Accepted for presentation at L2ID Workshop at CVPR 2021
19 - 25 June, 2021

Learning from Incomplete Features by Simultaneous Training of Neural Networks and Sparse Coding

Cesar F. Caiafa*†, Ziyao Wang‡, Jordi Solé-Casals§, and Qibin Zhao*
Tensor Learning Team, Center for Advanced Intelligence Project, RIKEN, JAPAN*
Instituto Argentino de Radioastronomía, CONICET-CCT La Plata / CIC-PBA / UNLP, ARGENTINA†
School of Automation, Southeast University, CHINA†‡
University of Vic - Central University of Catalonia, SPAIN§
Corresponding author: C. F. Caiafa (Email: ccaiafa@gmail.com)

***Abstract*—In this paper, the problem of training a classifier on a dataset with incomplete features is addressed. We assume that different subsets of features (random or structured) are available at each data instance. This situation typically occurs in the applications when not all the features are collected for every data sample. A new supervised learning method is developed to train a general classifier, such as a logistic regression or a deep neural network, using only a subset of features per sample, while assuming sparse representations of data vectors on an unknown dictionary. Sufficient conditions are identified, such that, if it is possible to train a classifier on incomplete observations so that their reconstructions are well separated by a hyperplane, then the same classifier also correctly separates the original (unobserved) data samples. Extensive simulation results on synthetic and well-known datasets are presented that validate our theoretical findings and demonstrate the effectiveness of the proposed method compared to traditional data imputation approaches and one state-of-the-art algorithm.**

I. Introduction

Learning methods from limited or imperfect data has attracted great attention in the literature recently. Datasets with limited, weak, noisy labels or incomplete features represent an important and still open problem. In this paper, we address the problem of training a classifier on a dataset with incomplete features, which arises in many machine learning applications where sometimes the measurements are incomplete, noisy or affected by artifacts. Examples of this situation include: recommendation systems built upon the information gathered by different users where not all the users have fully completed their forms; medical datasets where typically not all tests can be performed on every patient; or a self-driving vehicle or robot where objects in the view field can be partially occluded.

Handling correctly the incomplete-features problem is a classical challenge in machine learning. Skipping missing features by setting them to zero values damages the classification accuracy [1]. Most previous studies addressed this problem by using an imputation approach, which consists of performing data completion followed by training the classifier with those reconstructions (referred here as the sequential method). However, this strategy cannot ensure the statistical consistence of the classifier, as data completion is usually fully unsupervised or label information is partially or inefficiently exploited.

In this work, a new supervised learning method is developed to train a general classifier, such as a logistic regression or a deep neural network, using only a subset of features per sample, while assuming sparse representations of data vectors on an unknown dictionary. The proposed method simultaneously learns the classifier, the dictionary and the corresponding sparse representation of each input data sample. In this way, we combine the approximation power and simplicity of sparse coding with the extraordinary ability of neural networks (NNs) to model complex decision functions (classifiers) with the goal to successfully train a classifier based on incomplete features.

We analyze the limitations of the sequential approach (section I-C), *i.e.* imputation followed by training, and introduce the simultaneous classification and coding approach in section II. Our method consists of incorporating a sparse data representation model into a single cost function that is optimized for training the classifier and, at the same time, finding the best representation of the observed data. A learning algorithm is presented in section II-A to train a classifier on incomplete features and sufficient conditions under which such a classifier performs as good as the ideal classifier, *i.e.* the one that can be obtained from complete observations, is identified (section II-B). Extensive experimental results are presented in section III, using synthetic and well known benchmark datasets that validate our theoretical findings and illustrate the effectiveness of the proposed method.

A. Related work

Practical sequential methods based on statistical imputation, such as computing the "mean", "regression" and "multiple" imputation techniques are common practice [2]. Remarkably, it was shown that imputing with a constant, *e.g.* the mean, is Bayes-risk consistent only when missing features are not informative [3]. More elaborated completion methods were also explored, such as K-nearest neighbor estimators, multilayer or recurrent NNs and others, see [4] and references therein.

However, sequential methods do not fully exploit label information. Data labels can provide valuable information about missing features that could potentially improve the classifier learning process. Recent advances on probabilistic generative models have allowed for a formulation of supervised learning with incomplete features as a statistical inference problem

arriving at algorithms that significantly outperformed sequential methods. In the seminal work [5], a framework for maximum likelihood density estimation based on mixtures models was proposed and successfully applied to small incomplete-features problems. In particular, a Gaussian Mixture Model (GMM) was fitted to incomplete-data through an Expectation-Maximization (EM) algorithm. Building upon this generative model strategy, some approaches have considered integrating out the missing values based on a simple logistic regression function [6], [7]. Other versions of this approach proposed an explicit simultaneous learning of the model and the decision function [8], [9]. While probabilistic generative models provided a nice and elegant approach to the incomplete-features problem showing good results on small datasets, they are not suitable for many modern machine learning applications because: (1) despite some acceleration techniques were explored, *e.g.* [10], [11], those algorithms are computationally expensive becoming prohibitive for moderate to large datasets; (2) GMM is impractical for modeling high-dimensional datasets because the number of parameters to achieve good approximations becomes unmanageable; and (3) they do not consider complex classification functions as the ones provided by deep NN architectures.

Recently, some approaches based on the low-rank property of the features data matrix were investigated and algorithms for data completion were proposed incorporating the label information [12], [13], [14]. Since the rank estimation of a matrix is a computationally expensive task, usually based on the Singular Value Decomposition (SVD), the obtained algorithms are prohibitive to solve modern machine learning problems with large datasets. Additionally, as in the case of the probabilistic generative models, none of these methods considered complex classification functions. To overcome this drawback, more recently, a framework based on various NN architectures such as autoencoders, multilayer perceptrons and Radial Basis Function Networks (RBFNs), was proposed for handling missing input data by setting a probabilistic model, *e.g.* a GMM, for every missing feature, which is trained together with the NN weights [15]. This method combined the great capability of NNs to approximate complex decision functions with the nice formulation of the GMM to model missing data. However, it inherited the drawbacks of GMMs, *i.e.* they are not well suited to higher-dimensional datasets.

On the other hand, during the last few years in the signal processing community, there has been a rapid development of theory and algorithms for sparse coding approximations which, by exploiting the redundancy of natural signals, are able to provide simple and accurate models of complex data distributions, see [16], [17], [18], [19], [20] and references therein. Sparse coding is nearly ubiquitous in Nature, for example, it is found in the way that neurons encode sensory information [21], [22]. Sparse representations of data showed to be useful also in classification problems. In [23], a Linear Discriminant Analysis (LDA) classifier was trained on corrupted data providing a robust classification method. In [24], algorithms for learning *discriminative* sparse models, instead of purely *reconstructive* ones, were proposed based on simple linear and bilinear classifiers. Similar methods were also studied by either using class-specific dictionaries [25], [26] or using a single one for all classes [27]. However, these proposed methods neither were applied to the incomplete-features problem nor considered deep NN classifiers.

B. Problem formulation

We assume a supervised learning scenario with vector samples and labels $\{\mathbf{x}_i, y_i\}$, $i = 1, 2, \ldots, I$, $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in \{0, 1, \ldots, C-1\}$ (C classes). However, we are constrained to observe only subsets of features and their labels: $\{\mathbf{x}_i^o, y_i\}$, $\mathbf{x}_i^o \in \mathbb{R}^{M_i}$ with $M_i < N$. Unobserved (missing) features are denoted by $\mathbf{x}_i^m \in \mathbb{R}^{N-M_i}$. We consider arbitrary patterns of missing features, which are allowed to be different for each data instance i. The set of indices of missing features at sample i is denoted by $\mathcal{M}_i$, *i.e.* $\mathbf{x}_i^m = \mathbf{x}_i(\mathcal{M}_i)$ and $\mathbf{x}_i^o = \mathbf{x}_i\left(\overline{\mathcal{M}}_i\right)$.

We define the set of all K-sparse vectors $\Sigma_K^P = \{\mathbf{s} \in \mathbb{R}^P \text{ s.t. } \|\mathbf{s}\|_0 \leq K\}$ (containing at most K non-zero entries) and assume that data vectors $\mathbf{x}_i$ admit K-sparse representations over an unknown dictionary $\mathbf{D} \in \mathbb{R}^{N \times P}$ ($P \geq N$):

$$\mathbf{x}_i = \mathbf{D}\mathbf{s}_i, \text{ with } \mathbf{s}_i \in \Sigma_K^P. \tag{1}$$

The columns of a dictionary are called "atoms" because every data vector can be written as a linear combination of at most K elementary components. Sometimes dictionaries are orthogonal such as the ones derived from the Discrete Cosine or Wavelet [19] transforms. However, overcomplete ($P \geq N$) nonorthogonal dictionaries have demonstrated to play an important role in image processing tasks such as denoising, inpainting, etc [17], [28].

