re-GAIN: Do Not Believe Everything You See

Nodens Koren
The University of Melbourne
nkoren@student.unimelb.edu.au

Abstract

Noise and corruption are often unavoidable in data collection, either because of limitations of current technologies, mistakes in the execution, or degradations of raw data themselves. In the medical domain, we can commonly see such data that contain noise. GAIN is an excellent model in dealing with missing data. However, it blindly trusts all observed data given to it, which leads to a significant decline in imputation performance when given unreliable data. In this work, we propose an anomaly detection-based noise detection module and incorporate it with GAIN into a pipeline system. The noise detection module provides an extra *dirty* vector to inform GAIN which data it should not rely on. In this way, GAIN will exclude these unreliable data from its observed set to make predictions. By doing so, we will increase the robustness of GAIN towards data corruption and thus generalize it better to real settings. We tested our method on various datasets and found that our proposed system substantially ameliorates the performance of GAIN in the presence of noise.

1 Introduction

Missing data is a critical problem in many fields. For instance, when predictions need to be made based on real-time data, it is infeasible to re-conduct experiments to collect the missing parts. In the medical domain, it is of particular importance because many data are temporal-related and highly personalized. It is impossible to travel back in time to complete missing data from the past, and occasionally it is not practical to redo examinations to recollect present data.

GAIN [1] does an excellent job in addressing the missing data issue. However, it still has some limitations. First, it assumes the data are missing completely at random (MCAR), whereas many data are missing not at random (MNAR) in practice [2, 3]. Second, it expects the data to be $i.i.d.$, so it might not be generalized to non-$i.i.d.$ settings such as federated learning [4]. Last but most importantly, it assumes the observed data are always reliable and are representative of the true data distribution. It also presumes that the missing vectors are always predefined by simply marking non-empty data as 1. In this work, we focus on this last limitation and extend GAIN to the setting where test data are corrupted instead of completely missing.

Corrupt or noisy data are frequently seen in the medical domain. As an example, DNA is one of the most commonly used features to identify one's identity. It is highly vulnerable to the environment, and the structure can deteriorate over time [5]. Medical imaging is another example where one should be aware of the existence of noise. The presence of noise in images produced by medical imaging equipment is common and unavoidable [6]. Furthermore, in specific cases where one's body contains foreign objects such as implants or devices, the image might not reflect the true skeletal or organic condition [7]. In these scenarios, even if the data are observed, they should not be taken as granted as reliable data to represent the true distribution.

The biggest challenge to extend GAIN to this setting is to determine the missing vector $\mathbf{M}$. Unlike the case where data are completely missing such that we can arrive at an obvious conclusion, it

Preprint. Under review.

remains unclear how to decide whether to include a datum into our observed set. To this end, we propose an anomaly detection-based noise detection method using a generated adversarial network (GAN) to extract missing vectors. We then take advantage of GAIN's superb performance on data imputation to recover corrupted data.

In summary, our contributions are:

- We extend GAIN to the setting where test sets contain not only missing but corrupt data.
- We propose a pipeline system that enables GAIN to perform missing data imputation and *corrupt* data recovery simultaneously when a complete training set is unavailable.
- We empirically show that our proposed system can effectively reduce and recover dirty data in corrupt images.

We highlight that the novelty of our work is not the use of GAN to perform anomaly detection. The primary interest of anomaly detection is to classify whether given test data fall into the same distribution as the training data. The new idea here is the use of pixel-level anomaly detection to extract missing and dirty vectors and the joint framework to combine the strength of two models for different tasks to assist each other.

2 Related Work

2.1 Anomaly Detection

Anomaly detection is an important task in many fields. It has a number of applications in the medical domain. For instance, we can employ an anomaly detection model to detect the presence of pneumonia and even the type of pneumonia (i.e. bacterial or viral) from medical images [8]. Traditional methods are mainly supervised learning-based methods including the use of random forests [9], support vector machines (SVM) [10, 11], deep belief networks (DBN) [10], and convolutional neural networks (CNN) [11, 12], with features either hand-crafted or implicitly learned by neural networks. Supervised learning has some downsides, however. They require a training set to be fully annotated, which could incur a high cost. Even if a fully annotated set is available, the prediction power is limited to already known labels [8].

In recent years, owing to the invention of GAN, numerous unsupervised-based approaches have been proposed [8, 13–16]. GAN-based approaches typically require only a training set of normal images [8, 13–16]. They take advantage of this to make the generator learn only the manifold of normal images so that when the generator sees the latent space representation of an anomaly sample, it will generate a "what-if" image as if the anomaly sample is from the same distribution as normal samples [8].

