Supplementary Materials
GAIN: Missing Data Imputation using Generative Adversarial Nets

Jinsung Yoon [1 *] **James Jordon** [2 *] **Mihaela van der Schaar** [1 2 3]

1. Proofs

1.1. Proof of Lemma 1

For $t \in \{0,1\}$ define the set $M_t^i = \{\mathbf{m} \in \{0,1\}^d : m_i = t\}$.

$$\begin{aligned}
V(D,G) &= \mathbb{E}_{\tilde{\mathbf{X}},\mathbf{z},\mathbf{M},\mathbf{H}}\Big[\mathbf{M}^T \log D\big(G(\tilde{\mathbf{X}},\mathbf{M}),\mathbf{H}\big) + (\mathbf{1}-\mathbf{M})^T \log\Big(\mathbf{1} - D\big(G(\tilde{\mathbf{X}},\mathbf{M}),\mathbf{H}\big)\Big)\Big] \\
&= \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{H}}\Big[\mathbf{M}^T \log D(\hat{\mathbf{X}},\mathbf{H}) + (\mathbf{1}-\mathbf{M})^T \log\big(\mathbf{1} - D(\hat{\mathbf{X}},\mathbf{H})\big)\Big] \\
&= \int_{\mathcal{X}} \sum_{\mathbf{m}\in\{0,1\}^d} \int_{\mathcal{H}} \big(\mathbf{m}^T \log D(\mathbf{x},\mathbf{h}) + (\mathbf{1}-\mathbf{m})^T \log(\mathbf{1} - D(\mathbf{x},\mathbf{h}))\big)\, p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{\mathbf{m}\in\{0,1\}^d} \left(\sum_{i:m_i=1} \log D(\mathbf{x},\mathbf{h})_i + \sum_{i:m_i=0} \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i)\right) p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\overset{(1)}{=} \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^d \left(\sum_{\mathbf{m}\in M_1^i} \log D(\mathbf{x},\mathbf{h})_i + \sum_{\mathbf{m}\in M_0^i} \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i)\right) p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^d \left(\log D(\mathbf{x},\mathbf{h})_i \sum_{\mathbf{m}\in M_1^i} p(\mathbf{x},\mathbf{m},\mathbf{h})\right) + \left(\log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i) \sum_{\mathbf{m}\in M_0^i} p(\mathbf{x},\mathbf{m},\mathbf{h})\right) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^d \log D(\mathbf{x},\mathbf{h})_i p(\mathbf{x},\mathbf{h},m_i=1) + \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i) p(\mathbf{x},\mathbf{h},m_i=0) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x}
\end{aligned}$$

where (1) follows from switching the order of summation. We then note that $y \mapsto a\log y + b\log(1-y)$ achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$ and so $V(D,G)$ is maximized (for fixed G) when

$$D(\mathbf{x},\mathbf{h})_i = \frac{p(\mathbf{x},\mathbf{h},m_i=1)}{p(\mathbf{x},\mathbf{h},m_i=0) + p(\mathbf{x},\mathbf{h},m_i=1)} \tag{1}$$

for each $i \in \{1,...,d\}$. $\square$.

---

*Equal contribution [1]University of California, Los Angeles, CA, USA [2]University of Oxford, UK [3]Alan Turing Institute, UK. Correspondence to: Jinsung Yoon <jsyoon0823@gmail.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.2. Proof of Theorem 1

Denote by $\mathcal{H}_t^i$ the space $\{\mathbf{h} \in \mathcal{H} : p_h(\mathbf{h}|m_i = t) > 0\}$.