By partitioning $\mathbf{D}$ according to the pattern of missing features at sample i, we obtain $\mathbf{D}_i^o = \mathbf{D}\left(\overline{\mathcal{M}}_i, :\right) \in \mathbb{R}^{M_i \times P}$ and $\mathbf{D}_i^m = \mathbf{D}(\mathcal{M}_i, :) \in \mathbb{R}^{(N-M_i) \times P}$, which according to equation (1) implies:

$$\mathbf{x}_i^o = \mathbf{D}_i^o \mathbf{s}_i, \text{ and } \mathbf{x}_i^m = \mathbf{D}_i^m \mathbf{s}_i. \tag{2}$$

Let us assume that a perfect classifier, *e.g.* a logistic regression or deep NN, that assigns probability $p_\Theta(\hat{y}|\mathbf{x})$ to predicted label $\hat{y}$ given data $\mathbf{x}$ can be trained on the complete dataset $\{\mathbf{x}_i, y_i\}$, such that, in a two-classes scenario ($C = 2$), $p_\Theta(\hat{y} = y_i|\mathbf{x}_i) > p_\Theta(\hat{y} \neq y_i|\mathbf{x}_i)$, $\forall i = 1, 2, \ldots, I$, where Θ is the set of trained parameters. Our goal is to develop a method to obtain an estimate $\hat{\Theta}$ of parameters using only the incomplete dataset $\{\mathbf{x}_i^o, y_i\}$ and to identify conditions under which such a classifier is compatible with the ideal one.

C. Why training after imputation is difficult?

If the K-sparse representations of the observations $\mathbf{x}_i^o$ were unique, then $\mathbf{x}_i$ can be perfectly reconstructed from the incomplete observations and the classifier can be successfully trained using these reconstructions. In the particular case where the dictionary is known in advance, there exist conditions on the sampling patterns based on the *coherence*, *spark* or *RIP (Restricted Isometry Property)* of matrix $\mathbf{D}_i^o$ that can guarantee uniqueness [20]. However, these conditions are difficult to

meet in practice and determining RIP/Spark properties are NP-hard in general [29]. Moreover, in the general case where the dictionary $\mathbf{D}$ is unknown and needs to be learned from data, it is even more difficult to obtain well separated reconstructions which certainly leads to suboptimal or wrong classifiers.

Next, we provide some intuition about the limitation of the sequential approach through a toy example. Let us consider the classification of hand-written digit images belonging to two classes: "3s" and "8s" and assume that they admit 2-sparse representations over a dictionary. Fig. 1 (a-b) shows the representations of two example vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ belonging to classes "3" and "8", respectively. If only the right halves of the images are observed and no label information is provided, we are clearly faced with a problem because our observed samples from two different classes are identical, *i.e.* $\mathbf{x}_i^o = \mathbf{x}_j^o$. It is obvious that at least two possible 2-sparse representations for the observed data exist as illustrated in Fig. 1(c). When the sparse solution is not unique, we may end up reconstructing wrong vectors that could not be even well separated as illustrated in Fig. 1 (d-e). In general, sequential methods using only the information of observed features are prone to fail because the non-uniqueness of solutions can make the training of a good classifier an impossible task. However, we could solve this problem by incorporating the labelling information from the very beginning as it is proposed in the following section.

II. Simultaneous Learning and Coding Approach

We propose to train the classifier and find the proper representation, not only as sparse as possible but also providing the best separation of classes. We want to combine the training of the classifier together with the learning of a dictionary and optimal sparse representations such that the reconstructed data vectors are compatible with observations and well separated. To do that we propose to minimize the following global cost function:

$$J(\Theta, \mathbf{D}, \mathbf{s}_i) = \underbrace{\frac{1}{I}\sum_{i=1}^{I}\left\{J_0(\Theta, \hat{\mathbf{x}}_i, y_i) + \lambda_1 J_1(\mathbf{D}, \mathbf{s}_i)\right\}}_{F(\Theta, \mathbf{D}, \mathbf{s}_i)} + \underbrace{\frac{1}{I}\sum_{i=1}^{I}\left\{\lambda_2 J_2(\mathbf{s}_i)\right\}}_{G(\mathbf{s}_i)}, \tag{3}$$

with respect to Θ, $\mathbf{D}$ and $\mathbf{s}_i$ ($i = 1, 2, \ldots, I$), where Θ contains the classifier parameters, *i.e.* the vector of weights in a deep NN classifier architecture; $\mathbf{D} \in \mathbb{R}^{N \times P}$ ($P \geq N$) is a dictionary and $\mathbf{s}_i \in \Sigma_K^P$ are the representation coefficients such that the reconstructed data vectors are $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i$.

$J_0(\Theta, \hat{\mathbf{x}}_i, y_i)$ is a measure of the classification error for the reconstructed sample vector $\hat{\mathbf{x}}_i$. Typically, we use the crossentropy measure, *i.e.* $J_0(\Theta, \hat{\mathbf{x}}_i, y_i) = -\log[p_\Theta(y_i|\hat{\mathbf{x}}_i)]$, where $p_\Theta(y_i|\hat{\mathbf{x}}_i)$ is the probability assigned by the classifier to sample $\hat{\mathbf{x}}_i$ as belonging to class y_i. $J_1(\mathbf{D}, \mathbf{s}_i)$ is a measure of the approximation error of the reconstruction when it is restricted to observed features, which is defined as follows: $J_1(\mathbf{D}, \mathbf{s}_i) = \frac{M_i}{N}\|\mathbf{m}_i \odot (\mathbf{x}_i - \mathbf{D}\mathbf{s}_i)\|^2$, where $\odot$ stands for the entry-wise product, $\mathbf{m}_i \in \mathbb{R}^N$ is the observation mask for sample i, *i.e.* $m_i(n) = 0$ (1) if data entry $\mathbf{x}_i(n)$ is missing (available); and $J_2(\mathbf{s}_i) = \frac{1}{N}\|\mathbf{s}_i\|_1$ is proportional to the ℓ_1-norm whose minimization promotes the sparsity of the representation since ℓ_1-norm is a convenient proxy for ℓ_0-norm [30]. Finally, the hyper-parameters λ_1 and λ_2 allow us to give more or less importance to the representation accuracy and its sparsity, with respect to the classification error. Intuitively, minimizing equation (II) favors solutions that not only have sparse representations compatible with observed features, but also providing reconstructions that are best separated in the given classes.

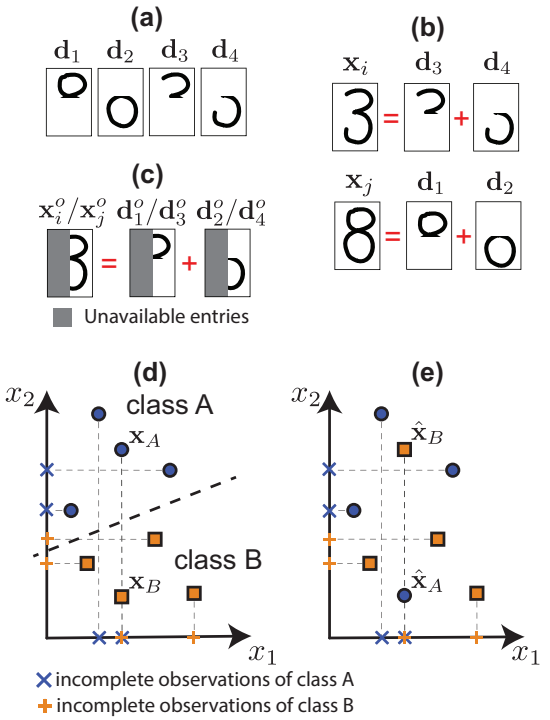

Fig. 1: Toy example: (a) 4 out P dictionary elements $\mathbf{d}_i$ (atoms). (b) Digits "3" and "8" can be represented by combining only two atoms in the dictionary (2-sparse representations). (c) A left-half occluded digit "3" or "8" admits more than one 2-sparse representation (sum of $\mathbf{d}_1^o$ and $\mathbf{d}_2^o$, or $\mathbf{d}_3^o$ and $\mathbf{d}_4^o$). (d) Linearly separable samples from two classes (A and B) having two features: $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$ where incomplete observations are taken by observing only one feature. Note that $\mathbf{x}_A$ and $\mathbf{x}_B$ belong to different classes but their observations are identical. (e) Without using label information, the sequential method could lead to wrong reconstructions of data vectors, *i.e.* $\hat{\mathbf{x}}_A \neq \mathbf{x}_A$ and $\hat{\mathbf{x}}_B \neq \mathbf{x}_B$ making the set of reconstructed vectors not linearly separable.

A. A sparsity-promoting sub-gradient optimization algorithm

To minimize the cost function in equation (II) we propose to alternate between the optimization over $\mathbf{s}_i$ ($i = 1, 2, \ldots, I$) and $\{\Theta, \mathbf{D}\}$ using the training dataset (incomplete).