AnoGAN [8] is one of the earliest works in GAN-based anomaly detection. The system learns only the mapping from the latent space to the image space, and it uses an iterative search method to search for the latent space representation given a new image. The system also utilizes a residual loss to calculates similarities in a hidden layer feature space instead of in the image space [8, 14]. Its biggest downside is the poor test-time performance due to the use of iterative search to find latent space representations [13–16]. Later works have primarily focused on addressing this issue. For instance, EGBAD [15] hires an extra encoder to jointly learn the inverse mapping from the image space to the latent space with the generator, and GANomally [16] structures the generator and the discriminator into an autoencoder architecture.

In our work, we will follow f-AnoGAN [14] for our dirty data detection module. f-AnoGAN also employs an encoder to learn the inverse mapping from the image space to the latent space, but unlike EGBAD, it does not need to be trained in conjunct with the generator [14].

3 Problem Formulation

Consider a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. Suppose that $\mathbf{X} = (X_1, \cdots, X_d)$ is a random variable (either continuous or binary) taking values in $\mathcal{X}$, whose distribution we will denote $P(\mathbf{X})$. Suppose that $\mathbf{M} = (M_1, \cdots, M_d)$ is a random variable taking values in $\{0, 1\}^d$. Suppose that

$\mathbf{D} = (D_1, \cdots, D_d)$ is a random variable taking values in $\{0, 1\}^d$. We will call $\mathbf{X}$ the data vector, $\mathbf{M}$ the mask vector, and $\mathbf{D}$ the dirty vector.

For each $i \in 1, \cdots, d$, we define a new space $\tilde{\mathcal{X}}_i = \mathcal{X}_i \cup \{*\}$ where $*$ is simply a point not in any $\mathcal{X}_i$, representing an unobserved value. Let $\tilde{X} = \tilde{X}_1 \times \cdots \times \tilde{X}_d$. We define a new random variable $\tilde{\mathbf{X}} = (\tilde{X}_1, \cdots, \tilde{X}_d) \in \tilde{\mathcal{X}}$ in the following way:

$$\tilde{X}_i = \begin{cases} X_i, & \text{if } M_i = 1 \text{ and } D_i = 0 \\ *, & \text{otherwise} \end{cases} \tag{1}$$

so that $\mathbf{M}$ indicates which components of $\mathbf{X}$ are observed and $\mathbf{D}$ indicates components of $\mathbf{X}$ that are corrupted. We can recover $\mathbf{M}$ from $\tilde{\mathbf{X}}$, but we cannot determine $\mathbf{D}$ without additional information.

We follow Yoon et al. [1] to use lower-case letters to denote realizations of random variables.

3.1 Dirty Data Detection

In the dirty data detection setting, we assume n $i.i.d.$ copies of $\mathbf{X}$ are realized or imputed by GAIN, and we denote them by $\mathbf{x}^1, \cdots, \mathbf{x}^n$. Now we define the dataset $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{d}^i = \mathbf{0})\}_{i=1}^n$. We assume this dataset is a semi-gold standard which might be artificially imputed but has no corruption.

Our goal at this stage is to determine the realization of the dirty vector $\mathbf{d}^j$ for each new sample $\tilde{\mathbf{x}}^j$. Formally speaking, we would like to generate dirty masks according to $P(\mathbf{D}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^j, \mathcal{D})$, the conditional distribution of $\mathbf{D}$ given $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^j$ and $\mathcal{D}$ the gold standard set. Simply put, given a new vector $\tilde{\mathbf{x}}^j$ we wish to identify which components of $\tilde{\mathbf{x}}^j$ are corrupted.

3.2 Imputation

For the imputation part, we follow the same setting in Yoon et al. [1]. We assume we have n $i.i.d.$ realized copies of $\tilde{\mathbf{X}}$ denoted by $\tilde{\mathbf{x}}^1, \cdots, \tilde{\mathbf{x}}^n$ and we define the dataset $\mathcal{D} = \{(\tilde{\mathbf{x}}^i, \mathbf{m}^i, \mathbf{d}^i)\}_{i=1}^n$, where $\mathbf{m}^i$ is the recovered realization of $\mathbf{M}$ corresponding to $\tilde{\mathbf{x}}^i$ and $\mathbf{d}^i$ is the predicted dirty vector corresponding to $\tilde{\mathbf{x}}^i$ given by the dirty data detection stage. Alternatively, we can merge $\mathbf{d}^i$ into $\mathbf{m}^i$ by setting entries of $\mathbf{m}^i$ to be 0 if the corresponding entries in $\mathbf{d}^i$ is 1.

