HexaGAN: Generative Adversarial Nets for Real World Classification

Uiwon Hwang 1  Dahuin Jung 1  Sungroh Yoon 1 ,2

Abstract
Most deep learning classification studies assume clean data. However, when dealing with the real world data, we encounter three problems such as 1) missing data, 2) class imbalance, and 3) missing label problems. These problems undermine the performance of a classifier. Various preprocessing techniques have been proposed to mitigate one of these problems, but an algorithm that assumes and resolves all three problems together has not been proposed yet. In this paper, we propose HexaGAN, a generative adversarial network framework that shows promising classification performance for all three problems. We interpret the three problems from a single perspective to solve them jointly. To enable this, the framework consists of six components, which interact with each other. We also devise novel loss functions corresponding to the architecture. The designed loss functions allow us to achieve state-of-the-art imputation performance, with up to a 14% improvement, and to generate 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 performance of combinations of state-of-the-art methods.

1. Introduction
As deep learning models have achieved super-human performance in image classification tasks (He et al., 2016), there have been increasing attempts to apply deep learning models to more complicated 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 are often dirty, which means that the elements and labels are missing, or there is an imbalance between different classes of data. This prevents a classifier from being fully effective, and thus a preprocessing phase is required. Despite a considerable amount of research, no preprocessing technique has been proposed to address these three problems concurrently. Therefore, we first propose a framework which deals robustly with dirty data.

The types of data which are typically bedeviled with missing information include the user data employed in recommender systems (Koren et al., 2009), and in electronic health records (Miotto et al., 2016) utilizing deep learning based classifiers. Rubin (1976) identifies three main types of missing data: 1) Data are missing completely at random (MCAR). This type of missing data has no pattern which can be correlated with any other variable, whether observed or not. 2)
Data are missing at random (MAR). In this case, the pattern of missing data can be correlated with one or more observed variables. 3) Data are missing (but) not at random (MNAR). The pattern of this type of missing data can be related to both observed and unobserved variables. In this paper, we are concerned with MCAR data. The replacement of missing information within data is called imputation (Van Buuren, 2018). Imputation techniques include matrix completion (Hastie et al., 2015), k-nearest neighbors (Troyanskaya et al., 2001), multivariate imputation by chained equations (MICE) (Buuren & Groothuis-Oudshoorn, 2010), denoising autoencoders (Vincent et al., 2008), and methods based on generative adversarial networks (GAN) (Yoon et al., 2018; Shang et al., 2017). Poor or inappropriate imputation can mislead deep learning based techniques due to the wrong data distribution.

Many real world datasets such as those related to anomaly detection (Chandola et al., 2009) and disease prediction (Khalilia et al., 2011) involve poorly balanced classes. The class imbalance problem can be overcome by techniques such as the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002) and adaptive synthetic (ADASYN) sampling (He et al., 2008). However, oversampling from the entire data distribution requires a large amount of memory. Cost sensitive loss (Sun et al., 2007) is also used to solve the class imbalance problem by differentiating cost weights to each class. However, cost sensitive loss tends to overfit to the minority classes (Elrahman & Abraham, 2013). We overcome the class imbalance problem by training a deep generative model to follow the true data distribution, and then generate samples of minority classes for each batch. This requires conditional generation, which we regard as imputation, as shown in Figure 1, in which entire elements are imputed according to the appropriate class label.

In deep learning, the amount of labeled training data has a significant impact on the performance. Insufficiency of labeled data is referred to as the missing label problem. It is encountered in real world applications such as natural language models (Turian et al., 2010) or healthcare systems (Beaulieu-Jones et al., 2016), where the cost of labeling is expensive. Related researchers have proposed semi-supervised methods by which to leverage unlabeled data. Semi-supervised learning is designed to make the best use of unlabeled data, using 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 raw unlabeled data in training the generative model (Kingma et al., 2014; Abbasnejad et al., 2018).
HexaGAN: Generative Adversarial Nets for Real World Classification

As depicted in Figure 1, we define the missing data, class imbalance, and missing label problems in terms of imputation. Based on insight concerning the imputation, we find out that networks used 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. In order to deal with the missing label problem, we use semi-supervised learning, in which a classifier generates a synthetic class label for unlabeled data and a discriminator distinguishes fake 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.