For fixed $\{\Theta, \mathbf{D}\}$, the optimization with respect to $\mathbf{s}_i$ is a non-smooth separable minimization sub-problem, which was extensively studied in the literature [31], [32]. In this sub-problem, the objective function is written as the sum of $F(\Theta, \mathbf{D}, \mathbf{s}_i)$ and a non-smooth separable function $G(\mathbf{s}_i)$, for which highly specialized, efficient and provable convergent

solvers, namely the Coordinate Gradient Descent (CGD), already exists. However, the following key differences in our setting makes it not suitable for the CGD approach: first, our function $F(\Theta, \mathbf{D}, \mathbf{s}_i)$ involves evaluation of a multi-layer NN classifier, which can be non-smooth due to involved activation functions like ReLU or others; second, and more importantly, the computation of its second derivatives (Hessian) becomes prohibitive. Therefore, we choose a simpler and standard first order (stochastic sub-gradient based) search of local minima with back-propagation. We take the strategy similar to the heuristics used in [33]. To update $\mathbf{s}_i$, we need to subtract $\sigma_{\mathbf{s}} \frac{\partial J}{\partial \mathbf{s}_i}(j)$ from each coordinate j provided that we do not cross zero in the process in order to avoid escaping from a region where $G(\mathbf{s}_i)$ is differentiable. In such a case, we let the new value of $\mathbf{s}_i(j)$ be exactly zero. More specifically, we define $\Delta_i(j) = -\sigma_{\mathbf{s}} \frac{\partial J}{\partial \mathbf{s}_i}(j)$ and, if $\mathbf{s}_i(j)[\mathbf{s}_i(j) + \Delta_i(j)] < 0$ (zero crossing condition), we re-define $\Delta_i(j) = -\mathbf{s}_i(j)$; finally we update $\mathbf{s}_i \leftarrow \mathbf{s}_i + \Delta_i$. It is noted that, once a coefficient $\mathbf{s}_i(j)$ reaches zero at a coordinate j, it becomes fixed, in other words, sparsity of solution $\mathbf{s}_i$ is monotonically increasing with iterations.

When $\mathbf{s}_i$ is fixed, our problem is reduced to minimize $F(\Theta, \mathbf{D}, \mathbf{s}_i)$ with respect to Θ and $\mathbf{D}$, which is easily done by standard first order (stochastic gradient based) search of local minima. The algorithm proposed for the training phase is presented as Algorithm 1.

In addition, for the testing phase, if the test dataset is incomplete, we need to find first the sparsest representation for the given observations, compute the reconstructions $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i$ and then apply the previously learned classifier to them as presented in Supp. material, Algorithm 2.

Algorithm 1 : Simultaneous classification and coding

Require: $\{\mathbf{x}_i^o, y_i\}$, $i = 1, 2, \ldots, I$, hyper-parameters λ_1 and λ_2, N_{iter} and update rates σ_Θ, $\sigma_{\mathbf{D}}$ and $\sigma_{\mathbf{s}}$
Ensure: Weights Θ and reconstructions $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$
1: `Randomly initialize` $\Theta, \mathbf{D}, \mathbf{s}_i, \forall i$
2: **for** $n \leq N_{iter}$ **do**
3: `Fix` $\mathbf{s}_i$, `update` Θ `and` $\mathbf{D}$:
4: $\Theta = \Theta - \sigma_\Theta \frac{\partial J}{\partial \Theta}$
5: $\mathbf{D} = \mathbf{D} - \sigma_{\mathbf{D}} \frac{\partial J}{\partial \mathbf{D}}$
6: `Normalize columns of matrix` $\mathbf{D}$
7: `Fix` Θ and $\mathbf{D}$, `update` $\mathbf{s}_i$, $\forall i$:
8: $\Delta_i = -\sigma_{\mathbf{s}} \frac{\partial J}{\partial \mathbf{s}_i}$, $\forall i$
9: **if** $\mathbf{s}_i(j)[\mathbf{s}_i(j) + \Delta_i(j)] < 0$ **then**
10: $\Delta_i(j) = -\mathbf{s}_i(j), \forall (i, j)$;
11: **end if**
12: $\mathbf{s}_i = \mathbf{s}_i + \Delta_i$, $\forall i$
13: **end for**
14: **return** $\Theta, \mathbf{D}, \mathbf{s}_i, \hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$

B. Theoretical analysis

Here, we investigate about conditions under which a perfect classifier of the complete data can be obtained from incomplete data samples.

1) Logistic regression: Let us first consider a logistic regression classifier [34] where the set of parameters $\Theta = \{\mathbf{w}, b\}$ are a vector $\mathbf{w} \in \mathbb{R}^N$ and a scalar b (bias). A perfect classifier exists if there is a hyperplane that separates both classes, *i.e.*, for each data vector $\mathbf{x}_i$: $f(\mathbf{x}_i) = \langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0$ if $y_i = 1$, and $f(\mathbf{x}_i) \leq 0$ if $y_i = 0$. We consider data samples admitting a K-sparse representations $\mathbf{x}_i = \mathbf{D}\mathbf{s}_i$ with dictionary $\mathbf{D} \in \mathbb{R}^{N \times P}$ having unit-norm columns. We also assume an arbitrary pattern of missing features $\mathcal{M}_i$ such that, data samples and dictionary are partitioned as $\{\mathbf{x}_i^m, \mathbf{x}_i^o\}$ and $\{\mathbf{D}_i^m, \mathbf{D}_i^o\}$, respectively. The following lemma identifies a sufficient condition under which, if we are able to train a classifier on incomplete observations such that the reconstructed data points are well separated by a hyperplane, then the same classifier correctly separates the original (unobserved) data vectors.

Lemma II.1 (Sufficient condition type I)**.** *Suppose that we have obtained an alternative dictionary $\mathbf{D}' \neq \mathbf{D} \in \mathbb{R}^{N \times P}$ such that, for the incomplete observations $\mathbf{x}_i^o \in \mathbb{R}^{M_i}$, the K-sparse representation solutions are non-unique, i.e. $\exists \mathbf{s}_i, \mathbf{s}_i' \in \Sigma_K^P$ such that $\mathbf{x}_i^o = \mathbf{D}_i^o \mathbf{s}_i = \mathbf{D}_i'^o \mathbf{s}_i'$, where $\mathbf{s}_i \in \mathbb{R}^P$ are the vectors of coefficients of the true data and $\mathbf{s}_i'$ provides reconstructions $\hat{\mathbf{x}}_i = \mathbf{D}'\mathbf{s}_i'$. If a perfect classifier $\{\mathbf{w}, b\}$ of the reconstructions $\hat{\mathbf{x}}_i$ exists s.t. $|f(\hat{\mathbf{x}}_i)| > \epsilon_i > 0$ and*

$$\epsilon_i > |\langle \mathbf{w}_i^m, \mathbf{e}_i^m \rangle| \tag{4}$$

with $\mathbf{e}_i^m = \mathbf{x}_i^m - \hat{\mathbf{x}}_i^m$, then the full data vectors $\mathbf{x}_i$ are also perfectly separated with this classifier, in other words: $f(\mathbf{x}_i) = \langle \mathbf{w}_i, \mathbf{x}_i \rangle + b > 0$ (≤ 0) if $y_i = 1$ ($y_i = 0$).

Proof. By using the missing/observed partition and omitting the sample index i, we can write: $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}^o, \mathbf{x}^o \rangle + \langle \mathbf{w}^m, \mathbf{x}^m \rangle + b$. If we add and subtract the term $\langle \mathbf{w}^m, \hat{\mathbf{x}}^m \rangle$ on the right left hand, arrange terms and use the fact that $\hat{\mathbf{x}}^o = \mathbf{D}'^o \mathbf{s}' = \mathbf{x}^o$, we get:

$$f(\mathbf{x}) = f(\hat{\mathbf{x}}) + \langle \mathbf{w}^m, \mathbf{e}^m \rangle. \tag{5}$$

Since we assumed that $f(\hat{\mathbf{x}}) > \epsilon > 0$ (for $y_i = 1$) and $|\langle \mathbf{w}^m, \mathbf{e}^m \rangle| < \epsilon$, it implies that $f(\mathbf{x}) > 0$. □

Basically, condition (4) means requiring that reconstruction vector $\hat{\mathbf{x}}$ has a distance ϵ to the separating hyperplane larger than the absolute dot product between $\mathbf{w}$ and the residual $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$, which of course is true when the reconstruction is accurate, *i.e.* $\mathbf{x} \approx \hat{\mathbf{x}}$. However, in practice, reconstructions are not accurate so we are interested in conditions under which Lemma II.1 can still holds. Below, we derive a more restrictive but useful sufficient condition:

Proposition II.1 (Sufficient condition type II)**.** *Under the same hypothesis of Lemma II.1, the following condition is enough to guarantee a proper classifier trained on incomplete data:*

$$\epsilon > |\langle \mathbf{w}^m, \mathbf{x}^m \rangle| + |\langle \mathbf{w}^m, \hat{\mathbf{x}}^m \rangle|. \tag{6}$$

Proof. By using the fact that $|\langle \mathbf{w}^m, \mathbf{e}^m \rangle| = |\langle \mathbf{w}^m, \mathbf{x}^m - \hat{\mathbf{x}}^m \rangle| \leq |\langle \mathbf{w}^m, \mathbf{x}^m \rangle| + |\langle \mathbf{w}^m, \hat{\mathbf{x}}^m \rangle|$, and applying Lemma II.1 the proof is completed. □

We highlight that, in our experiments, we were able to verify that Sufficient Condition type II is met in practice (see section III, Fig. 3).

In Supp. material section V-B, we derive an additional sufficient condition based on the Restricted Isometry Property (RIP) of the dictionary $\mathbf{D}$ and sparsity level K, showing that sufficient condition (6) is easier to hold for datasets admitting highly sparse representations on dictionaries as close to orthogonal ones as possible.