$$\begin{aligned}
C(G) &= \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{H}}\Big(\sum_{i:M_i=1} \log p_m(m_i = 1|\hat{\mathbf{X}}, \mathbf{H}) + \sum_{i:M_i=0} \log p_m(m_i = 0|\hat{\mathbf{X}}, \mathbf{H})\Big) \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^{d} \big[p(\mathbf{x}, \mathbf{h}, m_i = 1) \log p_m(m_i = 1|\mathbf{x}, \mathbf{h}) + p(\mathbf{x}, \mathbf{h}, m_i = 0) \log p_m(m_i = 0|\mathbf{x}, \mathbf{h})\big] \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log p_m(m_i = t|\mathbf{x}, \mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{p(\mathbf{x}, m_i = t|\mathbf{h})}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{p(\mathbf{x}, m_i = t|\mathbf{h}) p_m(m_i = t|\mathbf{h})}{\hat{p}(\mathbf{x}|\mathbf{h}) p_m(m_i = t|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\overset{(2)}{=} \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t)}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(m_i = t, \mathbf{h}) \hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) \log \frac{\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t)}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\qquad + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p_m(m_i = t, \mathbf{h}) D_{\mathrm{KL}}(\hat{p}(\cdot|\mathbf{h}, m_i = t)||\hat{p}(\cdot|\mathbf{h})) + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h}
\end{aligned}$$

where (2) follows by using the identity $\log(ab) = \log(a) + \log(b)$ and integrating out $\mathbf{x}$ from the resulting second term.

We then note that for any two densities, p and q, the Kullback-Leibler divergence $D_{\mathrm{KL}}(p||q)$ of p with respect to q is 0 if and only if $p = q$ (almost everywhere).

It follows that C(G) is minimized if only if for every $i \in \{1, ..., d\}$, $t \in \{0, 1\}$, $\mathbf{h} \in \mathcal{H}_t^i$ we have that for (almost) every $\mathbf{x}$, $\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{h})$.

The minimum value is then given by $\sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h})\mathrm{d}\mathbf{h}$, which lines up with the intuition that the worst a discriminator can (always) do is to predict the mask vector based solely on the hint, using no information in $\hat{\mathbf{x}}$, but exploiting correlations between $\mathbf{m}$ and $\mathbf{h}$. $\square$.

1.3. Proof of Proposition 1

We prove the first claim by constructing an example in which multiple solutions to (12) exist. Suppose that $\mathbf{H}$ is independent of $\mathbf{M}$ (note that this is equivalent to simply not having a hinting mechanism, since we assume $\mathbf{H}$ is (conditionally) independent of $\mathbf{X}$ given $\mathbf{M}$).

Then (12) becomes

$$\hat{p}(\mathbf{x}|m_i = t) = \hat{p}(\mathbf{x}), \text{ for all } i \in \{1, ..., d\}, t \in \{0, 1\}. \tag{2}$$

We then note that (2) holds if and only if

$$\hat{p}(\mathbf{x}|m_i = 0) = \hat{p}(\mathbf{x}|m_i = 1), \text{ for all } i \in \{1, ..., d\}. \tag{3}$$

Consider the case when $\mathbf{X} = (X_1, X_2, X_3)$ with each X_i being a Bernoulli random variable. Then the number of parameters to be specified by the generator (and therefore the number of parameters in p and $\hat{p}$) is 38, whereas the total number of linear equalities defined by (3) is 24. Therefore $\hat{p}$ is not uniquely determined. $\square$.

1.4. Proof of Lemma 2

The result follows immediately by observing that $p_m(m_i = t|h_i = t) = 1$ for $t \in \{0, 1\}$ and by Lemma 1. $\square$.

1.5. Proof of Proposition 2

Let $\mathbf{m}_0, \mathbf{m}_1 \in \{0, 1\}^d$ be such that they differ only on one component and let this component be the ith with the ith component of $\mathbf{m}_0$ being 0 and of $\mathbf{m}_1$ being 1. Then the hint defined by

$$h_j = \begin{cases} m_j \text{ if } j \neq i \\ 0.5 \text{ if } j = i \end{cases} \tag{4}$$

is such that $p_h(\mathbf{h}|m_i = t) > 0$ for $t = 0$ *and* $t = 1$. By Theorem 1 we then have that

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = 0) = \hat{p}(\mathbf{x}|\mathbf{h}, m_i = 1) \tag{5}$$

for all $\mathbf{x} \in \mathcal{X}$.

