HexaGAN: Generative Adversarial Nets for Real World Classification

Uiwon Hwang [1] **Dahuin Jung** [1] **Sungroh Yoon** [1]

Abstract

Most deep learning classification studies assume clean data. However, dirty data is prevalent in real world, and this undermines the classification performance. The data we practically encounter has problems such as 1) missing data, 2) class imbalance, and 3) missing label. Preprocessing techniques assume one of these problems and mitigate it, but an algorithm that assumes all three problems and resolves them has not yet been proposed. Therefore, in this paper, we propose HexaGAN, a generative adversarial network (GAN) framework that shows good classification performance for all three problems. We interpret the three problems from a similar perspective to solve them jointly. To enable this, the framework consists of six components, which interact in an end-to-end manner. We also devise novel loss functions corresponding to the architecture. The designed loss functions achieve state-of-the-art imputation performance with up to a 14% improvement and high-quality class-conditional data. We evaluate the classification performance (F1-score) of the proposed method with 20% missingness and confirm up to a 5% improvement in comparison with the combinations of state-of-the-art methods.

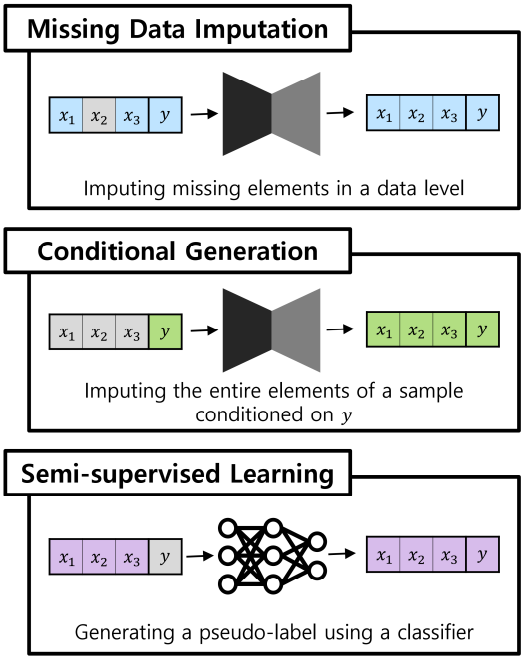


Figure 1. Tasks for three main problems in real world classification. We define missing data imputation as a task that fills in missing data elements. Conditional generation can be defined as a task that imputes the entire elements in an instance conditioned on a certain class. Semi-supervised learning can be defined as a task that imputes missing labels.

1. Introduction

As deep learning models have achieved super-human performance in image classification tasks (He et al., 2016), there have been growing attempts to apply them to more complex tasks such as object detection (Ren et al., 2015), text classification (Zhang et al., 2015), and disease prediction (Hwang et al., 2017). However, real world data is somewhat *dirty*, which prevents a classifier from being fully effective. Generally, dirty data is a data with missing data, class imbalance, and missing label problems. A preprocessing phase is inevitable to improve classification performance and has long been the subject of research. However, a preprocessing technique that solves the three problems concurrently has not yet been proposed. Therefore, we first propose a framework that shows robust performance when used with data that has all three problems.

The work to fill in missing information in data is called imputation (Van Buuren, 2018). The problem of missing data not only leads to deep learning models being unable to fully exploit the given data, but improperly imputed data can mislead the models to learn the wrong data distribution. For example, there are many missing values in user data for recommender systems (Koren et al., 2009) and in electronic health records for ubiquitous healthcare systems (Miotto et al., 2016), which is a considerable barrier to utilizing deep learning based classifiers. There are three main

[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea. Correspondence to: Sungroh Yoon <sryoon@snu.ac.kr>.

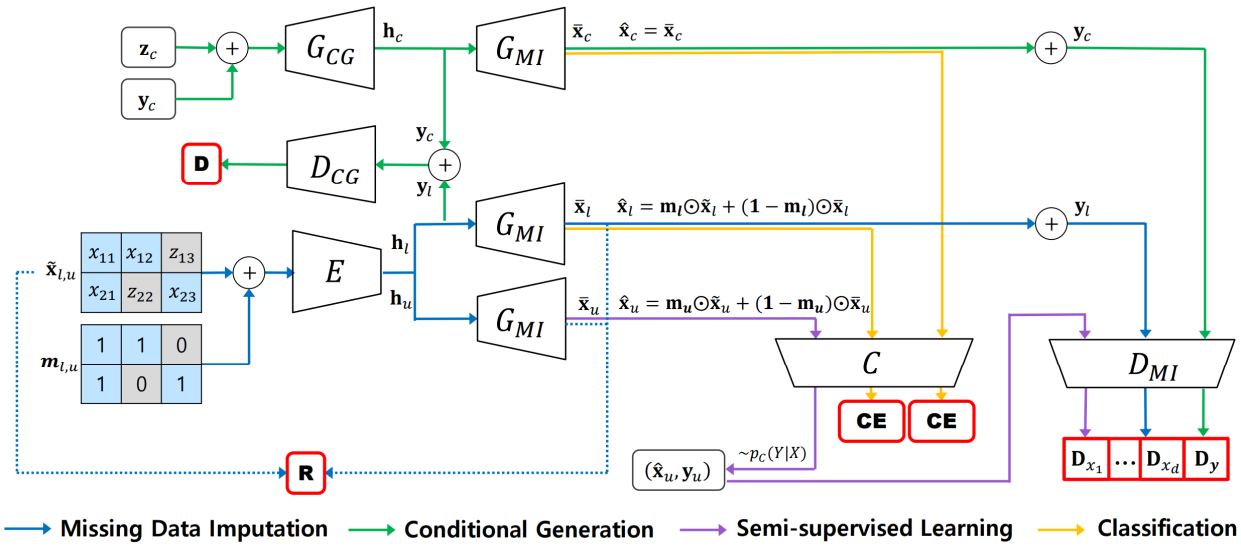


Figure 2. Overview of HexaGAN model. Subscripts l, u, and c indicate that a vector is from labeled data, unlabeled data, and class-conditional data respectively. $\tilde{\mathbf{x}}$ denotes a data instance whose missing values are replaced with noise. $\bar{\mathbf{x}}$ denotes a data generated by G_{MI}. $\hat{\mathbf{x}}$ denotes a data whose missing values are filled with the generated values. $\mathbf{y}$ is a class label. Unlike $\mathbf{y}_l$ and $\mathbf{y}_c$, $\mathbf{y}_u$ is produced by C. $\mathbf{m}$ is a vector that indicates whether corresponding elements are missing or not. $\mathbf{h}$ is a vector in the hidden space. $\mathbf{R}$ is the reconstruction loss. $\mathbf{D}$ is the adversarial loss function between G_{CG} and D_{CG}. $\mathbf{D}_{x_i}$ and $\mathbf{D}_y$ represent the element-wise adversarial loss function. **CE** represents the cross-entropy loss.

types of missing data (Rubin, 1976): 1) MCAR means "data are missing completely at random" and its missingness has no relationship with observed or unobserved variables. 2) MAR means "data are missing at random" and its missingness is related only to observed variables. 3) MNAR means "data are missing not at random" and its missingness can be related to both observed and unobserved variables. Of these three types, our theoretical results and experiments are under MCAR. There are existing techniques used to solve the missing data problem such as matrix completion (Hastie et al., 2015), k-nearest neighbors (Troyanskaya et al., 2001), MICE (Buuren & Groothuis-Oudshoorn, 2010), denoising autoencoders (Vincent et al., 2008), and a generative adversarial networks (GAN) based methods (Yoon et al., 2018; Shang et al., 2017).