- To implement real world datasets into existing classifiers, we must apply suitable preprocessing techniques to the datasets. However, our framework is simple to use and works automatically when the absence of data elements and labels is indicated (m and $m_y$, See Section 3).

- We devise a combination of six components and the corresponding cost functions. More specifically, we propose a novel adversarial loss function and gradient penalty for element-wise imputation, confirming that our imputation performance produces 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

Generative models, which include GANs (Goodfellow et al., 2014), are capable of generating high-quality synthetic data for many applications. Although GANs are the most advanced than other techniques, model training can be unstable. Many studies have tried to stabilize GANs. Among them, Arjovsky et al. (2017) proposed the Wasserstein GAN (WGAN), which has a smoother gradient by introducing the Wasserstein (Earth Mover) distance. Several gradient penalties have also been proposed (Gulrajani et al., 2017; Mescheder et al., 2018) to make WGAN training more stable. In this paper, we modify the 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 model.

GAIN (Yoon et al., 2018) is the first method to use a GAN for imputing MCAR data. The typical discriminator predicts whether each instance is real or fake. However, this task is difficult if all instances have missing data. Instead, GAIN labels each element of an instance as missing or not, so that the discriminator can discriminate between real and fake elements. Our imputation method shares some similarity with GAIN because both methods label elements 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 appears to be unstable. However, HexaGAN provides a stable imputation performance, and usability by including class-conditional generation to address the class imbalance problem, and through the use of semi-supervised learning.

TripleGAN (Li et al., 2017) is a GAN for semi-supervised learning in which a classifier, a generator, and a discriminator interact. The classifier creates pseudo-labels for unlabeled data, and image-label pairs are then passed to 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, as illustrated in Figure 2:

- $E$: the encoder, that transfers both labeled and unlabeled instances into the hidden space.
- $G_{MI}$: a generator that imputes missing data.
- $D_{MI}$: a discriminator for missing imputation, that distinguishes between missing and non-missing elements and labels.
- $G_{CG}$: a generator that creates conditional hidden vectors $h_c$.
- $D_{CG}$: a discriminator for conditional generation, that
determines whether a hidden vector is from the dataset or has been created by \( G_{CG} \).

- \( C \): the classifier, that estimates class labels. This also works as the label generator.

HexaGAN operates on datasets containing instances \( x_1, \ldots, x_n \in \mathbb{R}^d \), where \( n \) is the number of instances and \( d \) is the number of elements in an instance. The \( i \)-th element in a single instance \( x_i \) is a scalar, and some of these elements may be missing. The first \( n_l \) instances are labeled data, and the remaining \( n - n_l \) instances are unlabeled data. There are class labels \( m_1, \ldots, m_n \in \mathbb{R}^c \), where \( c \) is the number of classes. Boolean vectors \( m_1, \ldots, m_n \in \mathbb{R}^d \) indicate whether each element in an instance is missing or not. If \( m_i = 0 \) (the \( i \)-th element of a vector \( m \)) is 0, \( x_i \) is missing. So, \( m \) indicates whether an instance has a label or not. If \( m_i = 0 \), the label is missing. Thus, labeled instances exist as a set of \( D_l = \{ (x_1, y_1, m_1, m_1^t = 1) \}_{j=1}^{n_l} \), and unlabeled instances exist as a set of \( D_u = \{ (x_1, m_1, m_1^t = 0) \}_{j=n_l+1}^n \).

### 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 HexaGAN, missing data imputation is performed by \( E, G_{MI} \), and \( D_{MI} \). An instance received by \( D_{MI} \) is not labeled as real or fake, but each element is labeled as real (non-missing) or fake (missing).

From now on, we omit the superscript for a clearer explanation (i.e., \( x_i = x_i \)). First, we make a noise vector \( z \in \mathbb{R}^d \) with the same dimension as an input instance \( x \in (x_l \cup x_u) \) by sampling from a uniform distribution \( U(0, 1) \). We replace the missing elements in the instance with elements of \( z \) to generate \( \hat{x} \):

\[
\hat{x} = m \odot x + (1 - m) \odot z
\]  

(1)

where \( \odot \) is element-wise multiplication. The objective of our framework is to sample the patterns stored in the model that are the most suitable replacements for the missing data (i.e., to generate samples which follow \( p(x|m) \)). Then, \( \hat{x} \) is concatenated with \( m \), and from the pair \((\hat{x}, m)\), the encoder \( E \) generates a hidden variable \( h = E(\hat{x}, m) \) in the hidden space, which has the dimension \( d_h \).