2) Multilayer-perceptron: Lemma II.1 can be straightforwardly generalized to multilayer-perceptron NNs where, if a softmax function is used at the output of the last layer then, as before, the prediction is based on the sign of the linear function:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}^{(L)} \rangle + b, \text{ with } \mathbf{x}^{(l)} = h\left(\mathbf{W}_l^T \mathbf{x}^{(l-1)} + \mathbf{B}_l\right), \quad (7)$$

$l = 1, 2, \ldots, L$, where $L+1$ is the total number of layers, N_l is the number of neurons in layer l, $\mathbf{w} \in \mathbb{R}^{N_{L+1}}$ contains the weights in the last layer, $h(\cdot)$ is an activation function, *e.g.* ReLU, $\mathbf{W}_l \in \mathbb{R}^{N_{l-1} \times N_l}$ and $\mathbf{B}_l \in \mathbb{R}_l^N$ contain the weights and biases associated to neurons at layer l; and $\mathbf{x}^{(0)} = \mathbf{x}$ is the input data vector. In this case, the first layer matrix $\mathbf{W}_1 \in \mathbb{R}^{N \times N_1}$ can be partitioned into submatrices $\mathbf{W}_{1i}^o \in \mathbb{R}^{M \times N_1}$ and $\mathbf{W}_{1i}^m \in \mathbb{R}^{(N-M) \times N_1}$ according to the observed and missing input features, respectively.

Proposition II.2. *Under the same conditions of Lemma II.1, if a NN-classifier* $\{\mathbf{W}_l, \mathbf{B}_l (l = 1, 2, \ldots L), \mathbf{w}, b, h = ReLU\}$ *of the reconstruction* $\hat{\mathbf{x}}_i$ *exists such that*

$$\epsilon_i > A \max_j |\langle \mathbf{W}_{1i}^m(:, j), \mathbf{e}_i^m \rangle|, \quad (8)$$

where $A = \|\mathbf{w}\| \prod_{l=2}^{L} \|\mathbf{W}_l\|_2$ *and* $\mathbf{e}_i^m = \mathbf{x}_i^m - \hat{\mathbf{x}}_i^m$*, then the full data vector* $\mathbf{x}_i$ *is also perfectly separated, in other words:* $f(\mathbf{x}_i) > 0$ *(*< 0*) if* $y_i = 1$ *(*$y_i = 0$*).*

In the proof of Lemma II.1, we were interested in finding a bound of the output error when the input $\mathbf{x}$ of a classifier is perturbed, *i.e.* we found conditions such that $|f(\mathbf{x}) - f(\mathbf{x}+\delta)| < \epsilon$. By generalizing the classifier to the case of a multilayer perceptron we can derive the proof as follows:

Proof. Given a perturbation $\delta^{(l-1)} \in \mathbb{R}^{N_l}$ at the input of layer $l-1$, *i.e.* $\hat{\mathbf{x}}^{(l-1)} = \mathbf{x}^{(l-1)} + \delta^{(l-1)}$, it is propagated to the output of layer l. By writing the error at the output we obtain:

$$\delta^{(l)} = h\big(\mathbf{W}_l^T \mathbf{x}^{(l-1)} + \mathbf{B}_l + \mathbf{W}_l^T \delta^{(l-1)}\big) - h\big(\mathbf{W}_l^T \mathbf{x}^{(l-1)} + \mathbf{B}_l\big), \quad (9)$$

and, by using the sub-additivity of ReLU function $h(\cdot)$, *i.e.* $h(a+b) \leq h(a) + h(b)$, we derive the following entry-wise inequality:

$$\delta^{(l)} \leq h\left(\mathbf{W}_l^T \delta^{(l-1)}\right), \quad (10)$$

and, by considering the property of ReLU activation function $\|h(\mathbf{x})\| \leq \|\mathbf{x}\|$, it turns out:

$$\|\delta^{(l)}\| \leq \|\mathbf{W}_l^T \delta^{(l-1)}\|, \quad (11)$$

Since the last layer of the NN is a linear classifier as in the case of Lemma II.1, we can ask that $\langle \mathbf{w}, \delta^{(L)} \rangle < \epsilon$. Thus, by recursively using equation (11), we write

$$\langle \mathbf{w}, \delta^{(L)} \rangle \leq \|\mathbf{w}\| \|\delta^{(L)}\| \leq \|\mathbf{w}\| \|\mathbf{W}_L\|_2 \|\mathbf{W}_{l-1}\|_2 \cdots \|\mathbf{W}_2\|_2 \|\delta^{(1)}\|. \quad (12)$$

By defining $A = \|\mathbf{w}\| \prod_{l=2}^{L} \|\mathbf{W}_l\|_2$, evaluating equation (11) with $l = 1$ and taking into account that perturbation at the input of first layer is $\delta^{(0)} = \mathbf{e}$ with $\mathbf{e}^o = \mathbf{0}$, we arrive at:

$$\langle \mathbf{w}, \delta^{(L)} \rangle \leq A \|\mathbf{W}_1^{mT} \mathbf{e}^m\| \leq A \max_j |\langle \mathbf{W}_1^m(:, j), \mathbf{e}^m \rangle| < \epsilon, \quad (13)$$

which completes the proof. □

It is interesting to note that $A = 1$ is attained when unit-norm filters (columns of $\mathbf{W}_l$) are orthogonal, which can be imposed by using orthogonality regularization [35].

III. Experimental results

We implemented all the algorithms in Pytorch 1.0.0 on a single GPU. Implementation details are reported in Supplemental material, sections V-C1 and V-C2. The code is available at [1].

Synthetic datasets: We synthetically generated $I = 11,000$ ($10,000$ training + $1,000$ test) K-sparse data vectors $\mathbf{x}_i \in \mathbb{R}^{100}$ using a dictionary $\mathbf{D} \in \mathbb{R}^{100 \times 200}$ obtained from a Gaussian distribution with normalized atoms, *i.e.* $\|\mathbf{D}(:, j)\| = 1, \forall j$. A random hyperplane $\{\mathbf{w}, b\}$ with $\mathbf{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$ was randomly chosen dividing data vectors into two classes according to the sign of the expression $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$, which defined the label y_i. We also controlled the degree of separation between classes by discarding all data vectors with distances to the hyperplane lower than a pre-specified threshold, *i.e.* $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| < d$. We used $n = 10$ repetitions of each experiment with different masks and input data in order to compute statistics.

We applied our simultaneous method (**Simult.**) with hyper-parameters λ_1 and λ_2 in the cost function (II) tuned via cross-validation to train a logistic regression classifier on incomplete datasets with randomly distributed missing features. Then, we computed the classification accuracy on the complete test dataset and compared the results against the following standard sequential methods:

Sequential Sparsity based (Seq. Sp.): reconstructions are obtained by finding the sparsest representation compatible with the observations solving a LASSO problem. We used Algorithm 3 as shown in the Supp. material;

Zero Fill (ZF): missing features are filled with zeros, which is equivalent to ignore unknown values;

Mean Unsupervised (MU): missing features are filled with the mean computed on the available values;

Mean Supervised (MS): as in the previous case but the mean is computed on the same class vectors only;

K-Nearest Neighbor (KNN): as in the previous case but the mean is computed on the K-nearest neighbors of the same class vectors only.

To compare the performance of classifiers, we computed the mean accuracy $\pm$ standard error of the mean (s.e.m.), with $n = 10$, on complete test datasets using all the methods for two levels of separation between classes ($d = 0.0, 0.2$), two levels of sparsity ($K = 4, 32$) and missing features in the

[1] https://github.com/ccaiafa/SimultCodClass

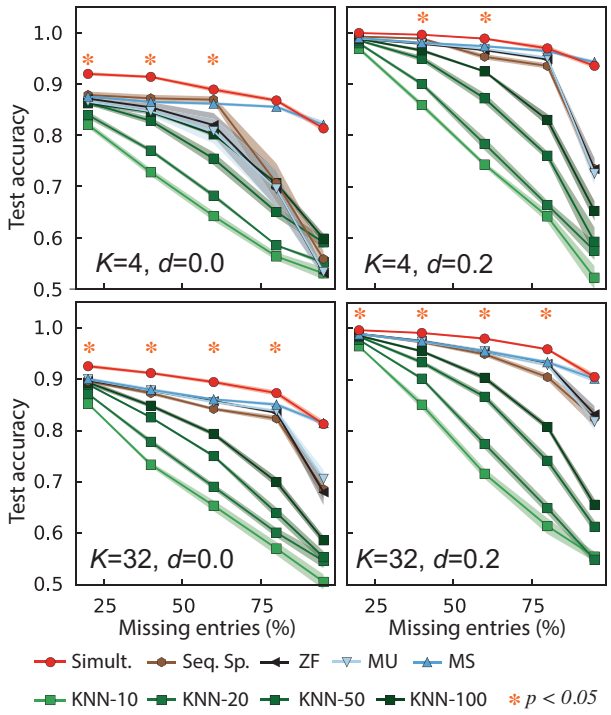

Fig. 2: Experimental results on synthetic dataset with random masks using our algorithm (red) and compared to various sequential methods. Test accuracy (mean $\pm$ s.e.m with $n = 10$) is shown as a function of the percentage of missing features for separation of classes $d = 0.0, 0.2$ and levels of sparsity $K = 4, 32$. Statistical significance for the difference between Simult. and MS is shown ($p < 0.05$).