When data is collected for real world applications such as anomaly detection (Chandola et al., 2009) and disease prediction (Khalilia et al., 2011), the classes are often imbalanced. Hitherto, various preprocessing techniques to overcome the class imbalance problem have been studied. For example, there are techniques for oversampling minority classes such as the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002) and adaptive synthetic (ADASYN) sampling (He et al., 2008). Cost sensitive loss is also used to solve the class imbalance problem by differentiating cost weights according to classes (Sun et al., 2007). However, in order to oversample from the entire data distribution, all data samples must be considered at the same time, which causes an increase in memory consumption. It has been demonstrated that cost sensitive loss overfits to the data belonging to minority classes (Elrahman & Abraham, 2013).

To consume less memory, as well as to circumvent overfitting, we train a deep generative model to follow the true data distribution and generate samples of minority classes in each batch. In order to address the class imbalance problem using the generative model, the model must be capable of controlling conditional generation. As shown in Figure 1, we take into account the conditional generation as imputation because conditional generation can be defined as imputing whole elements conditioned on a certain class label.

In deep learning, the amount of training data has a significant impact on performance. When the amount of labeled data is small, the amount of data available for supervised learning is reduced, which commonly hinders the performance of the model. This is called a missing label problem. In real world applications such as natural language model (Turian et al., 2010) or healthcare system (Beaulieu-Jones et al., 2016), unlabeled data is abundant, and the cost for labeling is expensive. Thus, related researchers have proposed semi-supervised methods to leverage unlabeled data.

Semi-supervised learning can be divided into regularization and generative approaches. The regularization approach adds a regularization loss term, which is designed on the assumption that adjacent data points or the same architectural

data points are likely to have the same label (Wang & Zhang, 2008; Laine & Aila, 2017; Grandvalet & Bengio, 2005; Miyato et al., 2015; Tarvainen & Valpola, 2017). Unlike the regularization approach, the generative approach enhances the performance of a classifier by utilizing unlabeled data in the process of training the generative model (Kingma et al., 2014; Abbasnejad et al., 2017; Salimans et al., 2016; Springenberg, 2015; Dai et al., 2017).

As depicted in Figure 1, we define missing data, class imbalance, and missing label problems in terms of imputation. Based on the insight about the imputation, we find out that networks for imputation can play multiple roles. Moreover, solving the three data problems simultaneously is more effective than solving them in a cascading form. In this paper, we propose a GAN framework consisting of six components to solve the three problems in real world classifications. We derive a new objective function for the imputation of missing data, and demonstrate that it performs better than the existing state-of-the-art imputation methods. We define conditional generation from the perspective of conditional imputation, and confirm that the proposed method works successfully by designing the imputation model to be a part of the framework. Finally, in order to facilitate semi-supervised learning, a classifier generates a synthetic class label for unlabeled data, and a discriminator distinguishes fake labels from real labels.

In summary, our contributions are as follows:

- To the best of our knowledge, this is one of the first studies that defines the three problems (missing data, class imbalance, and missing label) in terms of imputation. Then, we propose HexaGAN to encourage thorough imputation of data with these three problems.
- In order to implement real world datasets to existing classifiers, we must apply suitable preprocessing techniques to the datasets. However, our end-to-end framework is simple to use and works automatically when the absence of data elements and labels is indicated ($\mathbf{m}$ and m_y, See Section 3).
- We devise a combination of six components and corresponding cost functions. More specifically, we propose a novel adversarial loss function and gradient penalty for element-wise imputation, confirming that our imputation performance shows stable, state-of-the-art results.
- In real world classification, the proposed method significantly outperforms cascading combinations of the existing state-of-the-art methods. As a result, we demonstrate that the components of our framework interplay to solve the problems effectively.

2. Generative Adversarial Networks

Along with the development of discriminative models represented by classifiers, generative models have also evolved. In particular, GANs (Goodfellow et al., 2014) have been utilized for the generation of high-quality synthetic data in many applications. Although GANs have the most advanced data generation capability, they have a disadvantage that model training is unstable, and there have been many studies done for the quest of stable GANs. Among them, Arjovsky et al. (2017) proposed the Wasserstein GAN (WGAN), which has a smoother gradient by introducing the Wasserstein distance. Several gradient penalties have been proposed (Gulrajani et al., 2017; Mescheder et al., 2018) to make WGAN more stable. In this paper, we modify WGAN loss and zero-centered gradient penalty for missing data imputation. Experimentally, we show that the proposed method has more stable and better imputation performance than the existing vanilla GAN loss-based models.

GAIN (Yoon et al., 2018) is the first method to apply GAN to missing data imputation under MCAR. The typical discriminator predicts whether each *sample* is real or fake. However, it is difficult to identify the difference between real and fake if all data samples have missing data. Instead, GAIN labels each *element* of a sample as missing or not, so that the discriminator can discriminate between real and fake. Our imputation method shares some similarity with this work in terms of labeling each element as real or fake. The imputation performance of GAIN measured by our own implementation with the specific dataset is lower than that of the autoencoder, and the learning curve is also unstable. However, our method provides stable imputation performance and usability by including class-conditional generation for the class imbalance problem, and by interacting with semi-supervised framework.

TripleGAN is a GAN for semi-supervised learning (Li et al., 2017). The key characteristic of this model is that a classifier, a generator, and a discriminator interact together to enable semi-supervised learning. When the classifier creates a pseudo-label for the unlabeled data, an image-label pair is entered into the discriminator. The classifier and discriminator are trained competitively. In this paper, we adopt the pseudo-labeling technique of TripleGAN to allow HexaGAN to perform semi-supervised learning.

3. Proposed Method

The HexaGAN framework is comprised of six components: an encoder (E) that encodes labeled and unlabeled data into the hidden space, a generator for missing imputation (G_{MI}) that imputes missing data, a discriminator for missing imputation (D_{MI}) that distinguishes between missing and non-missing elements and labels, a generator for conditional

generation (G_{CG}) that generates conditional hidden vector $\mathbf{h}_c$, a discriminator for conditional generation (D_{CG}) that discriminates whether a hidden vector is from the dataset or G_{CG}, and a classifier (C) that estimates a class label. The overview of our model is illustrated in Figure 2.

We consider data instances $\mathbf{x}^1, ..., \mathbf{x}^n \in \mathbb{R}^d$, where n is the number of instances and d is the number of elements in an instance. The i-th element in a single data instance x_i is a scalar and some elements are missing. The first n_l instances are labeled data, and the remaining instances are unlabeled. Class labels corresponding to each data instance are denoted as $\mathbf{y}^1, ..., \mathbf{y}^{n_l} \in \mathbb{R}^{n_c}$, where n_c is the number of classes. We also assume boolean vectors $\mathbf{m}^1, ..., \mathbf{m}^n \in \mathbb{R}^d$ indicating whether each element in a sample is missing. If m_i, the i-th element of a vector $\mathbf{m}$, is 0, this means that the corresponding element is missing. We consider $m_y \in \mathbb{R}$ to indicate whether a label of an instance is missing. If m_y is 0, this means the label is missing. Thus, labeled instances exist as a set of $D_l = \{(\mathbf{x}^j, \mathbf{y}^j, \mathbf{m}^j, m_y^j = 1)\}_{j=1}^{n_l}$, and unlabeled instances exist as pairs of $D_u = \{(\mathbf{x}^j, \mathbf{m}^j, m_y^j = 0)\}_{j=1}^{n-n_l}$. We explain our methods for missing data imputation, class-conditional generation, and semi-supervised classification.

3.1. Missing data imputation

Missing data imputation aims to fill in missing elements using the distribution of data represented by the generative model. In our framework, missing data imputation is performed by E, G_{MI}, and D_{MI}. From the viewpoint of D_{MI}, real and fake are not labeled for each data instance, but real (non-missing) and fake (missing) are labeled for each *element* in an instance.

