.17	-	23.82	0.421	0.611	-	0.353	83.97	20.79	-	75.02
	AE&GA [1]	26.81	23.01	20.73	18.23	0.105	0.362	0.225	0.401	84.01	68.83	42.33	74.24
	DA [33]	29.62	24.05	22.09	24.38	0.079	0.341	0.171	0.276	85.77	69.17	44.34	76.02
	sDA [33]	29.74	24.07	23.31	20.13	*0.079*	*0.331*	0.151	0.372	86.02	69.54	45.32	75.96
	Mul-mod AE [30]	30.01	*24.81*	*25.91*	18.23	0.105	0.362	*0.113*	0.401	86.01	*70.21*	*54.24*	76.20
	DCCAE [34]	*30.05*	24.11			0.079	0.341			*86.25*	69.94		
	CRA w/o opt	30.80	25.25	25.99	26.63	0.077	0.265	0.112	0.225	86.38	70.63	54.32	76.58
	CRA w/ opt	**31.04**	25.93	26.55	27.51	**0.076**	0.248	0.105	0.222	**86.42**	71.04	56.42	77.88
	Conv CRA		**26.12**	**27.05**	**28.01**		**0.234**	**0.093**	**0.209**		**71.81**	**57.10**	**78.03**
Bounds	Full data									**88.46**	**78.32**	**59.24**	**80.00**
	1^{st} modality only									79.52	*68.72*		
	2^{nd} modality only									63.91	64.48		
	Partial data									*83.24*	56.51	*41.14*	*72.21*
	Relative margin	3.29	5.28	4.40	6.38	3.79	2.93	17.70	9.91	3.26	12.50	16.02	21.43

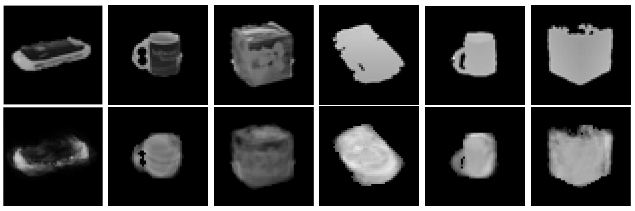
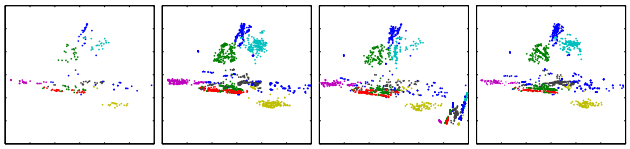


Figure 10: Imputed images by CRA. Left three columns are grayscale images and the right three columns are depth images. The top row is original images that are missing. The bottom row is imputed images using the available modality via CRA.

RGB-D dataset. CRA can reconstruct images well approximating the original ones.

4.4. Results of Recognition Experiments

The recognition rates of classifiers trained on imputed training data are reported in Tab. 3. These results are also compared with baselines where classifiers are trained on different amount of data: full training data, partial training data or a single modality, as in Fig. 5b. We observe that in all datasets, CRA with optimization or convolutional CRA achieves the best performance among all methods. Also, the achieved performance is more closer to the upper bound than to the lower bound, which shows the power of learning with missing modality. Finally, since the difference of the two bounds defines the potential range of performance that could be achieved by the classifier learned from the imputed training data, we use such a difference as the normalization to compute the relative margin. The average relative margin of 13% across four datasets demonstrates the effectiveness of CRA in object recognition with missing modalities.



(a) Partial data (b) Full data (c) SoftImpute (d) CRA

Figure 11: Visualization of all training data of GRSS (recovered and original) in the same space spanned by the first two principal components of full data. Colors represent classes.

Fig. 11 visualizes the distribution of training samples of GRSS, before and after the imputation of two methods. CRA produces the similar distribution as the original full data. Meanwhile the matrix completion method of SoftImpute generates some outliners, at the lower-right corner of Fig. 11c. This figure also helps to explain why using imputed data samples may benefit classifier learning, i.e., the original data distribution is well represented by the imputed data, but not the partial data.

5. Conclusions

Motivated by the need to exploit the advantage of multimodal data, this paper draws attention to a relatively less explored problem of imputing data with missing modality. To this end, we propose a novel approach to combine a series of residual autoencoders into a cascaded architecture to learn complex relationship among data from different modalities. The cascaded residual autoencoder provides a data imputation framework that leverages strengths of residual learning and autoencoder networks. Extensive experiments on benchmark datasets demonstrate the superiority of our algorithm in both data imputation and object recognition.