But then note

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{m}_t, \mathbf{b}) = \frac{\hat{p}(\mathbf{x}|\mathbf{m}_t)}{\mathbb{P}(\mathbf{B} = \mathbf{b}|\mathbf{M} = \mathbf{m}_t)} = \frac{\hat{p}(\mathbf{x}|\mathbf{m}_t)}{\mathbb{P}(\mathbf{B} = \mathbf{b})} \tag{6}$$

where the final equality follows from independence of $\mathbf{B}$ and $\mathbf{M}$.

Substituting this into (5) and multiplying by $\mathbb{P}(\mathbf{B} = \mathbf{b})$ we get

$$\hat{p}(\mathbf{x}|\mathbf{m}_0) = \hat{p}(\mathbf{x}|\mathbf{m}_1). \tag{7}$$

Note that this holds for any $\mathbf{m}_0$, $\mathbf{m}_1$ that differ in at most one component.

Now let $\mathbf{m}_1, \mathbf{m}_2$ be *any* two vectors in $\{0, 1\}^d$. Then there exists $k \in \mathbb{N}$ and a sequence of vectors $\mathbf{m}'_1, ..., \mathbf{m}'_k$ such that $\mathbf{m}'_i$ and $\mathbf{m}'_{i+1}$ differ only on the ith component and $\mathbf{m}_1 = \mathbf{m}'_1$ and $\mathbf{m}_2 = \mathbf{m}'_k$.

It follows from (7) that

$$\hat{p}(\mathbf{x}|\mathbf{m}_1) = \hat{p}(\mathbf{x}|\mathbf{m}'_1) = ... = \hat{p}(\mathbf{x}|\mathbf{m}'_k) = \hat{p}(\mathbf{x}|\mathbf{m}_2) \tag{8}$$

and so in particular, for any $\mathbf{m} \in \{0, 1\}^d$ we have that

$$\hat{p}(\mathbf{x}|\mathbf{m}) = \hat{p}(\mathbf{x}|\mathbf{1}). \tag{9}$$

Since $\hat{p}(\mathbf{x}|\mathbf{1})$ is the density of $\mathbf{X}$, it is unique, and so, the density satisfying (12) is necessarily unique. $\square$.

2. Dataset description in details

2.1. Synthetic data generation

We construct a synthetic dataset using a joint Gaussian distribution with zero means and randomly generated co-variance matrix (C). More specifically, each element of a matrix A is randomly sampled from the uniform distribution $\mathcal{U}(0, 1/2)$. Then, the co-variance matrix C is given by $C = A + A^T$.

Table 1. Statistics of the datasets. S_{cont}: the number of continuous variables, S_{cat}: the number of categorical variables, Avg. Corr: the average absolute correlations among features.

Dataset	N	$\mathbf{S_{cont}}$	$\mathbf{S_{cat}}$	Avg. Corr
Breast	569	30	0	0.3949
Spam	4,601	57	0	0.0608
Letter	20,000	0	16	0.1829
Credit	30,000	14	9	0.1633
News	39,797	44	14	0.0688

3. Hyper-parameters

In all experiments, the depth of the generator and discriminator in both GAIN and auto-encoder is set to 3. The number of hidden nodes in each layer is d, $d/2$ and d, respectively. We use $\tanh$ as the activation functions of each layer except for the output layer where we use the sigmoid activation function and the number of batches is 64 for both the generator and discriminator. For the GAIN algorithm, we use cross-validation to select α among $\{0.1, 0.5, 1, 2, 10\}$.

We use tensorflow to implement GAIN and auto-encoder. We use R to implement MICE (R package MICE (Buuren & Groothuis-Oudshoorn, 2011)), MissForest (R package MissForest (Stekhoven, 2013)), Matrix Completion (R package softImpute (Hastie & Mazumder, 2015)), and EM (R package Amelia (Honaker et al., 2011)).

4. Additional experiments

4.1. Understanding GAIN

Fig. 1 visualizes the convergence of the generator and the discriminator on synthetic data generated as described above. The top figure shows the ground truth of the mask matrix. The three left figures show the estimations of the mask matrix made by the discriminator. As the epoch number increases, it becomes more difficult for the discriminator to distinguish the imputed and observed components (the generator produces imputed components closer to the ground truth). The three right figures show the imputation accuracy of the generator. As the epoch number increases, the imputed components are closer to the ground truth of the missing components.