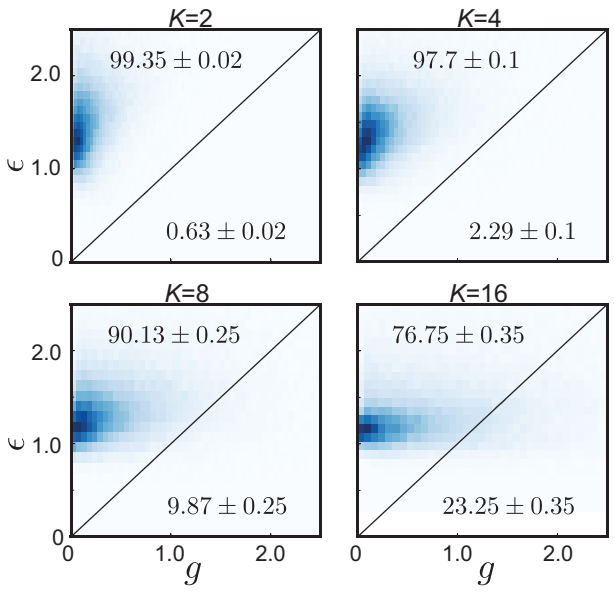

Fig. 3: Verification of the sufficient condition (6) for various levels of sparsity K: 2D-histogram of ϵ versus $g = |\langle \mathbf{w}^m, \mathbf{x}^m \rangle| + |\langle \mathbf{w}^m, \hat{\mathbf{x}}^m \rangle|$. Mean + s.e.m ($n = 10$) percentage of correctly classified data samples are shown for $\epsilon > g$ and $\epsilon < g$.

training dataset ranging from 25% to 95% as shown in Fig. 2. Our results show that the simultaneous algorithm clearly outperforms all the sequential methods. A t-test was performed to evaluate the statistical significance with $p < 0.05$ of the difference between our algorithm and MS. It is interesting to note that, when classes has some degree of separation ($d = 0.2$), using the simple MS method, can give good results but not better than our algorithm.

In the second experiment, we generated $I = 10,000$ K-sparse data vectors $\mathbf{x}_i \in \mathbb{R}^{100}$ using $\mathbf{D} \in \mathbb{R}^{100 \times 100}$ and we evaluated the sufficient condition of equation (6) on $n = 10$ repetitions of the experiment with 95% missing features and separation $d = 0.0$. Fig. 3 clearly shows that the sufficient condition is mostly met in practice, especially for highly sparse representations of input data (small K). This means that in practice it is not necessary to accurately reconstruct the input vectors, it is enough to capture the intrinsic characteristics of the classes such that the distances of reconstructions to the separating hyperplane satisfy the sufficient condition (3).

Benchmark datasets: We also considered three popular computer vision datasets: MNIST [36] and Fashion [37] consisting of 70,000 images (60,000 train + 10,000 test) each; and CIFAR10 [38] having 60,000 images (50,000 train + 10,000 test). MNIST/Fashion datasets contains 28×28 gray scale images while CIFAR10 dataset is built upon $32 \times 32 \times 3$ color images of different objects. The corresponding data sample size is $N = 28 \times 28 = 784$ for MNIST/Fashion and $N = 32 \times 32 \times 3 = 3,072$ for CIFAR10. We considered a dictionary of size 784×784 (MNIST/Fashion) and $1,024 \times 1,024$ (CIFAR10) and applied our simultaneous algorithm to learn the classifier on incomplete data using uniform random missing masks with several levels of missing data (25%, 50% and 75%) and 50% for random partial occlusions with MNIST/Fashion.

We used a logistic regression classifier (single layer NN) and a 4-layer convolutional neural network [39] (CNN4) for the MNIST/Fashion dataset using batch normalization (BN) [40] in the Fashion dataset. For CIFAR10 dataset, an 18-layer residual neural network, Resnet-18 [41] was implemented. We did not use any data augmentation strategy. The hyper-parameters λ_1 and λ_2 in cost function (II) were adjusted by cross-validation through a grid-search, as shown in Supp. material (Table IV and Fig. 5). We compared our proposed algorithm with the following standard sequential methods: ZF, MS, KNN-10, KNN-20, KNN-50 and KNN-100; and against the recently proposed method from [15], referred here as NN-GMM, which uses the same NN classifier as in our method and models missing features through GMM[2]. We trained the classifiers on incomplete data with random masks and tested them on complete data for MNIST and CIFAR10 datasets. The obtained mean Test Accuracy $\pm$ s.e.m ($n = 10$) are reported in Table I. It is noted that NN-GMM provided good results with MNIST dataset compared to sequential methods, however, our simultaneous method outperformed all the methods. Interestingly, NN-GMM performed worst than any other method with CIFAR10 dataset. It seems that NN-GMM is not robust to large amount of missing data because,

[2]https://github.com/lstruski/Processing-of-missing-data-by-neural-networks

when we reduced the missing entries to 10%, the test accuracy sensibly increased to 52.57%. Additionally, our method showed to have little variability (small s.e.m) compared to the second best method (NN-GMM for MNIST and ZF for CIFAR10).

In Table II, test accuracies obtained when the learned model is applied to incomplete and complete test data, are shown. The right-most column shows the baseline results obtained by training the model on complete datasets using a CNN4 [39] and a Resnet-18 [41], whose implementations can be found at [3] and [4]. It is interesting to note that for the logistic regression classifier, we obtained better results when training with incomplete data rather than using complete data. Also, it is highlighted that training on incomplete data with 50% or fewer random missing features, provides similar test accuracy as training on complete data for MNIST dataset. This could be explained by noting two facts: (1) random missing features is similar to applying dropout, with the exception that missing data do not change during training; and (2) our model has more parameters (Dictionary + sparse coefficients + Linear layer) compared to the baseline logistic regression classifier. To provide a deeper understanding of this effect, we ran the baseline with Dropout at the input and we obtained: 91.95%, 91.97% and 92.01% for $p = 0.0$, 0.1 and 0.25, respectively, which shows that the improvement we obtained with our method is not solely caused by a dropout alike behavior.

In Fig. 4, we present some randomly selected visual examples comparing the original images in the MNIST/Fashion test dataset, their observations using random masks and partial occlusions, and the reconstructions using the dictionary learned from the incomplete training data. It is clear that, despite the reconstructions may be not very similar to the original images (see "5" digit example), they clearly own the properties of the class to which they belong to. Additional examples are provided in Supp. material (Figs. 6 and 7).

IV. Discussion

It is well known that sparse coding has the ability to accurately model complex distributions of data, such as natural signals (images, audio, EEG, etc). In this work, we demonstrated that assuming a sparse representation for input data allows for the successful training of a general NN when incomplete data is given outperforming traditional sequential approaches and other start-of-the-art methods. It is highlighted that our method can be used with potentially any deep NN architecture, thus relying on their extraordinary capability to accommodate complex decision boundaries as usually needed in modern machine learning.

Our method overcomes well known issues of previous approaches: (1) compared to imputation methods, our algorithm successfully incorporates the labelling information into the modeling of missing features; (2) sparse coding allows for a simple way to train dictionaries through linear methods such as stochastic gradient descent with back-propagation

[3]https://github.com/pytorch/examples/tree/master/mnist

[4]https://github.com/kuangliu/pytorch-cifar

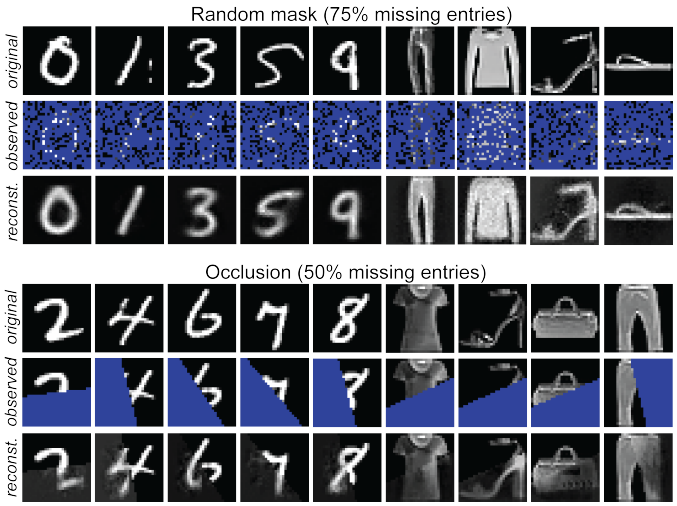

Fig. 4: Original (top), observed (middle) and reconstructed (bottom) MNIST and Fashion test images.

compared to the very expensive EM estimators for GMM used in probabilistic generative models, or SVD based algorithms for matrix rank minimization in matrix completion; (3) sparse coding can be more accurate modeling missing values in natural signals compared to GMM, especially for high dimensional data where GMM may require a huge number of parameters making it computationally prohibitive.