First, we make a noise vector $\mathbf{z} \in \mathbb{R}^d$ that has the same dimension as an input instance $\mathbf{x} = (\mathbf{x}_l \cup \mathbf{x}_u)$, and replace the missing elements in the instance with elements of $\mathbf{z}$ to generate $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \mathbf{m} \odot \mathbf{x} + (1 - \mathbf{m}) \odot \mathbf{z} \tag{1}$$

where $\odot$ is element-wise multiplication. The objective of our framework is to sample the patterns stored in the model that are most suitable for the missing data (i.e., generating samples according to $p(\mathbf{x}|\tilde{\mathbf{x}}, \mathbf{m})$). Then $\tilde{\mathbf{x}}$ is concatenated with m, and the encoder generates a hidden variable $\mathbf{h} \in \mathbb{R}^{d_h}$ on the hidden space:

$$\mathbf{h} = E(\tilde{\mathbf{x}}, \mathbf{m}) \tag{2}$$

where d_h is the dimension of the hidden space. $\mathbf{h}$ goes into G_{MI} to generate $\bar{\mathbf{x}}$. Only the missing elements in the input instance are imputed with the generated values, resulting in $\hat{\mathbf{x}}$ as follows:

$$\bar{\mathbf{x}} = G_{MI}(\mathbf{h}) \tag{3}$$

$$\hat{\mathbf{x}} = \mathbf{m} \odot \mathbf{x} + (1 - \mathbf{m}) \odot \bar{\mathbf{x}} \tag{4}$$

Algorithm 1 Missing data imputation

input : $\mathbf{x}$ - data with missing values sampled from D_l and D_u;
$\mathbf{m}$ - vector indicating whether elements are missing;
$\mathbf{z}$ - noise vector sampled from uniform(0,1)
output : $\hat{\mathbf{x}}$ - imputed data
repeat
  Sample a batch of pairs $(\mathbf{x}, \mathbf{m}, \mathbf{z})$
  $\tilde{\mathbf{x}} \leftarrow \mathbf{m} \odot \mathbf{x} + (\mathbf{1} - \mathbf{m}) \odot \mathbf{z}$
  $\mathbf{h} \leftarrow E(\tilde{\mathbf{x}}, \mathbf{m})$
  $\bar{\mathbf{x}} \leftarrow G_{MI}(\mathbf{h})$
  $\hat{\mathbf{x}} \leftarrow \mathbf{m} \odot \mathbf{x} + (\mathbf{1} - \mathbf{m}) \odot \bar{\mathbf{x}}$
  Update D_{MI} using stochastic gradient descent (SGD)
  $\nabla_{D_{MI}} \mathcal{L}_{D_{MI}} + \lambda_1 \mathcal{L}_{\mathrm{GP}}$
  Update E & G_{MI} using SGD
  $\nabla_{G_{MI}} \mathcal{L}_{G_{MI}} + \alpha_1 \mathcal{L}_{\mathrm{recon}}$
until training loss is converged

$\hat{\mathbf{x}}$, whose missing data are imputed by G_{MI}, is paired with $\mathbf{y}$, ($\hat{\mathbf{x}}$, $\mathbf{y}$). D_{MI} determines whether each element of the pair is real or fake, and the label for ($\hat{\mathbf{x}}$, $\mathbf{y}$) is represented as $\mathbf{m}_{\mathbf{xy}} = (\mathbf{m}, m_y) \in \mathbb{R}^{d+1}$. D_{MI} calculates the adversarial losses by whether the missingness is correctly predicted for each element and label, which is then used to train E, G_{MI}, and D_{MI}. The adversarial loss $\mathcal{L}_{G_{MI}}$ is calculated to train E and G_{MI}, and $\mathcal{L}_{D_{MI}}$ is calculated for D_{MI} as follows:

$$\mathcal{L}_{D_{MI}} = \sum_{i=1}^{d+1} \mathbb{E}_{\hat{\mathbf{x}},\mathbf{y},\mathbf{m}_{xy}} \left[(1 - m_{xy_i}) \cdot D_{MI}(\hat{\mathbf{x}}, \mathbf{y})_i\right] - \mathbb{E}_{\hat{\mathbf{x}},\mathbf{y},\mathbf{m}_{xy}} \left[m_{xy_i} \cdot D_{MI}(\hat{\mathbf{x}}, \mathbf{y})_i\right] \tag{5}$$

$$\mathcal{L}_{G_{MI}} = -\sum_{i=1}^{d+1} \mathbb{E}_{\hat{\mathbf{x}},\mathbf{y},\mathbf{m}_{xy}} \left[(1 - m_{xy_i}) \cdot D_{MI}(\hat{\mathbf{x}}, \mathbf{y})_i\right] \tag{6}$$

where $D_{MI}(\cdot)_i$ is the i-th output element of D_{MI}. The following theorem confirms that the proposed adversarial loss functions make the generator distribution converge to the desired data distribution.

Theorem 1 *A generator distribution $p(\mathbf{x}|\mathbf{m}_i = 0)$ is a global optimum for the min-max game of G_{MI} and D_{MI}, if and only if $p(\mathbf{x}|\mathbf{m}_i = 1) = p(\mathbf{x}|\mathbf{m}_i = 0)$ for all $\mathbf{x} \in \mathbb{R}^d$, except possibly on a set of zero Lebesgue measure.*

Proof of Theorem 1 is provided in Supplementary Materials.

Moreover, we add a reconstruction loss to the loss function of E and G_{MI} in order to exploit information from non-missing elements, as follows:

$$\mathcal{L}_{\mathrm{recon}} = \mathbb{E}_{\hat{\mathbf{x}}|\mathbf{x},\mathbf{m}} \left[\sum_{i=1}^{d} m_i (x_i - \bar{x}_i)^2\right] \tag{7}$$

For more stable GAN training, we modify a simplified version of the zero-centered gradient penalty proposed by Mescheder et al. (2018) and add the gradient penalty to the loss function of D_{MI}. The modified regularizer penalizes gradients of each output unit of the D_{MI} on $p_{\mathcal{D}}(x_i)$:

$$\mathcal{L}_{\text{GP}} = \sum_{i=1}^{d} \mathbb{E}_{p_{\mathcal{D}}(x_i)} \left[||\nabla_{\hat{\mathbf{x}}} D_{MI}(\hat{\mathbf{x}})_i||_2^2 \right] \tag{8}$$

We define $\hat{\mathbf{x}}$ in $p_{\mathcal{D}}(x_i)$ as data with m_i is 1 (i.e., $\{\hat{\mathbf{x}}^j | m_i^j = 1\} \in p_{\mathcal{D}}(x_i)$). In other words, we penalize D_{MI} only on data where the i-th element is not missing in a batch.

Therefore, missing data imputation and model training are performed as described in Algorithm 1. We used 10 for both hyperparameters λ_1 and α_1 in our experiments.

3.2. Conditional generation

We define conditional generation for the class imbalance problem as the imputation of entire data elements on a given class label (i.e., generating $(x_1, ..., x_d)$ following $p(\mathbf{x}|\mathbf{y})$). Since we have G_{MI}, which is a generator for imputation, we can oversample data instances by feeding synthetic $\mathbf{h}$ into G_{MI}. Therefore, we introduce G_{CG} to generate a hidden variable $\mathbf{h}_c$ corresponding to the target class label $\mathbf{y}_c$, i.e., we sample $\mathbf{h}_c \sim p_{G_{CG}}(\mathbf{h}|\mathbf{y})$. We also introduce D_{CG} to distinguish pairs of generated hidden variables and target class labels $(\mathbf{h}_c, \mathbf{y}_c)$ (fake) from pairs of hidden variables for labeled data and corresponding class labels $(\mathbf{h}_l, \mathbf{y}_l)$ (real). G_{CG} and D_{CG} are trained with WGAN loss and zero-centered gradient penalty on $\mathbf{h}_l$ as follows:

$$\begin{aligned}\mathcal{L}_{D_{CG}} =& \mathbb{E}_{\mathbf{h}_c \sim p_{G_{CG}}(\mathbf{h}_c|\mathbf{y}_c)}[D_{CG}(\mathbf{h}_c, \mathbf{y}_c)] \\ &- \mathbb{E}_{\mathbf{h}_l \sim p_E(\mathbf{h}_l|x_l)}[D_{CG}(\mathbf{h}_l, \mathbf{y}_l)] \\ &+ \lambda_2 \mathbb{E}_{\mathbf{h}_l \sim p_E(\mathbf{h}_l|x_l)} \left[||\nabla_{\mathbf{h}_l} D_{CG}(\mathbf{h}_l, \mathbf{y}_l)||_2^2 \right]\end{aligned} \tag{9}$$

$$\mathcal{L}_{G_{CG}} = - \mathbb{E}_{\mathbf{h}_c \sim p_{G_{CG}}(\mathbf{h}_c|\mathbf{y}_c)}[D_{CG}(\mathbf{h}_c, \mathbf{y}_c)] \tag{10}$$

where λ_2 is a hyperparameter for the gradient penalty, and we set λ_2 to 10.