4.2. Learning curves of GAIN

Fig 2 illustrates the learning curves of the generator and the discriminator of GAIN. As seen in Fig 2 (a), the cross entropy loss of the discriminator converges around $0.7 (\simeq \log(0.5))$ which represents that the discriminator is barely able to distinguish the observed and imputed components. As seen in Fig 2 (b), the RMSE loss of the generator converges to less than 0.1 indicating that the generator imputes the missing components accurately. This result is consistent with Fig. 1.

4.3. Categorical value imputation

As an additional performance metric for comparing the imputation quality of categorical features, we compute the proportion of falsely classified entries (PFC) defined by

$$PFC = \frac{\sum_{i=1}^{n} \sum_{j \in \mathcal{S}_{cat}} (1 - m_j(i)) \mathbb{I}(\hat{x}_j(i) \neq x_j(i))}{\sum_{i=1}^{n} \sum_{j \in \mathcal{S}_{cat}} (1 - m_j(i))},$$

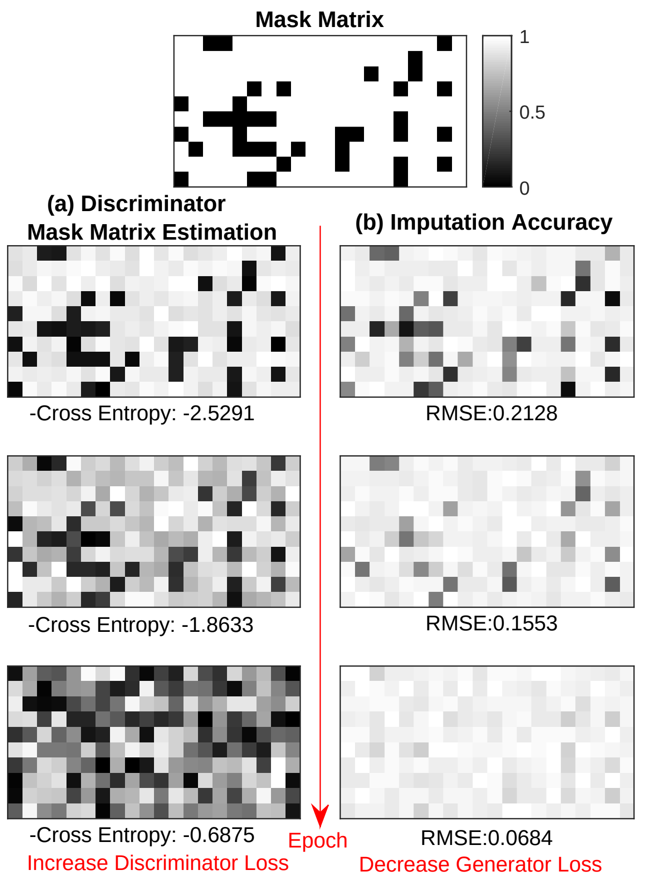

Figure 1. Visualization of the convergence of GAIN. (a) Discriminator outputs, (b) Imputation accuracy of the generator

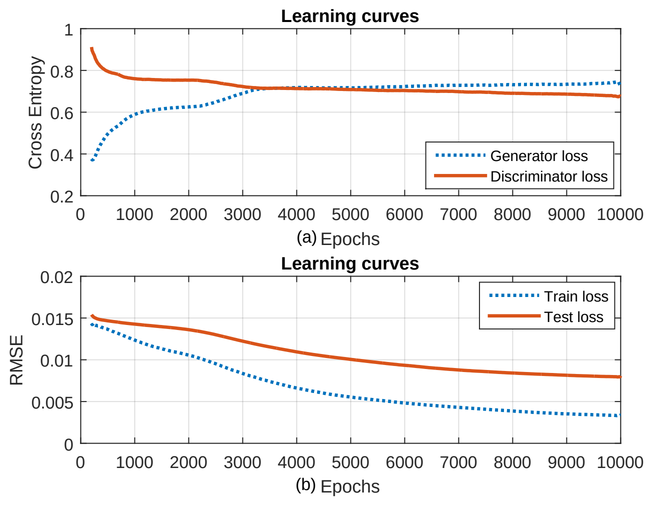

Figure 2. Learning curves of GAIN. (a) Cross entropy loss, (b) RMSE loss

where $\mathcal{S}_{cat}$ is the set of categorical variables. Note that a lower PFC indicates better imputation. We evaluate the PFC performance on the 3 real datasets which have categorical variables (Letter, Credit, News). As can be seen in Table 2, GAIN outperforms all the benchmarks in all the datasets in terms of the PFC metric as well.