We analyzed the limitations of the classical imputation approach and demonstrated through experiments with synthetical and real-world datasets that our simultaneous algorithm always outperforms them for various cases such as LASSO, zero-filling, supervised/unsupervised mean and KNN based methods as well as the *state-of-the-art* method based on NNs and GGM recently proposed in [15]. Nevertheless, our experimental results on synthetic and real-world dataset showed that, even though we only constrained dictionaries to have unit-norm columns but not enforcing any other kind of constraint like maximum coherence, the obtained results seem to be satisfactory enough. However, further analysis on the required properties of dictionaries could provide deeper insights and alternative ways to improve the algorithm, which we aim to address in a future work.

While current simple sub-gradient based optimization approach provided satisfactory results in terms of performance, it is remarked that observed convergence is slow requiring a thousand of iterations sometimes. We believe, it could be improved by trying to incorporate some second-order derivatives information for computing the updates. Although, full Hessian computation becomes prohibitive with multi-layer NNs a diagonal approximation approach could be explored. Also, a rigorous convergence analysis in the line of the analysis in [31], [32] and taking special properties of multi-layer NN classifier functions can be conducted in a future work.

Finally, we provided theoretical insights of the problem by providing sufficient conditions under which, if it is possible to train a classifier on incomplete observations so that its reconstructions are well separated by a hyperplane, then the

TABLE I: Test accuracy (mean $\pm$ s.e.m with $n = 10$) of various methods trained on incomplete data and tested on complete ones for MNIST and CIFAR10.

MNIST (CNN4)								
Miss.	ZF	MS	KNN10	KNN20	KNN50	KNN100	NN-GMM	Simult.
75%	84.86 ± 0.02	83.79 ± 0.01	88.16 ± 0.01	87.94 ± 0.01	87.03 ± 0.002	86.52 ± 0.01	96.36 ± 0.12	$\mathbf{98.09 \pm 0.04}$
50%	90.13 ± 0.06	88.55 ± 0.01	91.36 ± 0.02	91.11 ± 0.02	90.87 ± 0.01	90.82 ± 0.01	97.57 ± 0.37	$\mathbf{98.23 \pm 0.10}$
CIFAR10 (Resnet18)								
Miss.	ZF	MS	KNN10	KNN20	KNN50	KNN100	NN-GMM	Simult.
75%	32.22 ± 2.09	21.30 ± 0.40	22.84 ± 0.87	25.67 ± 0.80	26.52 ± 0.70	26.01 ± 0.52	12.10 ± 0.61	$\mathbf{54.81 \pm 0.47}$
50%	46.37 ± 1.93	17.90 ± 0.94	30.94 ± 0.54	29.68 ± 0.46	30.01 ± 0.51	26.23 ± 1.01	14.02 ± 0.75	$\mathbf{62.50 \pm 0.95}$

TABLE II: Test Accuracies obtained with our method on MNIST, Fashion and CIFAR10 datasets training with incomplete data and testing on incomplete/complete data. Baseline results obtained by training the models on complete data are shown for reference in the right-most column.

Dataset	**Classifier**	**Random missing features %Train / %Test**						**Occlusion %Train / %Test**		**Baseline %Train / %Test**
		75/75	**50/50**	**25/25**	**75/0**	**50/0**	**25/0**	**50/50**	**50/0**	**0/0**
MNIST	**Log. Reg.**	90.45	93.68	94.14	91.94	93.44	94.43	-	-	91.95
	CNN4	94.62	98.34	98.94	98.09	98.23	98.95	88.55	91.37	98.95
Fashion	**CNN4+BN**	83.71	86.09	86.38	86.39	87.11	87.04	81.73	82.47	90.76
CIFAR10	**Resnet18**	53.82	61.08	63.73	54.81	62.50	63.87	-	-	80.13

same classifier also correctly separates the original (unobserved) data samples.

Acknowledgments: We are thankful for the RAIDEN computing system and its support team at RIKEN AIP, Tokyo. This work was supported by the JSPS KAKENHI (Grant No. 20H04249, 20H04208).

REFERENCES

[1] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin Classification of Data with Absent Features." *Journal of Machine Learning Research (JMLR)*, vol. 9, pp. 1–21, 2008.

[2] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*. John Wiley & Sons, Aug. 2014.

[3] J. Josse, N. Prost, E. Scornet, and G. Varoquaux, "On the consistency of supervised learning with missing values," *arXiv.org*, p. arXiv:1902.06931, Feb. 2019.

[4] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2009.

[5] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach." in *NIPS*, 1993.

[6] D. Williams, X. Liao, Y. Xue, and L. Carin, "Incomplete-data classification using logistic regression." *ICML*, 2005.

[7] C. Bhattacharyya, P. K. Shivaswamy, and A. J. Smola, "A Second Order Cone programming Formulation for Classifying Missing Data." in *NIPS*, 2004.

[8] X. Liao, H. Li, and L. Carin, "Quadratically gated mixture of experts for incomplete data classification." in *ICML*, 2007.

[9] U. Dick, P. Haider, and T. Scheffer, "Learning from incomplete data with infinite imputations." in *ICML*, 2008.

[10] T. I. Lin, J. C. Lee, and H. J. Ho, "On fast supervised learning for normal mixture models with missing information," *Pattern Recognition*, vol. 39, no. 6, pp. 1177–1187, 2006.

[11] O. Delalleau, A. C. Courville, and Y. Bengio, "Efficient EM Training of Gaussian Mixtures with Missing Data," *arXiv*, vol. cs.LG, p. 1209.0521, 2012.

[12] A. B. Goldberg, X. Zhu, B. Recht, J.-M. Xu, and R. D. Nowak, "Transduction with Matrix Completion - Three Birds with One Stone." in *NIPS*, 2010.

[13] E. Hazan, R. Livni, and Y. Mansour, "Classification with Low Rank and Missing Data." in *ICML*, 2015.

[14] S.-J. Huang, M. Xu, M.-K. Xie, M. Sugiyama, G. Niu, and S. Chen, "Active Feature Acquisition with Supervised Matrix Completion," *arXiv*, vol. cs.LG, p. 1802.05380, 2018.

[15] M. Smieja, L. Struski, J. Tabor, B. Zielinski, and P. Spurek, "Processing of missing data by neural networks." in *NeurIPS*, 2018.

[16] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms." in *NIPS*, 2006.

[17] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding." in *ICML*, 2009.

[18] K. Gregor and Y. LeCun, "Learning Fast Approximations of Sparse Coding." in *ICML*, 2010.

[19] S. Mallat, *A Wavelet Tour of Signal Processing*, ser. The Sparse Way. Academic Press, 2009.

[20] M. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, *Introduction to compressed sensing*, ser. Theory and Applications. Cambridge: Cambridge University Press, 2012.

[21] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[22] B. Olshausen and D. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[23] K. Huang and S. Aviyente, "Sparse Representation for Signal Classification." in *NIPS*, 2006.

[24] J. Mairal, J. Ponce, G. Sapiro, A. Z. A. i. neural, and 2009, "Supervised dictionary learning," in *NIPS*, 2008.

[25] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features." in *CVPR*, 2010.

[26] P. Sprechmann and G. Sapiro, "Dictionary learning and sparse coding for unsupervised clustering." in *ICASSP*, 2010.

[27] I. Tosic and P. Frossard, "Dictionary Learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.

[28] R. Jenatton, J. Mairal, G. Obozinski, and F. R. Bach, "Proximal Methods for Sparse Hierarchical Dictionary Learning." in *ICML*, 2010.

[29] J. Weed, "Approximately Certifying the Restricted Isometry Property is Hard," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5488–5497, 2017.

[30] E. J. Candès and T. Tao, "Decoding by Linear Programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[31] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, no. 1-2, pp. 387–423, 2007.

[32] E. T. Hale, W. Yin, and Y. Zhang, "Fixed-Point Continuation for ℓ_1-Minimization: Methodology and Convergence," *SIAM J. OPTIM.*, vol. 19, no. 3, pp. 1107–1130, 2008.

[33] S. Shalev-Shwartz and A. Tewari, "Stochastic Methods for l1-regularized Loss Minimization." *Journal of Machine Learning Research (JMLR)*, 2011.

[34] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning - data mining, inference, and prediction, 2nd Edition.*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009.

[35] N. Bansal, X. Chen, and Z. Wang, "Can We Gain More from Orthogonality Regularizations in Training Deep CNNs?" in *NeurIPS*, 2018.

[36] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network." in *NIPS*, 1989.

[37] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv*, vol. cs.LG, p. arXiv:1708.07747, 2017.

[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," Ph.D. dissertation, Toronto University, Toronto, 2009.
[39] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning." in *Shape, Contour and Grouping in Computer Vision*, 1999.
[40] S. Ioffe and C. Szegedy, "Batch Normalization - Accelerating Deep Network Training by Reducing Internal Covariate Shift." in *ICML*, 2015.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*. IEEE, 2016, pp. 770–778.
[42] V. Naumova and K. Schnass, "Dictionary learning from incomplete data for efficient image restoration." *EUSIPCO*, 2017.
[43] J. Mairal, M. Elad, and G. Sapiro, "Sparse Representation for Color Image Restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, Jan. 2008.