The goal is to impute the unobserved values in each $\tilde{\mathbf{x}}^i$. Formally, we generate samples according to $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$ to fill in the missing data points in $\mathcal{D}$.

4 Proposed Method

In this section, we describe our approach for extracting the dirty vector $\mathbf{d}^j$ given the unseen data vector $\tilde{\mathbf{x}}^j$. We also introduce our proposed system which enables us to train with incomplete data and test with corrupt data. Throughout the rest of the paper, we will use the terms images and data interchangeably, and the terms pixels and dimensions interchangeably.

4.1 Dirty Data Detection Module

Our approach is a GAN-driven method, and we follow the structure of f-AnoGAN [14] for anomaly detection. The whole module can be summarized as an encoder-decoder architecture. Our hypothesis is that the latent space representation is more robust to noise and variations, and thus as long as the corruption does not devastate most core features, the model will be able reproduce a clean image that resembles the ground-truth image.

4.1.1 Generator and Discriminator

We employ a deep convolutional generated adversarial network (DCGAN) for the dirty data detection task. The setup is similar to the standard GAN training. The goal is to learn a mapping from the k-dimensional latent space $\mathcal{Z} \subset \mathbb{R}^k$ to the manifold $\mathcal{X}$ in the image space. The generator G takes in realizations of noise variable $\mathbf{z}$ sampled from the latent space $\mathcal{Z}$ and is trained to output samples that

follow a distribution as close as possible to the real distribution $\mathbb{P}_r$ over the image space $\mathcal{X}$. Formally speaking, the generator's goal is to learn a nonlinear function: $G(\mathbf{z}) = \mathbf{z} \mapsto \mathbf{x}$.

The discriminator measures how close the learned distribution is to the real distribution. Instead of measuring the difference in the scalar output directly, we measure the difference in the intermediate feature space. We define the discriminator loss to be:

$$\mathcal{L}_D(\mathbf{z}) = \sum \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(G(\mathbf{z}))\|, \tag{2}$$

where $\mathbf{f}(\cdot)$ represents the feature vector of a hidden layer of the discriminator.

We remark that the generator takes the role of the decoder in this module.

4.1.2 Encoder

We employ a feedforward deep neural network as our encoder to learn the inverse mapping $E(\mathbf{x}) = \mathbf{x} \mapsto \mathbf{z}$ from the image space $\mathcal{X}$ to latent space $\mathcal{Z}$. The *image-z-image architecture* is adopted to train the encoder with real data (c.f. the *z-image-z architecture* which trains the encoder with generated data instead). Aside from the input space, we take into account the feature space of the discriminator network. The training loss function for the encoder architecture is defined by:

$$\mathcal{L}_{izi_f}(\mathbf{x}) = \frac{1}{n}\|\mathbf{x} - G(E(\mathbf{x}))\|^2 + \frac{\kappa}{n_d}\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(G(E(\mathbf{x})))\|^2, \tag{3}$$

where $\mathbf{f}(\cdot)$ represents the feature vector of a hidden layer of the discriminator, n_d is the dimensionality of the hidden feature representation, and κ is a weighting factor.

4.1.3 Training, Testing, and Dirty Vector Extraction

We use only the "normal data" in the context of anomaly detection during training. Many GAN-based anomaly detection algorithms exploit the facts that the learned manifold contains only information about the normal data (hence biased) and that the latent space transition is smooth. Consequently, the encoder will map an anomaly image into a latent space representation, and the decoder will generate an output image based on the latent space representation as if the input is from the same distribution.

In the dirty data detection setting, we feed corrupt images from the same class instead of anomalies to the module at test time. Similarly, we hypothesize that the latent space representation of a corrupt image will allow the decoder to generate an output image as if the input image is noise-free (since the generator did not model noise into its learned manifold). Additionally, we hypothesize that the generated image is geometrically close to the ground-truth (clean) version of the input image. This is a reasonable guess because majority of pixels in a corrupt image are still from the correct distribution.