G_{MI} maps generated $\mathbf{h}_c$ into realistic $\hat{\mathbf{x}}_c$. Because $\mathcal{L}_{G_{CG}}$ is not enough to stably generate $\mathbf{h}_c$, we add the loss of D_{MI} from $\hat{\mathbf{x}}_c$. The label of $(\hat{\mathbf{x}}_c, \mathbf{y}_c)$ for D_{MI} is a $(d+1)$-dimensional zero vector.

In addition, the cross-entropy of $(\hat{\mathbf{x}}_c, \mathbf{y}_c)$ calculated from the prediction of C is also added to the loss function of G_{CG} in order to generate the data that is conditioned on the target class stably as follows:

$$\mathcal{L}_{\text{CE}}(\hat{\mathbf{x}}_c, \mathbf{y}_c) = -\mathbb{E}_{\hat{\mathbf{x}}_c|\mathbf{y}_c} \left[\sum_{k=1}^{n_c} \mathbf{y}_{c_k} \log(C(\hat{\mathbf{x}}_c)_k) \right] \tag{11}$$

where $C(\cdot)_k$ is the softmax output for the k-th class. Thus, G_{CG} is trained according to:

$$\min_{G_{CG}} \mathcal{L}_{D_{CG}} + \alpha_2 \mathcal{L}_{D_{MI}} + \alpha_3 \mathcal{L}_{\text{CE}}(\hat{\mathbf{x}}_c, y_c) \tag{12}$$

where α_2 and α_3 denotes hyperparameters and we set α_2 to 1 and α_3 to 0.01 in our experiments. Since the distribution of $\mathbf{h}_l$ moves according to the training of E, we set the number of update iteration of D_{CG} and G_{CG} per an update of E to be 10, so that $\mathbf{h}_c$ follows the distribution of $\mathbf{h}_l$ well.

3.3. Semi-supervised classification

3.3.1. Pseudo-labeling

We define semi-supervised learning as imputing missing labels by the pseudo-labeling technique, TripleGAN (Li et al., 2017). Semi-supervised learning is achieved by the interaction of C and D_{MI}. C generates a pseudo-label $\mathbf{y}_u$ of an unlabeled instance $\hat{\mathbf{x}}_u$, i.e, $\mathbf{y}_u$ is sampled from the classifier distribution $p_C(\mathbf{y}|\mathbf{x})$. Then, the data-label pair $(\hat{\mathbf{x}}_u, \mathbf{y}_u)$ enters D_{MI}. The last element of D_{MI} output determines whether the label is real or fake. C and D_{MI} are trained according to the following objective:

$$\begin{aligned}\min_C \max_{D_{MI}} & V_{\text{u}}(C, D_{MI}) \\ =& \mathbb{E}_{\mathbf{y}|\hat{\mathbf{x}} \sim p_{data}} [D_{MI}(\hat{\mathbf{x}}, \mathbf{y})_{d+1}] \\ &- \mathbb{E}_{\mathbf{y}_u|\hat{\mathbf{x}}_u \sim p_C} [D_{MI}(\hat{\mathbf{x}}_u, \mathbf{y}_u)_{d+1}]\end{aligned} \tag{13}$$

where p_{data} denotes the data distribution of y conditioned on $\hat{\mathbf{x}}$. If G_{MI} has learned the true data distribution, then we can postulate that p_{data} follows the true conditional distribution. We should note that the adversarial loss is identical to the loss function of WGAN between C and D_{MI}. Therefore, C plays a role as a label generator, and $D_{MI}(\cdot)_{d+1}$ acts as a label discriminator.

Through adversarial learning, we expect that the adversarial loss enhances the performance of C. It can be shown that C maximizing the adversarial loss $V_{\text{u}}(C, D_{MI})$ is equivalent to C minimizing the Kullback-Leibler (KL) divergence between $p_C(y_u|\mathbf{x}_u)$ and $p_{data}(y|\mathbf{x})$. Thus, the adversarial loss satisfies the output distribution matching (ODM) cost (Sutskever et al., 2015). Formally,

Theorem 2 *Let $\mathbb{P}_C$ be the distribution of C and $\mathbb{P}_{data}$ be the data distribution. Optimizing the adversarial loss $V_{\text{u}}(C, D_{MI})$ is equivalent to minimizing the Kullback-Leibler divergence between $\mathbb{P}_C$ and $\mathbb{P}_{data}$. Then, the adversarial loss for semi-supervised learning in HexaGAN satisfies the definition of the ODM cost.*

Proof of Theorem 2 is in Supplementary Materials.

By the characteristics of the ODM cost, the global optimum of supervised learning is also a global optimum of $V_{\text{u}}(C, D_{MI})$. Therefore, intuitively, $V_{\text{u}}(C, D_{MI})$ serves as a guide for finding the optimum point of the supervised loss.

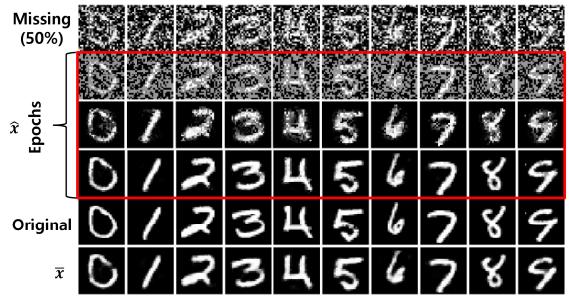


Figure 3. Imputation results with MNIST dataset. 1st row: MNIST images with 50% missing randomly as inputs of HexaGAN. 2nd∼4th rows (red box): images imputed by HexaGAN ($\hat{\mathbf{x}}$) at 1, 10, and 100 epochs. 5th row: Original images (no missing element). 6th row: images generated by G_{MI} for imputation ($\bar{\mathbf{x}}$).

3.3.2. Classification of HexaGAN

In order to train C, the two models E and G_{MI} impute the missing values of data instances $\hat{\mathbf{x}}_l$. G_{CG} produces hidden vectors $\mathbf{h}_c$ conditioned on the minority classes so that the number of data in the minority classes in each batch is equal to the number of data instances in the majority class of each batch, and G_{MI} generates class-conditional data $\hat{\mathbf{x}}_c$. Then, the cross-entropy between $\hat{\mathbf{x}}_{l,c} = \hat{\mathbf{x}}_l \cup \hat{\mathbf{x}}_c$ and $\mathbf{y}_{l,c} = \mathbf{y}_l \cup \mathbf{y}_c$ is calculated to train C. Unlabeled data $\hat{\mathbf{x}}_u$ is used to optimize V_{u}, the loss for pseudo-labeling, thereby training a more robust classifier.

Therefore, C is trained according to:

$$\min_C \mathcal{L}_{\mathrm{CE}}(\hat{\mathbf{x}}_{l,c}, \mathbf{y}_{l,c}) + \alpha_4 V_{\mathrm{u}} \tag{14}$$

where we used 0.1 for α_4 in our experiments.

4. Experiments

We present here the performance of the proposed method. We used datasets from the UCI machiene learning repository (Dheeru & Karra Taniskidou, 2017), including real world datasets (breast, credit, wine) and a synthetic dataset (madelon). We also used a handwritten digit dataset (MNIST). First, we show the imputation performance of HexaGAN. Then, we conduct experiments showing the quality of conditional generation using our framework. Finally, we present the classification performance of our proposed model assuming the problems in real world classification.