The \( G_{MI} \) receives \( h \) and generates \( \hat{x} = G_{MI}(h) \). The missing elements in the input instance are imputed with the generated values, resulting in \( \hat{x} \) as follows:

\[
\hat{x} = m \odot x + (1 - m) \odot \hat{x}
\]  

(2)

The \( D_{MI} \) now determines whether each element of the pair \((\hat{x}, y)\) is real or fake. The label for \( \hat{x} \) is \( m \). The \( D_{MI} \) calculates the adversarial losses by determining whether the missingness is correctly predicted for each element, which is then used to train \( E, G_{MI} \), and \( D_{MI} \). The adversarial loss \( \mathcal{L}_{G_{MI}} \) which is used to train \( E \) and \( G_{MI} \), and \( \mathcal{L}_{D_{MI}} \) which is used to train \( D_{MI} \) can be expressed as follows:

\[
\mathcal{L}_{G_{MI}} = - \sum_{i=1}^d E_{x,y,m} [(1 - m_i) \cdot D_{MI}(\hat{x}, y)_i]\]  

(3)

\[
\mathcal{L}_{D_{MI}} = \sum_{i=1}^d E_{x,y,m} [(1 - m_i) \cdot D_{MI}(\hat{x}, y)_i]
\]  

(4)

where \( D_{MI} \) 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 \( p(x|m_i = 0) \) is a global optimum for the min-max game of \( G_{MI} \) and \( D_{MI} \), if and only if \( p(x|m_i = 1) = p(x|m_i = 0) \) for all \( x \in \mathbb{R}^d \), 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 \( G_{MI} \) to exploit the information of non-missing elements, as follows:

\[
\mathcal{L}_{recon} = E_{x|m} \left[ \sum_{i=1}^d m_i (x_i - \hat{x}_i)^2 \right]
\]  

(5)

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

$$L_{GP_{MI}} = \sum_{i=1}^{d} \mathbb{E}_{p_D(x_i)} \left[ \left\| \nabla_{\hat{x}} D_{MI}(\hat{x}_i) \right\|_2^2 \right]$$  \hspace{1cm} (6)

We define $\hat{x}$ in $p_{D}(x_i)$ as data with $m_i$ is 1 (i.e., $p_{D}(x_i) = \{ \hat{x} \mid m_i = 1 \}$). In other words, as suggested by Mescheder et al. (2018), we penalize $D_{MI}$ only for data wherein the $i$-th element is not missing (real) in a batch. This helps balance an adversarial relationship between the generator and discriminator by forcing the discriminator closer to Nash Equilibrium.

Therefore, missing data imputation and model training are performed as described in Algorithm 1. We used 10 for both hyperparameters $\lambda_1$ and $\alpha_1$ 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 $p(x|y)$).

Since we have $G_{MI}$, which is a generator for imputation, we can oversample data instances by feeding synthetic $\hat{h}$ into $G_{MI}$. Therefore, we introduce $G_{CG}$ to generate a hidden variable $h_c$ corresponding to the target class label $y_c$, i.e., we sample $h_c \sim p_{GCG}(h|y)$.

We also introduce $D_{CG}$ to distinguish pairs of generated hidden variables and target class labels $(h_c, y_c)$ (fake) from pairs of hidden variables for labeled data and corresponding class labels $(h_l, y_l)$ (real). $G_{CG}$ and $D_{CG}$ are trained with WGAN loss and zero-centered gradient penalty on $h_l$ as follows:

$$L_{CG} = -\mathbb{E}_{h_c \sim p_{GCG}(h_c|y_c)} [D_{CG}(h_c, y_c)]$$

$$L_{DC} = \mathbb{E}_{h_c \sim p_{GCG}(h_c|y_c)} [D_{CG}(h_c, y_c)] - \mathbb{E}_{h_c \sim p_{GCG}(h_c|y_c)} [D_{CG}(h_c, y_c)]$$

$$L_{GP_{CG}} = \mathbb{E}_{h_c \sim p_{GCG}(h_c|y_c)} \left[ \left\| \nabla_{h_c} D_{CG}(h_c, y_c) \right\|_2^2 \right]$$

$G_{MI}$ maps generated $h_c$ into a realistic $\hat{x}_c$. Because $L_{CG}$ is not enough to stably generate $h_c$, we add the loss of $G_{MI}$ from $\hat{x}_c$. Since we defined conditional generation as imputation of all the elements, $G_{CG}$ and $D_{MI}$ are related adversarially. The label of $(\hat{x}_c, y_c)$ for $D_{MI}$ is a $(d + 1)$-dimensional zero vector.