Table 2. Imputation performance in terms of PFC (Average ± Std)

Algorithms	Letter	Credit	News
GAIN	**.714 ± .002**	**.478 ± .011**	**.063 ± .002**
MICE	.756 ± .001	.491 ± .013	.072 ± .008
MissForest	.787 ± .001	.499 ± .015	.068 ± .002
Matrix	.779 ± .002	.533 ± .029	.142 ± .003
Auto-encoder	.748 ± .003	.575 ± .016	.065 ± .009
EM	.801 ± .002	.738 ± .010	.091 ± .002

4.4. Estimation of joint distribution

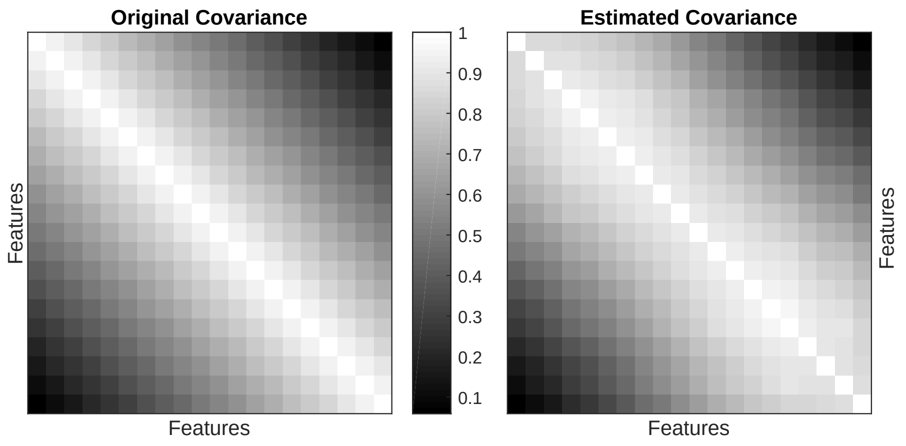

Figure 3. The estimation of joint distribution using GAIN. (a) Original co-variance, (b) Estimated co-variance by GAIN

Fig. 3 shows the qualitative performance of GAIN in estimating the joint distribution $P(\mathbf{X})$. In this experiment, we again use synthetic data and randomly remove 50% of the data. Then, we use GAIN to impute the missing components and estimate the co-variance matrix. As verified by Fig. 3 (a) and (b), the covariance matrix estimated from the GAIN-generated dataset is close to the covariance matrix of the actual data. This verifies that the distribution of $\hat{\mathbf{X}}$ is similar to that of $\mathbf{X}$.

4.5. Preliminary results in error concealment with MNIST

Fig. 4 and 5 visualize the performance of GAIN using MNIST data (LeCun & Cortes, 2010) as an application to error concealment. In this experiment, we randomly remove 50% of the original MNIST image and use GAIN to impute those missing components. Fig 4 shows the imputed images by GAIN after 100, 500, 1000, and 5000 epochs. As the epochs increase, the imputed images become more like the original image.

Fig. 5 shows a visualization of performing multiple imputations (Rubin, 2004) using GAIN. Three imputed images are generated with differently sampled random values $\mathbf{z} \sim \mathcal{U}((0,1)^d)$. The three imputed images are different which qualitatively shows that GAIN captures the uncertainty in the missing components.

4.6. Types of Missingness

As mentioned in Section 1, there are three types of missingness. Here we recall the definition of the first, and formalize the other two, and then provide empirical results demonstrating GAIN's performance in each of these settings.