We basically assume 20% missingness (MCAR) in the elements and labels of the UCI dataset, and 50% in the elements of the MNIST dataset to cause missing data and missing label problems. Every element was scaled to a range of [0,1]. We repeated each experiment 10 times and used 5-fold cross validation. We calculated the root mean square error (RMSE) between the imputed value and the true value as the performance metric for missing data imputation. As a performance metric for classification, we used the F1-score, which is the harmonic mean of the precision and recall.

Table 1. Performance comparison with other imputation methods (RMSE)

Method	Breast	Credit	Wine	Madelon	MNIST
Zeros	0.2699	0.2283	0.4213	0.5156	0.3319
Matrix	0.0976	0.1277	0.1772	0.1456	0.2540
K-NN	0.0872	0.1128	0.1695	0.1530	0.2267
MICE	0.0842	0.1073	0.1708	0.1479	0.2576
Autoencoder	0.0875	0.1073	0.1481	0.1426	0.1506
GAIN	0.0878	0.1059	0.1406	0.1426	0.1481
HexaGAN	**0.0769**	**0.1022**	**0.1372**	**0.1418**	**0.1452**

We analyzed the learning curve and found that the modified zero-centered gradient and RMSprop promote stable HexaGAN. The details are described in Supplementary Materials. The architecture of networks also can be found in Supplementary Materials.

4.1. Imputation performance

4.1.1. Comparison with real world datasets

We used UCI datasets and the MNIST dataset to evaluate imputation performance. Table 1 shows the comparison of imputation methods. We present here the imputation performance of zero imputation, matrix completion, k-nearest neighbors, MICE, autoencoder, GAIN, and HexaGAN. In our experiments, we observed that HexaGAN outperforms the state-of-the-art methods on all datasets (up to a 14% improvement). Two deep generative models, GAIN and HexaGAN, both use reconstruction loss of autoencoder. GAIN shows same or lower performance than autoencoder in certain datasets, whereas HexaGAN consistently outperforms autoencoder in all datasets. This shows that the novel adversarial loss boosts the imputation performance.

4.1.2. Qualitative analysis

Figure 3 visualizes the imputation performance with the MNIST dataset. Since MNIST is an image dataset, we designed our framework with convolutional and deconvolutional neural networks. The details of the convolutional architecture are presented in Supplementary Materials. The first row of Figure 3 shows MNIST data with 50% missing as the input for HexaGAN. The next three rows show $\hat{\mathbf{x}}$ after 1, 10, and 100 epochs, and it can be seen that higher quality imputed data is generated as the number of epochs increases. The next row presents the original data with no missing value, and the last row shows $\bar{\mathbf{x}}$ generated by G_{MI}. This suggests that the proposed method imputes missing values with very high-quality data. The RMSE value using the convolutional architecture is 0.0914.

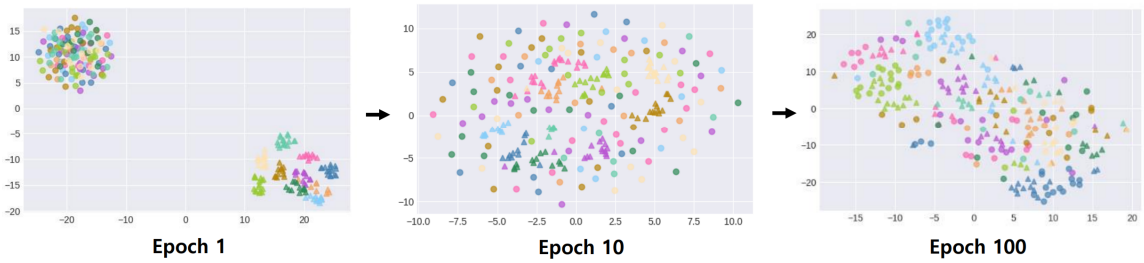


Figure 4. tSNE analysis with MNIST dataset at 1, 10, and 100 epochs. The circles stand for $\mathbf{h}_l$ (hidden vectors from E). The triangles denotes $\mathbf{h}_c$ (hidden vectors from G_{CG}). Different colors represent different class labels.

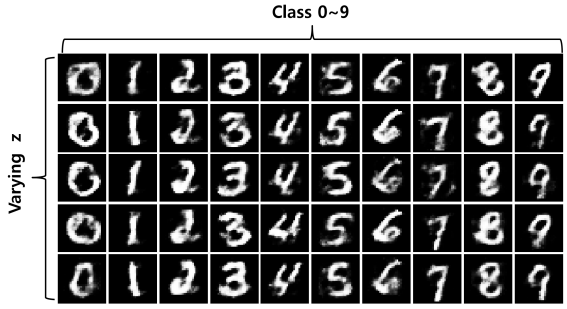


Figure 5. Class-conditional generation results with MNIST dataset. Each row visualizes generated images conditioned on 0∼9. Each column shows images generated by all different $\mathbf{z}$s.

4.2. Conditional generation performance

4.2.1. TSNE ANALYSIS

We used tSNE (Maaten & Hinton, 2008) to analyze $\mathbf{h}_l$ generated by E and $\mathbf{h}_c$ generated by G_{CG}. Figure 3 shows the changes of $\mathbf{h}_l$ (circle) and $\mathbf{h}_c$ (triangle) according to the iteration. Each color stands for a class label. At epoch 1, $\mathbf{h}_l$ and $\mathbf{h}_c$ have very different distributions, and form respective clusters. At epoch 10, the cluster of $\mathbf{h}_c$ is overlapped by the cluster of $\mathbf{h}_l$. At epoch 100, E learns the manifold of the hidden representation, so that $\mathbf{h}_l$ is gathered by class and $\mathbf{h}_c$ follows the distribution of $\mathbf{h}_l$ well. That is, G_{CG} creates high-quality $\mathbf{h}_c$ that is conditioned on a class label. The complete version of tSNE analysis is given in Supplementary Materials.

4.2.2. QUALITATIVE ANALYSIS

In order to evaluate the performance of conditional generation, we used the same architecture in Section 4.1.2, and generated synthetic MNIST data conditioned on 10 class labels. In Figure 5, we present the generated MNIST images. Each row represents the results of conditioning the class labels $0 \sim 9$, and each column represents the results of changing the noise vector $\mathbf{z}$. It can be seen that G_{CG} and G_{MI} produce realistic images of digits, and various image shapes are generated according to $\mathbf{z}$. Images conditioned on 9 in the second and fifth rows look like 7. This can be interpreted as a phenomenon in which hidden variables for 9 and 7 are placed in adjacent areas on the manifold of the hidden space.

4.3. Classification performance

The proposed method works without any problem for multi-class classification, but for the convenience of the report, we tested only binary classifications. The breast and credit datasets are imbalanced with a large number of negative samples. The wine dataset has three classes, and it was tested by binarizing the label 1 as negative, and labels 2 and 3 as positive to calculate an F1-score. The wine dataset was imbalanced with a large number of positive samples. Madelon is a synthetic dataset that randomly assigns binary labels to 32 clusters on 32 vertices of a 5-dimensional hypercube. Madelon is a balanced dataset.

4.3.1. ABLATION STUDY

The components affecting the classification performance of HexaGAN are G_{MI} to fill in missing data, G_{CG} to perform conditional generation, and $D_{MI}(\cdot)_{d+1}$ to enable semi-supervised learning. Table 2 compares the classification performance depending on the removal of these components. In the case of MLP, which is equivalent to HexaGAN without any of the three components, missing data were filled in with values sampled uniformly from [0,1].