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

$$L_{CE}(\hat{x}_c, y_c) = -\mathbb{E}_{\hat{x}_c|y_c} \left[ \sum_{k=1}^{n_k} y_{ck} \log(C(\hat{x}_c)_k) \right]$$  \hspace{1cm} (10)

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

$$\min_{D_{CG}} L_{DC} + \lambda_2 L_{GP_{CG}}$$  \hspace{1cm} (11)

$$\min_{G_{CG}} L_{CG} + \alpha_2 L_{GM1} + \alpha_3 L_{CE}(\hat{x}_c, y_c)$$  \hspace{1cm} (12)

where $\lambda_2$, $\alpha_2$, and $\alpha_3$ denote hyperparameters, and we set $\lambda_2$ to 10, $\alpha_2$ to 1, and $\alpha_3$ to 0.01 in our experiments. Since the distribution of $h_l$ moves according to the training of $E$, we set the number of update iterations of $D_{CG}$ and $G_{CG}$ per an update of $E$ to 10, so that $h_c$ follows the distribution of $h_l$ 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 $y_u$ of an unlabeled instance $\hat{x}_u$, i.e., $y_u$ is sampled from the classifier distribution $p_{C}(y|x)$. Then, the data-label pair $(\hat{x}_u, y_u)$ enters $D_{MI}$. The last element of the $D_{MI}$ output, $D_{MI}(\cdot)_{d+1}$, determines whether the label is real or fake. The label for pseudo-labeling is $m_u$. $C$ and $D_{MI}$ are trained according to the following loss functions:

$$L_C = -\mathbb{E}_{y_u|\hat{x}_u \sim p_{CG}} [D_{MI}(\hat{x}_u, y_u)]$$

$$L_{D_{MI}} = \mathbb{E}_{x_u|\hat{x}_u \sim p_{CG}} [D_{MI}(\hat{x}_u, y_u)]$$

where $p_{data}$ denotes the data distribution of $y$ conditioned on $\hat{x}$. $L_{D_{MI}^{d+1}}$ is added to the loss of $D_{MI}$, so that $i$ in Equation 4 expands from $d$ to $d + 1$. If $G_{MI}$ learns 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 $C$ minimizing the adversarial loss $L_C$ is equivalent to optimizing the output distribution matching (ODM) cost (Sutskever et al., 2015).

**Theorem 2** Optimizing the adversarial losses for $C$ and $D_{MI}(\cdot)_{d+1}$ imposes an unsupervised constraint on $C$. Then, the adversarial losses for semi-supervised learning in HexaGAN satisfy the definition of the ODM cost.

Proof of Theorem 2 is provided in Supplementary Materials.

According to the properties of the ODM cost, the global optimum of supervised learning is also a global optimum.
of semi-supervised learning. Therefore, intuitively, $L_C$ and $L_{MI}^{(C)}$ serve as guides for finding the optimum point of the supervised loss.

### 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 $\hat{x}_t$. $G_{CG}$ produces hidden vectors $h_x$ 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 $\hat{x}_c$. Then, the cross-entropy between $\hat{x}_{l,c} \in (\hat{x}_t \cup \hat{x}_c)$ and $y_{l,c} \in (y_t \cup y_c)$ is calculated to train $C$. Unlabeled data $\hat{x}_u$ is used to optimize $L_C$, the loss for pseudo-labeling, thereby training a more robust classifier.

Therefore, $C$ is trained according to:

$$
\min_C L_{CE}(\hat{x}_{l,c}, y_{l,c}) + \alpha_4 L_C
$$

where we used 0.1 for $\alpha_4$ in our experiments. The entire training procedure of HexaGAN is presented in Supplementary Materials.