With this belief, we assume that the difference between the corrupt data and the corresponding generated data provides a measurement of the dirty vector. That is, we assume:

$$\mathbf{d}_i = \begin{cases} 0, & \text{if } |\mathbf{x}_i - G(E(\mathbf{x}))_i| < \eta \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{d}_i$, $\mathbf{x}_i$, and $G(E(\mathbf{x}))_i$ are the i^{th} component of the dirty vector $\mathbf{d}$, the input vector $\mathbf{x}$, and the generated output vector $G(E(\mathbf{x}))$ respectively, and η is a threshold factor.

Note that we do not replace the corrupt image completely with the generated image. In this way, we can still preserve a portion of non-corrupted dimensions from the input space to provide information about the ground-truth distribution for GAIN to sample from.

4.2 The Complete System

We keep the GAIN model completely the same as presented in the original paper. In the case where training data are incomplete, we need to first train the GAIN model to extract a "filled-in" training set $\hat{\mathcal{D}}$, and then subsequently use $\hat{\mathcal{D}}$ to train the dirty data detection module. The learning of manifolds

in dirty data detection requires all components to be deterministic, or else the generator could learn several possible distributions that are all optimal with respect to the discriminator. In the case where the full training data is available, the training order of dirty data detection and data imputation does not matter. At test time, we always pass unseen data to the dirty data detection module first, and use the produced dirty vector in GAIN.

5 Experiments

5.1 Experiment Setting

5.1.1 Datasets

We evaluate the performance of our proposed system on the MNIST and the CIFAR-10 datasets.

MNIST For the MNIST dataset, we believe handwritten digits are simple enough and share many useful common features. We thus use all $60,000$ images in the training set as the normal set in training and ignore the labels. Each MNIST image is of dimension 28×28 and we keep the number of dimensions the same.

CIFAR-10 For the CIFAR-10 dataset, we test our system on three different classes of CIFAR-10 images: car, horse, and ship. There are $5,000$ training data and $1,000$ test data for each objective. We transform CIFAR-10 images from dimensions $3 \times 32 \times 32$ into dimensions $3 \times 64 \times 64$.

For both datasets, we normalize pixel values into the $[0, 1]$ range.

5.1.2 Basic Configuration

For all experiments, we use the Adam optimizer with a learning rate of 0.0002 with 10^{-5} weight decay. We use a latent space dimension of 100, a mini-batch size of 128, and a hint rate of 0.9. Unless otherwise stated we assume the training set is *complete* and 20% of pixels in each test image are dirty. Additionally, unless explicitly mentioned we assume the corruption level is unrestricted and the corruption happens at random locations. For the best results, we empirically set the threshold value η in Equation (4) to be 0.2.

5.2 Performance on Dirty Data Detection

We treat the dirty data detection problem as a binary classification problem at the pixel level to determine whether a pixel is dirty. If the difference between the ground-truth pixel and the clean pixel is under 20%, we count the pixel as clean. As a baseline, we train a deep convolutional neural network to output dirty vectors with $10,000$ noisy MNIST/CIFAR-10 data. We use the standard metrics for binary classification tasks: precision, recall, F1, and specificity to evaluate the performance, and we report the results in Table 1.

Table 1: Dirty Data Detection Performance on the MNIST Dataset and the CIFAR-10 Dataset.

Metric	Precision	Recall	F1	Specificity
re-GAIN (MNIST)	0.582	**0.918**	**0.712**	0.897
DCNN (MNIST)	**0.667**	0.712	0.689	**0.955**
re-GAIN (Car)	0.361	**0.711**	**0.479**	0.868
DCNN (Car)	**0.533**	0.422	0.471	**0.915**
re-GAIN (Horse)	0.375	**0.736**	**0.497**	0.876
DCNN (Horse)	**0.557**	0.432	0.486	**0.911**
re-GAIN (Ship)	0.401	**0.732**	**0.518**	0.845
DCNN (Ship)	**0.544**	0.425	0.477	**0.923**

As shown in the table, the proposed dirty data detection module has high recall and high specificity. Nonetheless, it has low precision. This indicates that the module can detect almost all dirty pixels from the given image, but it will also remove many extra clean pixels.

Note that our baseline model, on the other hand, has high precision but low recall. Our explanation for this observation is that the baseline model is not as sensitive to pixels with a low or medium level

of corruption. We also hypothesize that the baseline model will not be sensitive to continuous-type corruption (e.g. a local region of discoloration). To validate our claims, we investigate the models' performance when perturbations are constrained and when the corruption is continuous.