As a result, MLP shows the worst performance. When HexaGAN contains G_{CG} (from the second row to the fourth row), the biggest performance improvement is shown in the credit data which is the most imbalanced. The more components included in HexaGAN, the higher the classification performance obtained. HexaGAN with all components shows the highest performance on every dataset. Our delicately devised architecture improves classification performance up to 36%. It offers the advantage that any classifier that is state-of-the-art in a controlled environment can be plugged into the proposed framework and perform at its highest capacity.

Table 2. Ablation study of HexaGAN (F1-score)

Method	Breast	Credit	Wine	Madelon
MLP (HexaGAN w/o G_{MI} & G_{CG} & $D_{MI_{d+1}}$)	0.9171 ± 0.0101	0.3404 ± 0.0080	0.9368 ± 0.0040	0.6619 ± 0.0017
HexaGAN w/o G_{CG} & $D_{MI_{d+1}}$	0.9725 ± 0.0042	0.4312 ± 0.0028	0.9724 ± 0.0065	0.6676 ± 0.0038
HexaGAN w/o G_{CG}	0.9729 ± 0.0007	0.4382 ± 0.0075	0.9738 ± 0.0135	0.6695 ± 0.0043
HexaGAN w/o $D_{MI_{d+1}}$	0.9750 ± 0.0030	0.4604 ± 0.0097	0.9770 ± 0.0037	0.6699 ± 0.0022
HexaGAN	**0.9762 ± 0.0021**	**0.4627 ± 0.0040**	**0.9814 ± 0.0059**	**0.6716 ± 0.0019**

Table 3. Classification performance (F1-score) comparison with other combinations of state-of-the-art methods

Method	Breast	Credit	Wine	Madelon
MICE + CS + TripleGAN	0.9417 ± 0.0044	0.3836 ± 0.0052	0.9704 ± 0.0043	0.6681 ± 0.0028
GAIN + CS + TripleGAN	0.9684 ± 0.0102	0.4076 ± 0.0038	0.9727 ± 0.0046	0.6690 ± 0.0027
MICE + SMOTE + TripleGAN	0.9434 ± 0.0060	0.4163 ± 0.0029	0.9756 ± 0.0037	0.6712 ± 0.0008
GAIN + SMOTE + TripleGAN	0.9672 ± 0.0063	0.4401 ± 0.0031	0.9735 ± 0.0063	0.6703 ± 0.0032
HexaGAN	**0.9762 ± 0.0021**	**0.4627 ± 0.0040**	**0.9814 ± 0.0059**	**0.6716 ± 0.0019**

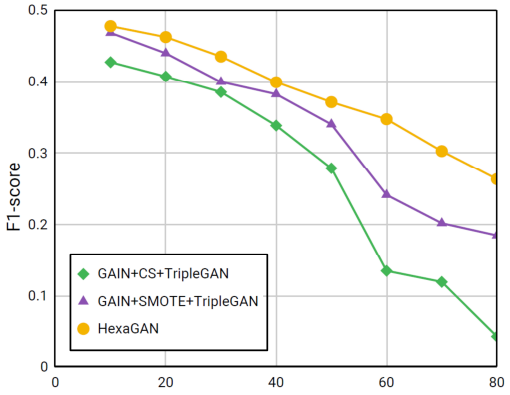


Figure 6. Classification performance (F1-score) comparison with respect to missing rate with credit dataset

4.3.2. Comparison with other combinations

In this experiment, we compared the classification performance of HexaGAN with combinations of state-of-the-art methods for the three data problems. For missing data imputation, we used MICE, which showed the best performance among machine learning based methods, and GAIN, which showed the best performance among deep generative models. For class imbalance, we used the cost sensitive loss (CS) and oversampled the minority class in a batch using SMOTE. We adopted the TripleGAN framework for semi-supervised learning. The classifier of TripleGAN used the same architecture as C of HexaGAN for a fair comparison.

As shown in Table 3, HexaGAN shows significantly better performance than the combinations of existing methods in cascading form (up to a 5% improvement). In addition, madelon dataset is balanced, thus, comparing HexaGAN without G_{CG} (the third row of Table 2) with the combination of MICE, CS, and TripleGAN (the first row of Table 3) and the combination of GAIN, CS, and TripleGAN (the second row of Table 3) demonstrates the classification performance with respect to imputation methods. As a result, we confirm that imputation method of HexaGAN guarantees better classification performance than other imputation methods.

4.3.3. Classification performance with respect to missing rate

The figure 6 compares the classification performance of HexaGAN with competitive combinations with various missing rates in credit dataset. We used the combination of GAIN, CS, and TripleGAN and the combitination of GAIN, SMOTE, and TripleGAN as benchmarks. As a result, HexaGAN outperforms the benchmarks with all missing rates. Moreover, our method shows a larger performance gap compared to the benchmarks with high missing rates. This means that HexaGAN works robustly in situations where only little information is available.

5. Conclusion

To interactively overcome the three main problems in real world classification (missing data, class imbalance, and missing label), we define the three problems in perspective with missing information. Then we propose a HexaGAN framework where six neural networks are actively correlated with others, and design several loss functions that maximize the utilization of any incomplete data. Our proposed method encourages more powerful performance in both imputation and classification than existing state-of-the-art methods. Moreover, HexaGAN is a one-stop solution that automatically solves the three problems commonly presented in real world classification. For future work, we plan to extend HexaGAN to time series datasets such as electronic health records.

References

Abbasnejad, M. E., Dick, A., and van den Hengel, A. Infinite variational autoencoder for semi-supervised learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 781–790. IEEE, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Beaulieu-Jones, B. K., Greene, C. S., et al. Semi-supervised learning of the electronic health record for phenotype stratification. *Journal of biomedical informatics*, 64:168–178, 2016.

Buuren, S. v. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pp. 1–68, 2010.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 15, 2009.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. R. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pp. 6510–6520, 2017.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Elrahman, S. M. A. and Abraham, A. A review of class imbalance problem. *Journal of Network and Innovative Computing*, 1(2013):332–340, 2013.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pp. 529–536, 2005.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Hastie, T., Mazumder, R., Lee, J. D., and Zadeh, R. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16 (1):3367–3402, 2015.

He, H., Bai, Y., Garcia, E. A., and Li, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 1322–1328. IEEE, 2008.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hwang, U., Choi, S., and Yoon, S. Disease prediction from electronic health records using generative adversarial networks. *arXiv preprint arXiv:1711.04126*, 2017.

Khalilia, M., Chakraborty, S., and Popescu, M. Predicting disease risks from highly imbalanced data using random forest. *BMC medical informatics and decision making*, 11(1):51, 2011.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, (8): 30–37, 2009.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

Li, C., Xu, T., Zhu, J., and Zhang, B. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems 30*, pp. 4088–4098. 2017.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Mescheder, L., Geiger, A., and Nowozin, S. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pp. 3478–3487, 2018.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Rubin, D. B. Inference and missing data. *Biometrika*, 63(3): 581–592, 1976.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shang, C., Palmer, A., Sun, J., Chen, K.-S., Lu, J., and Bi, J. Vigan: Missing view imputation with generative adversarial networks. In *Big Data (Big Data), 2017 IEEE International Conference on*, pp. 766–775. IEEE, 2017.

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

Sutskever, I., Jozefowicz, R., Gregor, K., Rezende, D., Lillicrap, T., and Vinyals, O. Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*, 2015.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

Turian, J., Ratinov, L., and Bengio, Y. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394. Association for Computational Linguistics, 2010.

Van Buuren, S. *Flexible imputation of missing data*. Chapman and Hall/CRC, 2018.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Wang, F. and Zhang, C. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008.

Yoon, J., Jordon, J., and van der Schaar, M. Gain: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5689–5698. PMLR, 2018.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

Supplementary Materials
HexaGAN: Generative Adversarial Nets for Real World Classification

Uiwon Hwang [1] **Dahuin Jung** [1] **Sungroh Yoon** [1]

A. Global optimality of $p(\mathbf{x}|\mathbf{m}_i = 1) = p(\mathbf{x}|\mathbf{m}_i = 0)$ for HexaGAN

Proof of Theorem 1: Let $D_{MI}(\cdot)$ be $D(\cdot)$, and $G_{MI}(E(\cdot))$ be $G(\cdot)$ for convenience.