References

[1] M. Abdella and T. Marwala. The use of genetic algorithms and neural networks to approximate missing data in database. In *Computational Cybernetics, 2005. ICCC 2005. IEEE 3rd International Conference on*, pages 207–212. IEEE, 2005. 2, 7, 8

[2] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013. 6

[3] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. 2, 7, 8

[4] E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010. 2

[5] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009. 2

[6] J. A. Cruz, X. Yin, X. Liu, S. M. Imran, D. D. Morris, D. M. Kramer, and J. Chen. Multi-modality imagery database for plant phenotyping. 7:1–15, October 2015. 1

[7] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. Morphological attribute profiles for the analysis of very high resolution images. *Geoscience and Remote Sensing, IEEE Transactions on*, 48(10):3747–3762, 2010. 6

[8] C. Debes, A. Merentitis, R. Heremans, J. Hahn, N. Frangiadakis, T. van Kasteren, W. Liao, R. Bellens, A. Pizurica, S. Gautama, et al. Hyperspectral and LiDAR data fusion: Outcome of the 2013 GRSS data fusion contest. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 7(6):2405–2418, 2014. 6

[9] S. M. Dhlamini, F. V. Nelwamondo, and T. Marwala. Condition monitoring of HV bushings in the presence of missing data using evolutionary computing. *WSEAS Transactions on Power Systems*, 1(2):280–287, 2006. 2

[10] W. Di, L. Zhang, D. Zhang, and Q. Pan. Studies on hyperspectral face recognition in visible spectrum with feature band selection. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(6):1354–1361, 2010. 1, 6

[11] Y. Duan, L. Yisheng, W. Kang, and Y. Zhao. A deep learning based approach for traffic data imputation. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 912–917. IEEE, 2014. 2, 3

[12] J. Elseberg, D. Borrmann, and A. Nüchter. Full wave analysis in 3D laser scans for vegetation detection in urban environments. In *Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on*, pages 1–7. IEEE, 2011. 1

[13] L. Gomez-Chova, D. Tuia, G. Moser, and G. Camps-Valls. Multimodal classification of remote sensing images: a review and future directions. *Proceedings of the IEEE*, 103(9):1560–1584, 2015. 1

[14] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. *Image and Vision Computing*, 28(5):807–813, 2010. 6

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[16] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[17] J. Im and J. R. Jensen. Hyperspectral remote sensing of vegetation. *Geography Compass*, 2(6):1943–1961, 2008. 1

[18] J. Jordan and E. Angelopoulou. Supervised multispectral image segmentation with power watersheds. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1585–1588. IEEE, 2012. 1

[19] A. Kanaev, B. Daniel, J. Neumann, A. Kim, and K. Lee. Object level HSI-LiDAR data fusion for automated detection of difficult targets. *Optics express*, 19(21):20916–20929, 2011. 1

[20] R. H. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 324–328. IEEE, 2009. 2, 7, 8

[21] M. Khodadadzadeh, J. Li, S. Prasad, and A. Plaza. Fusion of hyperspectral and LiDAR remote sensing data using multiple feature learning. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 8(6):2971–2983, 2015. 1, 6

[22] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011. 6

[23] J. Li, J. M. Bioucas-Dias, and A. Plaza. Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(3):809–823, 2012. 6

[24] S. Z. Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009. 6

[25] E. H. Lim and D. Suter. 3D terrestrial LiDAR classifications with super-voxels and multi-scale conditional random fields. *Computer-Aided Design*, 41(10):701–710, 2009. 1

[26] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010. 2, 7, 8

[27] V. Miranda, J. Krstulovic, H. Keko, C. Moreira, and J. Pereira. Reconstructing missing data in state estimation with autoencoders. *Power Systems, IEEE Transactions on*, 27(2):604–611, 2012. 3

[28] F. V. Nelwamondo, S. Mohamed, and T. Marwala. Missing data: A comparison of neural network and expectation maximisation techniques. *arXiv preprint arXiv:0704.3474*, 2007. 2, 3

[29] A. Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011. 5

[30] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 689–696, 2011. 3, 7, 8

[31] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3):243–250, 1996. 7

[32] M. Uzair, A. Mahmood, and A. Mian. Hyperspectral face recognition with spatiospectral information fusion and PLS regression. *Image Processing, IEEE Transactions on*, 24(3):1127–1137, 2015. 6

[33] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. of Int. Conf. Machine Learning (ICML)*, pages 1096–1103. ACM, 2008. 2, 7, 8

[34] W. Wang, R. Arora, K. Livescu, and J. Bilmes. On deep multi-view representation learning. In *Proc. of Int. Conf. Machine Learning (ICML)*, pages 1083–1092, 2015. 3, 7, 8

[35] L. Z. Wong, H. Chen, S. Lin, and D. C. Chen. Imputing missing values in sensor networks using sparse data representations. In *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 227–230. ACM, 2014. 2, 3

[36] L. Zhang, M. Yang, and X. Feng. Sparse representation or collaborative representation: Which helps face recognition? In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 471–478. IEEE, 2011. 6