MCAR: Data is said to be Missing Completely at Random (MCAR) if:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{M} \tag{10}$$

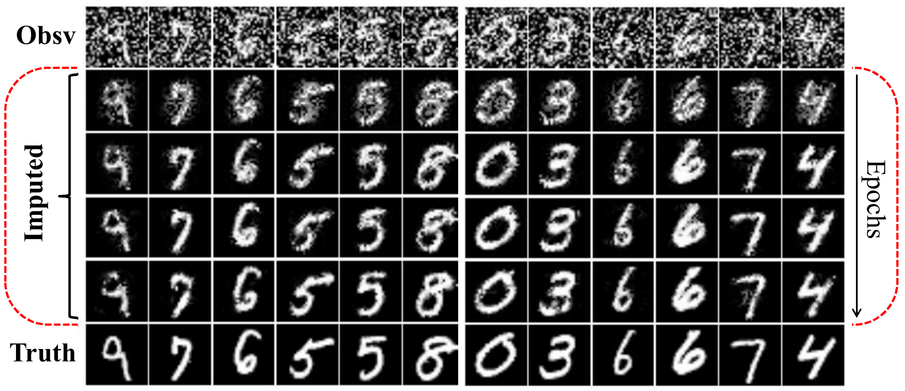

Figure 4. The results of error concealment applied to MNIST dataset. The first row represents the original image with 50% missing. From the second to fifth rows represent the imputed image with different epochs (100, 500, 1000, 5000) by GAIN. The last row represents the original image.

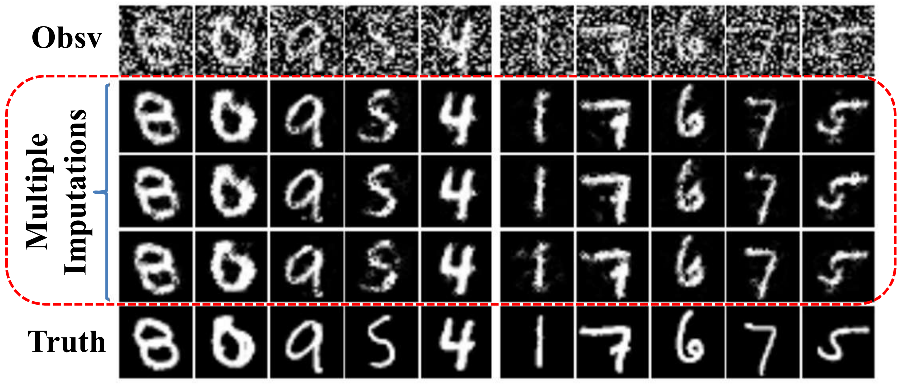

Figure 5. Multiple imputation results with MNIST dataset. Rows 2-4 represent the multiple imputed images by GAIN.

MAR: Data is said to be Missing at Random (MAR) if:

$$\begin{aligned}\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \mathbf{m} \in \{0,1\}^d \text{ s.t. } \tilde{\mathbf{x}}_1 = \tilde{\mathbf{x}}_2 \text{ (w.r.t. } \mathbf{m}) \\ \mathbb{P}(\mathbf{M} = \mathbf{m} | \mathbf{X} = \mathbf{x}_1) = \mathbb{P}(\mathbf{M} = \mathbf{m} | \mathbf{X} = \mathbf{x}_2)\end{aligned} \tag{11}$$

MNAR: Data is said to be Missing Not at Random (MNAR) if it is neither MCAR or MAR (in particular, the missingness can depend on the values of the *unobserved* data points).

The following explains how we constructed datasets that satisfy the latter two mechanisms.

Missing at random (MAR): To create an MAR dataset, we sequentially define the probability that the ith component of the nth sample is observed conditional on the missingness and values (if observed) of the previous $i-1$ components to be

$$P_i^m(n) = \frac{p^m(i) \cdot N \cdot e^{-\sum_{j<i} w_j m_j(n) x_j(n) + b_j(1-m_j(n))}}{\sum_{l=1}^N e^{-\sum_{j<i} w_j m_j(l) x_j(l) + b_j(1-m_j(l))}}$$

where $p^m(i)$ corresponds to the average missing rate of the ith feature, and w_j, b_j are sampled from $\mathcal{U}(0,1)$ (but are only sampled once for the entire dataset). We sequentially sample $m_1, ..., m_d$ for each feature vector.