The min-max loss of HexaGAN for missing data imputation is given by:

$$
\begin{aligned}
V_{MI}(D,G) &= \mathbb{E}_{\mathbf{x},\mathbf{z},\mathbf{m}}\left[\mathbf{m}^T D(G(\tilde{\mathbf{x}}|\mathbf{m})) - (\mathbf{1}-\mathbf{m})^T D(G(\tilde{\mathbf{x}}|\mathbf{m}))\right] && (1)\\
&= \mathbb{E}_{\hat{\mathbf{x}},\mathbf{m}}\left[\mathbf{m}^T D(\hat{\mathbf{x}}) - (\mathbf{1}-\mathbf{m})^T D(\hat{\mathbf{x}})\right] && (2)\\
&= \int_{\hat{\mathcal{X}}} \sum_{\mathbf{m}\in\{0,1\}^d} \left(\mathbf{m}^T D(\mathbf{x}) - (1-\mathbf{m})^T D(\mathbf{x})\right) p(\mathbf{x}|\mathbf{m}) d\mathbf{x} && (3)\\
&= \int_{\hat{\mathcal{X}}} \sum_{i=1}^{d} \left(\sum_{\mathbf{m}\in\{\mathbf{m}|m_i=1\}} D(\mathbf{x})_i - \sum_{\mathbf{m}\in\{\mathbf{m}|m_i=0\}} D(\mathbf{x})_i \right) p(\mathbf{x}|\mathbf{m}) d\mathbf{x} && (4)\\
&= \int_{\hat{\mathcal{X}}} \sum_{i=1}^{d} \left(D(\mathbf{x})_i \sum_{\mathbf{m}\in\{\mathbf{m}|m_i=1\}} p(\mathbf{x}|\mathbf{m}) - D(\mathbf{x})_i \sum_{\mathbf{m}\in\{\mathbf{m}|m_i=0\}} p(\mathbf{x}|\mathbf{m}) \right) d\mathbf{x} && (5)\\
&= \int_{\hat{\mathcal{X}}} \sum_{i=1}^{d} D(\mathbf{x})_i p(\mathbf{x}|m_i=1) - D(\mathbf{x})_i p(\mathbf{x}|m_i=0) d\mathbf{x} && (6)\\
&= \int_{\hat{\mathcal{X}}} \sum_{i=1}^{d} \left(p(\mathbf{x}|m_i=1) - p(\mathbf{x}|m_i=0)\right) D(\mathbf{x})_i d\mathbf{x} && (7)\\
& && (8)
\end{aligned}
$$

For a fixed G, the optimal discriminator $D(\mathbf{x})_i$ which maximizes $V_{MI}(D,G)$ is such that:

$$
D^*_G(\mathbf{x})_i = \begin{cases} 1, & \text{if } p(\mathbf{x}|m_i=1) \geq p(\mathbf{x}|m_i=0) \\ 0, & \text{otherwise} \end{cases} \tag{9}
$$

Plugging D^*_G back into Equation 7, we get:

$$
\begin{aligned}
V_{MI}(D^*_G, G) &= \int_{\hat{\mathcal{X}}} \sum_{i=1}^{d} \left(p(\mathbf{x}|m_i=1) - p(\mathbf{x}|m_i=0)\right) D^*_G(\mathbf{x})_i d\mathbf{x} && (10)\\
&= \sum_{i=1}^{d} \int_{\{\mathbf{x}|p(\mathbf{x}|m_i=1)\geq p(\mathbf{x}|m_i=0)\}} \left(p(\mathbf{x}|m_i=1) - p(\mathbf{x}|m_i=0)\right) d\mathbf{x} && (11)
\end{aligned}
$$

[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea. Correspondence to: Sungroh Yoon <sryoon@snu.ac.kr>.

Let $\mathcal{X} = \{\mathbf{x}|p(\mathbf{x}|m_i = 1) \geq p(\mathbf{x}|m_i = 0)\}$. To minimize Equation 11, we need to set $p(\mathbf{x}|m_i = 1) = p(\mathbf{x}|m_i = 0)$ for $\mathbf{x} \in \mathcal{X}$. Then, since both probability density functions should integrate to 1,

$$\int_{\mathcal{X}^c} p(\mathbf{x}|m_i = 1)d\mathbf{x} = \int_{\mathcal{X}^c} p(\mathbf{x}|m_i = 0)d\mathbf{x} \tag{12}$$

However, this is a contradiction because $p(\mathbf{x}|m_i = 1) < p(\mathbf{x}|m_i = 0)$ for $\mathbf{x} \in \mathcal{X}^c$, unless $\lambda(X^c) = 0$ where λ is the Lebesgue measure. This finishes the proof.

B. Optimization of components for imputation

From Equation 6,

$$V_{MI}(D, G)_i = \int_{\hat{\mathcal{X}}} p(\mathbf{x}|m_i = 1)D(\mathbf{x})_i - p(\mathbf{x}|m_i = 0)D(\mathbf{x})_i d\mathbf{x} \tag{13}$$

$$= \mathbb{E}_{\tilde{\mathbf{x}},\mathbf{z},\mathbf{m}}\left[m_i \cdot D(G(\tilde{\mathbf{x}}|\mathbf{m}))_i\right] - \mathbb{E}_{\tilde{\mathbf{x}},\mathbf{z},\mathbf{m}}\left[(1 - m_i) \cdot D(G(\tilde{\mathbf{x}}|\mathbf{m}))_i\right] \tag{14}$$

G is then trained according to $\min_G \sum_{i=1}^{d} V_{MI}(D, G)_i$, and D is trained according to $\max_D \sum_{i=1}^{d} V_{MI}(D, G)_i$.

C. Relation between pseudo-labeling and the ODM cost

Proof of Theorem 2: Optimizing the adversarial loss function $V_{\mathrm{u}}(C, D_{MI})$ minimizes the Wasserstein distance between $\mathbb{P}_c$ and $\mathbb{P}_{data}$.

Since the Wasserstein distance $(W(p, q))$ is a much weaker metric than Kullback-Leibler divergence $(KL(p||q))$ (Arjovsky et al., 2017), the following proposition holds

$$W(\mathbb{P}_C, \mathbb{P}_{data}) \to 0 \;\; \Rightarrow \;\; KL(\mathbb{P}_C||\mathbb{P}_{data}) \to 0 \tag{15}$$

This means that minimizing the Wasserstein distance between $\mathbb{P}_C$ and $\mathbb{P}_{data}$ is minimizing the Kullback-Leibler divergence between $\mathbb{P}_C$ and $\mathbb{P}_{data}$. Therefore, the adversarial loss of D_{MI} and C satisfy the definition of the output distribution matching (ODM) cost function, concluding the proof.

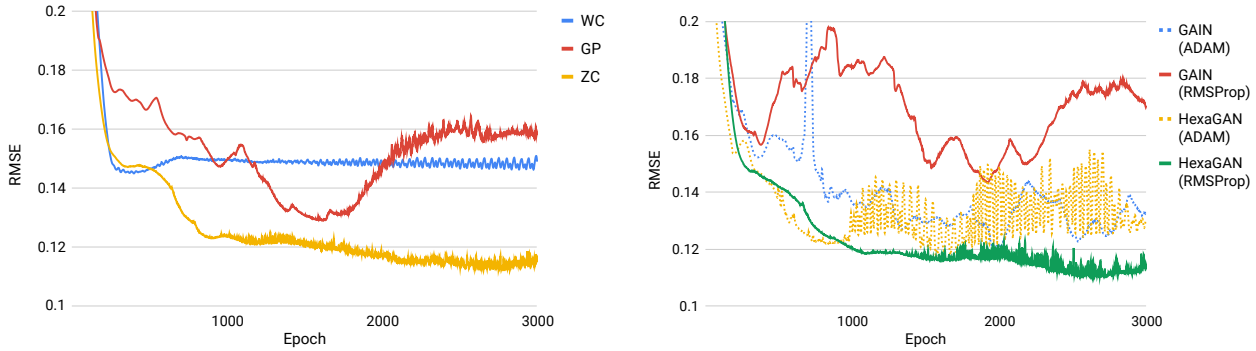
D. HexaGAN architecture

Excluding the experiments in Sections 4.1.2 and 4.2, all six components used an architecture with three fully-connected layers. The number of hidden units in each layer is d, $d/2$, and d. As an activation function, we use the rectified linear unit (ReLU) function for all hidden layers and the output layer of E and G_{CG}, the sigmoid function for the output layer of G_{MI}, D_{CG}, no activation function for the output layer of D_{MI}, and the softmax function for the output layer of C.