## 4. Experiments

Here, we present the performance of the proposed method. We used datasets from the UCI machine learning repository (Dheeru & Karra Taniskidou, 2017), including real world datasets (breast, credit, wine) and a synthetic dataset (madelon). Detailed descriptions are presented in Supplementary Materials. We also used a handwritten digit dataset (MNIST). First, we show the imputation performance of HexaGAN. Then, we show 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. As the performance metric, we calculated the root mean square error (RMSE) for missing data imputation and the F1-score for classification. We analyzed the learning curve and found that the modified zero-centered gradient and RMSprop promote stability in HexaGAN. The details are described in Supplementary Materials. The architecture of HexaGAN can also 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 the imputation performance. Table 1 shows 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, use both the reconstruction loss and the adversarial loss. GAIN shows the same or lower performance than the autoencoder on certain datasets, whereas HexaGAN consistently outperforms the autoencoder on 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 HexaGAN with convolutional and deconvolutional neural networks. The first row of Figure 3 shows MNIST data with 50\% missing as the input for HexaGAN. The next three rows show $\bar{x}$ after 1, 10, and 100 epochs, and it can be seen that higher quality imputed data are generated as the number of epochs increases. The next row presents the original data with no missing value, and the last row shows $\bar{x}$ generated by $G_{MI}$. This suggests that the proposed method imputes missing values with very high-quality data. The
Figure 4. tSNE analysis with the MNIST dataset at 1, 10, and 100 epochs. The circles stand for \( h_l \) (hidden vectors from \( E \)). The triangles denote \( h_c \) (hidden vectors from \( G_{CG} \)). Different colors represent different class labels.

Figure 5. Class-conditional generation results with the MNIST dataset. Each row visualizes generated images conditioned on 0~9. Each column shows images generated by all different \( z \)s.

RMSE value using the convolutional architecture is 0.0914.

4.2. Conditional generation performance

4.2.1. TSNE analysis

We used tSNE (Maaten & Hinton, 2008) to analyze \( h_l \) generated by \( E \) and \( h_c \) generated by \( G_{CG} \). Figure 4 shows the changes of \( h_l \) (circle) and \( h_c \) (triangle) according to the iteration. Each color stands for a class label. At epoch 1, \( h_l \) and \( h_c \) have very different distributions, and form respective clusters. At epoch 10, the cluster of \( h_c \) is overlapped by the cluster of \( h_l \). At epoch 100, \( E \) learns the manifold of the hidden representation, so that \( h_l \) is gathered by class and \( h_c \) follows the distribution of \( h_l \) well. That is, \( G_{CG} \) creates a high-quality \( h_c \) that is conditioned on a class label. The complete version of the tSNE analysis is given in Supplementary Materials.

4.2.2. Qualitative analysis

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

4.3. Classification performance

HexaGAN 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 balanced synthetic dataset that randomly assigns binary labels to 32 clusters on 32 vertices of a 5-dimensional hypercube.

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 these 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 the 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 the classification performance by 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 the...
Breast 0.9368 0.4627 0.4604 0.9725 0.9750 0.9770 0.9724 0.9750 0.9725
Wine 0.9814 0.3404 0.9738 0.6619 0.9171 0.4382 0.9729 0.0007 0.4312 0.9724 0.9750 0.9725
Credit 0.4382 0.9729 0.6716 0.4312 0.6699 0.6676 0.6695 0.0043 0.4604 0.9724 0.9750 0.9725

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, the 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) shows the classification performance with respect to imputation methods. We confirm that the imputation method of HexaGAN guarantees better classification performance than the other imputation methods.

### 4.3.2. COMPARISON WITH OTHER COMBINATIONS

In this experiment, we compared the classification performance of HexaGAN with those of combinations of state-of-the-art methods for the three 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 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, the 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) shows the classification performance with respect to imputation methods. We confirm that the imputation method of HexaGAN guarantees better classification performance than the other imputation methods.

### 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 from the perspective of missing information. Then, we propose a HexaGAN framework wherein 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.
Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) [No. 2018R1A2B3001628, No. 2014M3C9A3063541], Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI) funded by the Ministry of Health & Welfare [HI15C3224], AIR Lab (AI Research Lab) in Hyundai Motor Company through HMC-SNU AI Consortium Fund, and Brain Korea 21 Plus Project in 2019.

References


HexaGAN: Generative Adversarial Nets for Real World Classification