V. Supplemental material

A. Additional pseudocodes

Here, additional pseudocode of the algorithms discussed in the paper are provided. Once the classifier is trained by using Algorithm 1, we are able to apply it to incomplete test data by using Algorithm 2, where for fixed Θ and $\mathbf{D}$, we need to find the corresponding sparse coefficients $\mathbf{s}_i$, compute the full data vector estimations and, finally, apply the classifier.

A sparsity-based sequential method is presented in Algorithm 3 (sequential approach), which consists on learning first the optimal dictionary $\mathbf{D}$ and sparse coefficients $\mathbf{s}_i$ compatible with the incomplete observations (dictionary learning and coding phase), followed by the training phase, where the classifier weights are tuned in order to minimize the classification error of the reconstructed input data vectors $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i$. It is noted that for the imputation stage (lines 2-12) other and more specialized dictionary learning algorithms with missing data can be applied, such as the ones proposed in [42] for high-dimensional data or [43] for color image data.

Algorithm 2 : Testing on incomplete data

Require: Incomplete data vectors $\{\mathbf{x}_i^o\}$, $i = 1, 2, \ldots, I$, classifier parameters Θ, dictionary $\mathbf{D}$, hyper-parameters λ_1 and λ_2, number of iterations N_{iter} and update rate $\sigma_{\mathbf{s}}$
Ensure: $\hat{y}_i$ and reconstructions $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$
1: **Sparse coding stage:** for fixed dictionary $\mathbf{D}$ find sparse representations of observations $\mathbf{x}_i^o$
2: `Initialize` $\mathbf{s}_i, \forall i$ `randomly`
3: **for** $n \leq N_{iter}$ **do**
4: $\quad \Delta_i = -\sigma_{\mathbf{s}}\big[\lambda_1 \frac{\partial J_1}{\partial \mathbf{s}_i} + \lambda_2 \frac{\partial J_2}{\partial \mathbf{s}_i}\big]$, $\forall i$
5: $\quad$ **if** $\mathbf{s}_i(j)[\mathbf{s}_i(j) + \Delta_i(j)] < 0$ **then**
6: $\quad\quad \Delta_i(j) = -\mathbf{s}_i(j)$; avoid zero crossing
7: $\quad$ **end if**
8: $\quad \mathbf{s}_i = \mathbf{s}_i + \Delta_i$, $\forall i$
9: **end for**
10: $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$; Compute reconstructions
11: **Classification stage:** apply classifier to reconstructions $\hat{\mathbf{x}}_i$
12: $\hat{y}_i = \arg\max_y (p_{\Theta}^y(\hat{\mathbf{x}}_i))$
13: **return** $\Theta, \hat{y}_i, \mathbf{s}_i, \hat{\mathbf{x}}_i, \forall i$

Algorithm 3 : Sequential sparsity based approach

Require: Incomplete data vectors and their labels $\{\mathbf{x}_i^o, y_i\}$, $i = 1, 2, \ldots, I$, hyper-parameters λ_1 and λ_2, number of iterations N_{iter} and update rate σ_{Θ}, $\sigma_{\mathbf{D}}$ and $\sigma_{\mathbf{s}}$
Ensure: Classifier weights Θ and reconstructions $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$
1: `Randomly initialize` $\mathbf{D}, \mathbf{s}_i, \forall i$
2: **Imputation stage: learning of $\mathbf{D}$ and $\mathbf{s}_i$**
3: **for** $n \leq N_{iter}$ **do**
4: $\quad \mathbf{D} = \mathbf{D} - \sigma_{\mathbf{D}} \frac{\partial J_1}{\partial \mathbf{D}}$
5: $\quad$ Normalize columns of matrix $\mathbf{D}$
6: $\quad \Delta_i = -\sigma_{\mathbf{s}}\big[\lambda_1 \frac{\partial J_1}{\partial \mathbf{s}_i} + \lambda_2 \frac{\partial J_2}{\partial \mathbf{s}_i}\big]$, $\forall i$
7: $\quad$ **if** $\mathbf{s}_i(j)[\mathbf{s}_i(j) + \Delta_i(j)] < 0$ **then**
8: $\quad\quad \Delta_i(j) = -\mathbf{s}_i(j)$; avoid zero crossing
9: $\quad$ **end if**
10: $\quad \mathbf{s}_i = \mathbf{s}_i + \Delta_i$, $\forall i$
11: **end for**
12: $\hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$; Compute reconstructions
13: **Training stage: update Θ**
14: **for** $n \leq N$ **do**
15: $\quad \Theta = \Theta - \sigma_{\Theta} \frac{\partial J_0}{\partial \Theta}$;
16: **end for**
17: **return** $\Theta, \mathbf{D}, \mathbf{s}_i, \hat{\mathbf{x}}_i = \mathbf{D}\mathbf{s}_i, \forall i$

B. A condition based on RIP and sparsity

The Restricted Isometry Property (RIP): An overcomplete dictionary $\mathbf{D}$ satisfies the RIP of order K if there exists $\delta_K \in [0, 1)$ s.t.

$$(1 - \delta_K)\|\mathbf{s}\|_2^2 \leq \|\mathbf{D}\mathbf{s}\|_2^2 \leq (1 + \delta_K)\|\mathbf{s}\|_2^2, \tag{14}$$

holds for all $\mathbf{s} \in \Sigma_K^P$. RIP was introduced in [30] and characterizes matrices which are nearly orthonormal when operating on sparse vectors.

In the following theorem, we show that, by imposing conditions on the sparsity level of the representation and the RIP constant of a sub-matrix of the dictionary, we can guarantee to meet the sufficient condition (6).

Theorem V.1. *Given a dataset $\{\mathbf{x}_i, y_i\}$, $i = 1, 2, \ldots, I$ with normalized data vectors ($\|\mathbf{x}_i\| \leq 1$) admitting a K-sparse representation over a dictionary $\mathbf{D} \in \mathbb{R}^{N \times P}$ with unit-norm columns, whose sub-matrices $\mathbf{D}_i^m$ satisfy the RIP of order K with constant δ_K^i, and suppose that, we have obtained an alternative dictionary $\mathbf{D}' \in \mathbb{R}^{N \times P}$, whose sub-matrices $\mathbf{D}_i'^m$ also satisfy the RIP of order K with constant δ_K^i such that, for the incomplete observation $\mathbf{x}_i^o \in \mathbb{R}^{M_i}$, the K-sparse representation solution is non-unique, i.e. $\exists \mathbf{s}_i, \mathbf{s}_i' \in \Sigma_K^P$ such that $\mathbf{x}_i^o = \mathbf{D}_i^o \mathbf{s}_i = \mathbf{D}_i'^o \mathbf{s}_i'$, where $\mathbf{s}_i \in \mathbb{R}^P$ is the vector of coefficients of the true data, i.e. $\mathbf{x}_i = \mathbf{D}\mathbf{s}_i$ and $\mathbf{s}_i'$ provides a plausible reconstruction through $\hat{\mathbf{x}}_i = \mathbf{D}'\mathbf{s}_i'$ with $\|\hat{\mathbf{x}}_i\| \leq 1$. If a perfect classifier $\{\mathbf{w}, b\}$ of the reconstruction $\hat{\mathbf{x}}_i$ exists such that $|f(\hat{\mathbf{x}})| = |\langle \mathbf{w}, \hat{\mathbf{x}} \rangle + b| > \epsilon_i > 0$ and*

$$\epsilon_i > 2\|\mathbf{w}_i^m\|_1 \sqrt{\frac{K}{1 - \delta_K^i}}, \tag{15}$$

then the full data vector $\mathbf{x}_i$ is also perfectly separated with this classifier, in other words: $f(\mathbf{x}_i) = \langle \mathbf{w}_i, \mathbf{x}_i \rangle + b > 0$ (≤ 0) if $y_i = 1$ ($y_i = 0$).

Proof. Let us prove that the sufficient condition (6) is met under the hypothesis of Theorem V.1. Taking into account that

TABLE III: Experimental settings for MNIST and CIFAR10 datasets: Number of iterations N_{iter}, batch size bs, learning rate σ_Θ, momentum m, update rate σ (training and test)

Dataset	Classifier	N_{iter}	bs	σ_Θ	m	σ (train)	σ (test)
MNIST	Log. Reg.	3000	250	0.1	0.5	1.0	5.0
	CNN4	3500	250	05	0.5	0.4	0.5
CIFAR10	Resnet18	1000	64	0.01	0.5	1.0	2.5

$\mathbf{x}_i^m = \mathbf{D}_i^m \mathbf{s}_i$, we can write

$$|\langle \mathbf{w}^m, \mathbf{D}_i^m \mathbf{s}_i \rangle| = \Big| \sum_{j=1}^{N-M_i} \mathbf{w}^m(j) \sum_{n=1}^{N} \mathbf{D}_i^m(j,n)\mathbf{s}_i(n) \Big| \leq \sum_{j=1}^{N-M_i} |\mathbf{w}^m(j)| \sum_{n=1}^{N} |\mathbf{D}_i^m(j,n)||\mathbf{s}_i(n)|. \quad (16)$$

Since we assumed normalized vectors $\|\mathbf{x}_i\| \leq 1$, by applying the left-hand side of the RIP we obtain: $\|\mathbf{s}_i\| \leq 1/\sqrt{1-\delta_K^i}$, and, taking into account that $\|\mathbf{s}_i\|_1 \leq \sqrt{K}\|\mathbf{s}_i\|$ and $|\mathbf{D}_i^m(j,n)| \leq 1$ (columns of $\mathbf{D}$ are unit-norm), we obtain:

$$|\langle \mathbf{w}^m, \mathbf{D}_i^m \mathbf{s}_i \rangle| \leq \sqrt{\frac{K}{1-\delta_K^i}} \sum_{j=1}^{N-M_i} |\mathbf{w}^m(j)| = \sqrt{\frac{K}{1-\delta_K^i}} \|\mathbf{w}_i^m\|_1. \quad (17)$$

Similarly, using $\hat{\mathbf{x}}_i{}^m = \mathbf{D}_i'^m \mathbf{s}_i'$, we can obtain that

$$|\langle \mathbf{w}^m, \mathbf{D}_i'^m \mathbf{s}_i' \rangle| \leq \sqrt{\frac{K}{1-\delta_K^i}} \|\mathbf{w}_i^m\|_1. \quad (18)$$

Putting equations (17) and (18) together with equation (15) complete the proof of the sufficient condition (6). □

C. Experimental results details

1) Implementation: We implemented all the algorithms in Pytorch 1.0.0 on a single GPU. The code is available at[5].