Missing not at random (MNAR): To create an MNAR dataset, we define the probability that the ith component of the nth sample is observed ($P_i^m(n)$) to be

$$P_i^m(n) = \frac{p^m(i) \cdot N \cdot e^{-w_i x_i(n)}}{\sum_{l=1}^N e^{-w_i x_i(l)}}$$

where again $p^m(i)$ corresponds to the average missing rate of the ith feature and w_i is sampled from $\mathcal{U}(0, 1)$. In particular, the missingness of a data point is directly dependent on its value (with dependence determined by the weight w_i).

We compare the RMSE of GAIN against other imputation algorithms on both an MAR and MNAR version of the Credit dataset. To make a fair comparison, we pass the mask matrix to all the benchmarks as an additional input so that they can also utilize the informative missingness captured by it.

Different missing rates for different features: In order to also explore the effect of different missing rates across features on the imputation performance of GAIN, we compare the MCAR, MAR and MNAR settings when $p^m(i) = 0.2 \; \forall i \in \{1, ..., d\}$ (uniform) and when $p^m(i) = 0.4 \times \hat{p}^m(i)$ where $\hat{p}^m(i) \sim \mathcal{U}(0, 1)$ (non-uniform). The average missing rate in both cases is 0.2.

Table 3. Imputation performance with uniform and non-uniform $p^m(i)$ in terms of MCAR, MAR, and MNAR (Average ± Std of RMSE)

Setting	Uniform			Non-uniform		
	MCAR	MAR	MNAR	MCAR	MAR	MNAR
GAIN	**.1858 ± .0010**	**.1974 ± .0006**	**.4046 ± .0053**	**.2114 ± .0007**	**.2245 ± .0008**	**.4672 ± .0066**
MICE	.2585 ± .0011	.2574 ± .0035	.5310 ± .0207	.2574 ± .0014	.2344 ± .0068	.5355 ± .0036
MissForest	.1976 ± .0015	.2194 ± .0065	.4286 ± .0087	.2496 ± .0065	.2537 ± .0097	.4784 ± .0102
Matrix	.2602 ± .0073	.2473 ± .0070	.4328 ± .0036	.2356 ± .0022	.2440 ± .0122	.5216 ± .0084
Auto-encoder	.2388 ± .0005	.2405 ± .0070	.4876 ± .0097	.2444 ± .0037	.2498 ± .0129	.5017 ± .0078
EM	.2604 ± .0015	.2755 ± .0063	.5157 ± .0039	.2620 ± .0010	.3339 ± .0024	.4998 ± .0053

As can be seen in Table 3, GAIN outperforms other state-of-the-art imputation methods in all three missingness settings (both when feature missingness is uniform and non-uniform) and shows significantly better performance in the MNAR setting.

As can also be seen from the right hand side of Table 3, GAIN still outperforms all benchmarks in the non-uniform setting, although the performance of both GAIN and MissForest (its closest competitor in the uniform setting) both decrease similarly, while MICE and Matrix completion both show improvements for the non-uniform setting.

Note that the standard deviation of the total number of missing points is higher for non-uniform $p^m(i)$ than uniform $p^m(i)$. As consistent with Fig. 2(a), higher/lower missing rates yield higher/lower imputation errors; and so, due to the increased standard deviation, there is a greater variance in the performance in the non-uniform setting.

References

Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

Hastie, T. and Mazumder, R. *softImpute: Matrix Completion via Iterative Soft-Thresholded SVD*, 2015. URL `https://CRAN.R-project.org/package=softImpute`. R package version 1.4.

Honaker, J., King, G., and Blackwell, M. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7): 1–47, 2011. URL `http://www.jstatsoft.org/v45/i07/`.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Rubin, D. B. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

Stekhoven, D. J. *missForest: Nonparametric Missing Value Imputation using Random Forest*, 2013. R package version 1.4.