Table 1 describes the network architectures used in Sections 4.1.2 and 4.2. In the table, FC(n) denotes a fully-connected layer with n output units. Conv(n, $k \times k$, s) denotes a convolutional network with n feature maps, filter size $k \times k$, and stride s. Deconv(n, $k \times k$, s) denotes a deconvolutional network with n feature maps, filter size $k \times k$, and stride s.

Table 1. Convolutional neural network architectures used for MNIST dataset

G_{CG}	D_{CG}	E	G_{MI}	D_{MI}	C
FC(512)	FC(1024)	Conv(32, 5×5, 2)	Deconv(64, 5×5, 2)	Conv(32, 5×5, 2)	Conv(32, 5×5, 2)
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
FC(1024)	FC(512)	Conv(64, 5×5, 2)	Deconv(32, 5×5, 2)	Conv(64, 5×5, 2)	Conv(64, 5×5, 2)
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
FC(2048)	FC(1)	Conv(128, 5×5, 2)	Deconv(1, 5×5, 2)	Conv(128, 5×5, 2)	Conv(128, 5×5, 2)
ReLU	Sigmoid	ReLU	ReLU	ReLU	ReLU
			FC(784)	FC(785)	FC(10)
			Sigmoid	Sigmoid	Softmax



(a) Comparison of gradient penalty

(b) Comparison of adversarial loss and optimizer

Figure 1. Learning curve comparison for optimal GAN imputation method

E. Learning curve analysis

On the breast dataset, we measured the RMSE to evaluate the performance of the proposed adversarial losses ($\mathcal{L}_{D_{MI}}, \mathcal{L}_{G_{MI}}$). We compared learning curves of weight clipping (WC) proposed by Arjovsky et al. (2017), modified gradient penalty (GP) of Gulrajani et al. (2017), and modified zero-centered gradient penalty (ZC, ours) to determine the most appropriate gradient penalty for our framework. As shown in Figure 1(a), ZC shows stable and good performance (small RMSE). In Figure 1(b), we exclude $\mathcal{L}_{\text{recon}}$ from the loss of E and G_{MI} and plot learning curves to accurately compare the adversarial losses of GAIN and HexaGAN. We also compare the two optimizers of ADAM (Kingma & Ba, 2014) and RMSProp (Tieleman & Hinton, 2012). In our experiment, it shows that RMSProp is a more stable optimizer than ADAM, and HexaGAN shows more stable and better imputation performance than GAIN.

F. Imputation performance with respect to missing rate

We measured the imputation performance of HexaGAN with various missing rates in credit dataset. To compare performance with competitive benchmarks, we used MICE, which is a state-of-the-art machine learning algorithm, and GAIN, which is a state-of-the-art deep generative model. As seen in Figure 2, HexaGAN shows the best performance with all missing rates except 50%.

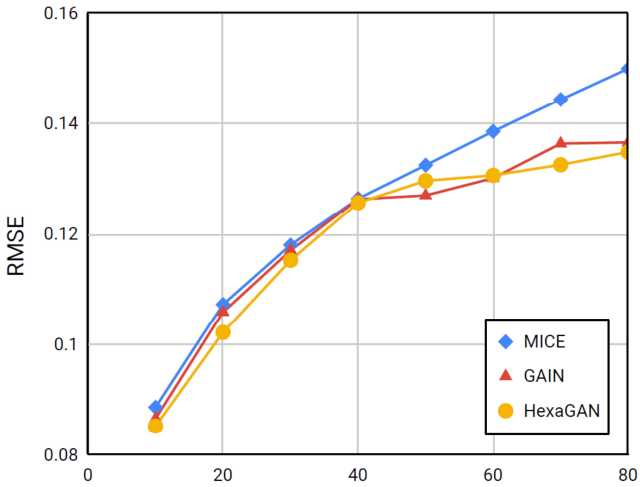


Figure 2. Imputation performance (RMSE) comparison with respect to missing rate with credit dataset

G. tSNE analysis on conditional generation of HexaGAN

Figure 3 is the complete version of tSNE analysis in Section 4.2.1. The tSNE plot below shows an analysis of the manifold of the hidden space. We confirm that the synthetic data around the original data looks similar to the original data. Therefore, it can be seen that E learns the data manifold well in hidden space.

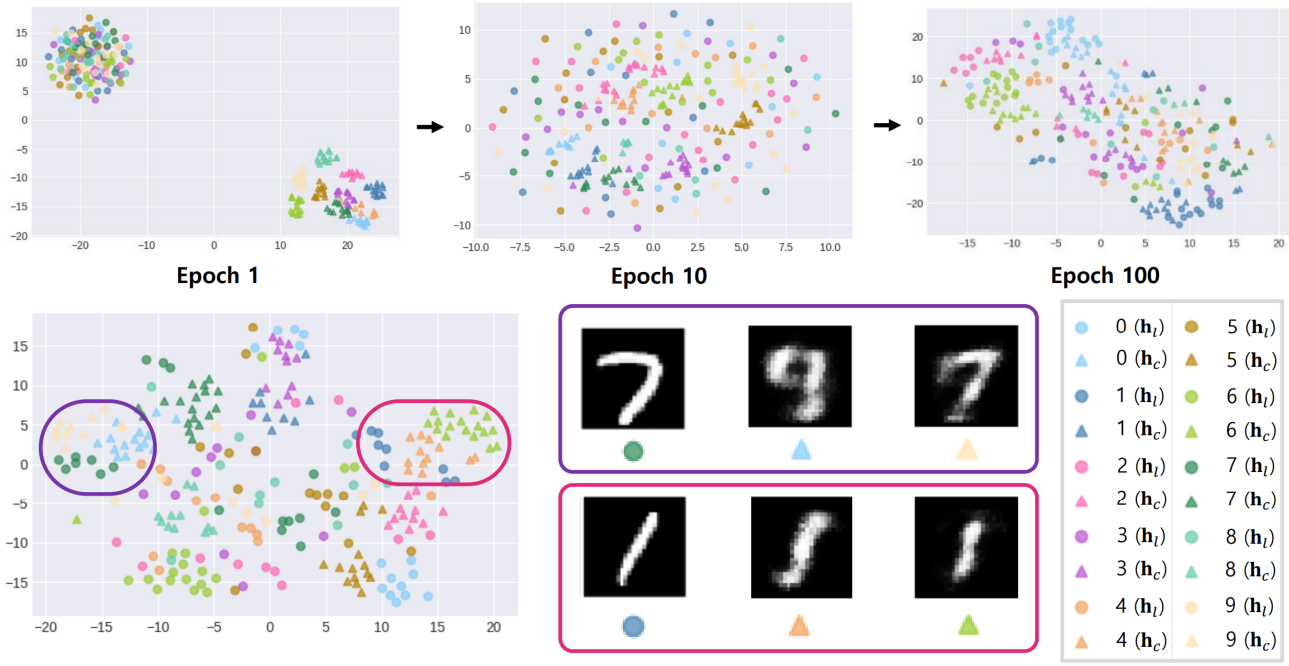


Figure 3. tSNE analysis with MNIST dataset

References in Supplementary Materials

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.