Initializations of dictionary $\mathbf{D}$ and coefficients $\mathbf{s}_i$ were made at random. However, we think some improvements in convergence could be achieved by using some dedicated dictionaries such as the case of Wavelet or Cosine Transform matrices.

To update NN weights (Θ), we used standard Stochastic Gradient Descent (SGD) with learning rate σ_Θ and momentum m, while for updating dictionary $\mathbf{D}$ and vector coefficients $\mathbf{s}_i$, we used fixed update rate $\sigma = \sigma_\mathbf{D} = \sigma_\mathbf{s}$. It is noted that we used different update rates for training and testing stages. In Table III, we report the settings used for experiments for MNIST and CIFAR10 datasets, which includes Number of iterations N_{iter}, batch size bs, learning rate σ_Θ, momentum m, update rate σ (training and test).

2) Hyperparameter tuning: In Table IV we present the results of the grid search for hyper-parameter tuning on MNIST and CIFAR10 datasets. We fit our model to the training dataset for a range of values of parameters λ_1 and λ_2 and apply it to a validation data set. Figure 5 shows the validation accuracy obtained with different classifiers and levels of missing entries for MNIST dataset.

[5]https://github.com/ccaiafa/SimultCodClass.

TABLE IV: Hyper-parameter tuning: crossvalidated hyperparameters λ_1 and λ_2 obtained for MNIST and CIFAR10 datasets with the classifiers used in our experiments.

Dataset	Classifier	Random missing entries						Occlusion	
		75%		50%		25%		50%	
		λ_1	λ_2	λ_1	λ_2	λ_1	λ_2	λ_1	λ_2
MNIST	Log. Reg.	0.32	1.28	0.64	1.28	0.64	1.28	-	-
	CNN4	1.28	1.28	2.56	1.28	5.12	1.28	10.24	10.24
CIFAR10	Resnet18	0.024	0.008	0.032	0.004	0.032	0.01	-	-

3) Additional visual results: To visually evaluate our results, additional randomly selected examples of original (complete) images of the test dataset in MNIST and Fashion, together with their given incomplete observations and obtained reconstructions, are shown in Figure 6 and Figure 7

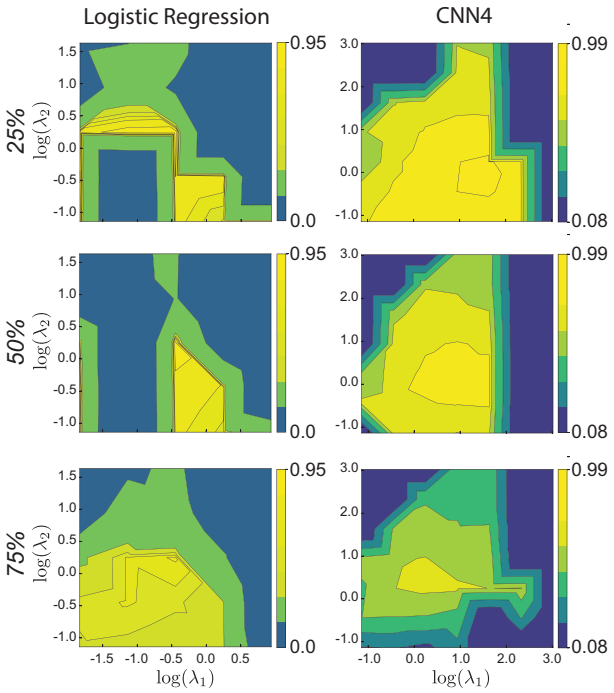

Fig. 5: Test accuracy in the grid search for hyper-parameter tuning in MNIST dataset: λ_1 and λ_2 were tuned by cross-validation for various levels of missing entries: 25%, 50% and 75%.

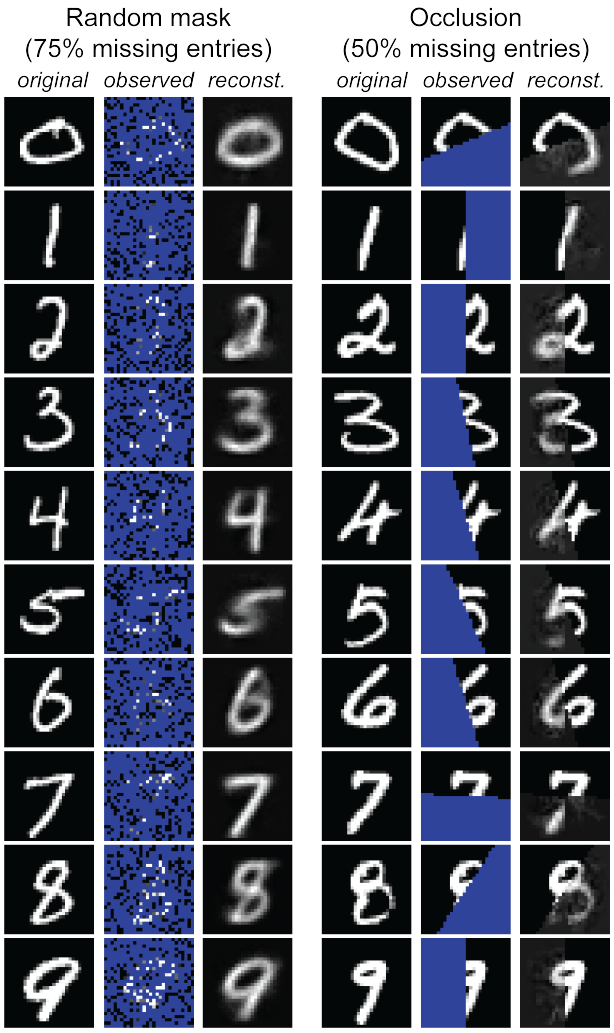

Fig. 6: Reconstructions of incomplete test MNIST dataset images by applying our simultaneous classification and coding algorithm with the CNN4 architecture.

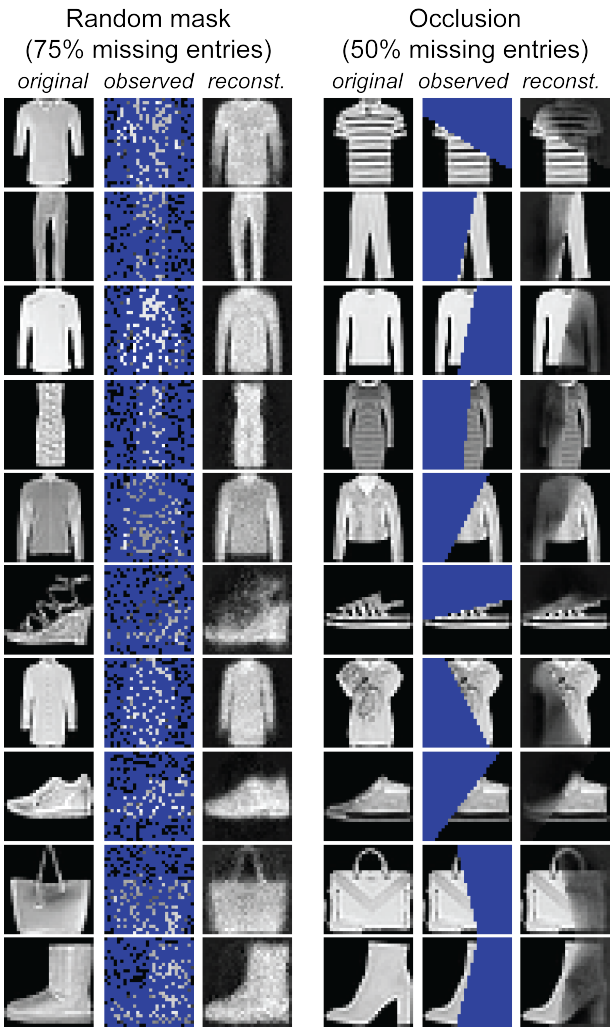

Fig. 7: Reconstructions of incomplete test Fashion dataset images by applying our simultaneous classification and coding algorithm with the CNN4 architecture with Batch Normalization (BN).