5.2.1 Low Corruption Level and Continuous-Type Corruption

We now validate our hypothesis about the baseline model and the proposed system under different types of corruption. To construct a test set with a low corruption level, we set the maximum perturbation to be 30%. To construct a test set with continuous-type corruption, we randomly discolor a small region that occupies 20% of the area in each image. We perform this experiment on the CIFAR-10 (ship) dataset and provide the results in Table 2.

Table 2: Dirty Data Detection Performance on Low Corruption and Continuous-Type Corruption.

Model	Precision	Recall	F1	Specificity
re-GAIN (Low Corruption)	0.389	**0.734**	**0.508**	0.809
DCNN (Low Corruption)	**0.673**	0.272	0.381	**0.965**
re-GAIN (Cont. Corruption)	0.445	**0.813**	**0.575**	0.879
DCNN (Cont. Corruption)	**0.693**	0.159	0.259	**0.958**

As we guessed, the baseline DCNN model is not proficient in distinguishing different types of corruption. It performs particularly poorly on identifying continuous-type corruption. From Table 2, we can see that the baseline model's precision remains high (because it did not make many positive predictions), whereas its recall and F1 drop substantially. On the other hand, the statistics remain similar for our proposed dirty data detection module. In the case of continuous-type corruption, our proposed module has even higher precision. These results validate our claim in the previous section and prove that our proposed module is more generalizable to any kind of noise distribution and corruption pattern.

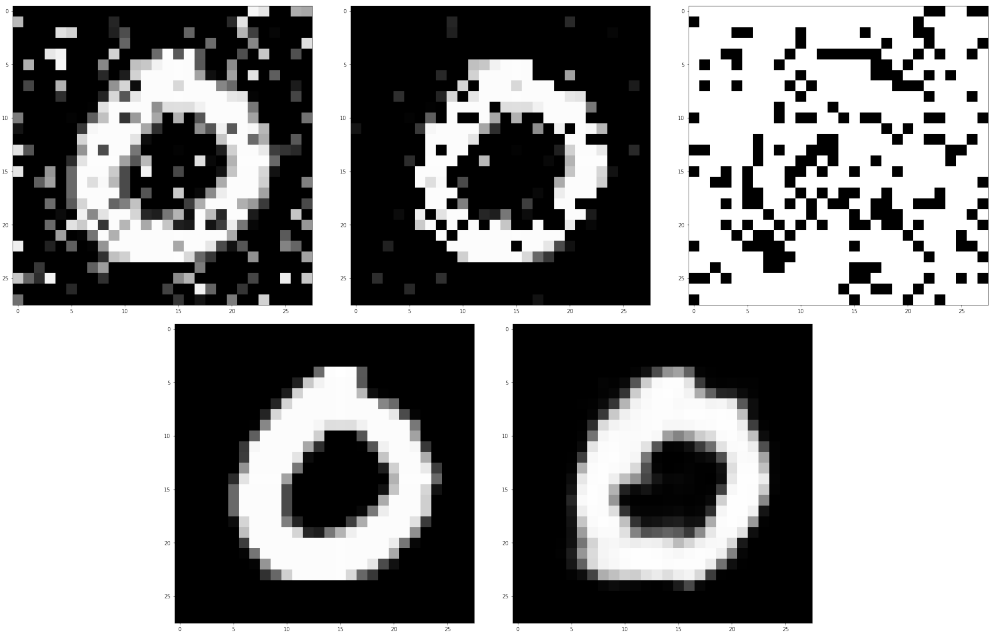

Figure 1: A sample result generated by re-GAIN. Top-left: a corrupt image, top-center: difference between the corrupt image and the corresponding generated image, top-right: the dirty vector (black means dirty), bottom-left: the corresponding original image, and bottom-right: the recovered image.

5.3 Performance on Corrupt Data Recovery

In this section, we present the overall performance of the data recovery pipeline system. The results are given in Table 3. We also give the root mean square error (RMSE) between dirty images and their

corresponding ground-truth images. A sample output from the corrupt data recovery system is given in Figure 1.

Table 3: RMSE of purified images and corrupt images.

Image	MNIST	Car	Horse	Ship
Purified	**0.149 ± 0.031**	**0.137 ± 0.025**	**0.125 ± 0.031**	**0.125 ± 0.034**
Corrupt	0.250 ± 0.010	0.175 ± 0.016	0.178 ± 0.015	0.177 ± 0.012

5.3.1 Tolerance of the Proposed System Towards Corruption

To evaluate how much noise our proposed pipeline system can tolerate in the task of corrupt data recovery, we plot the system's performance when 10% to 80% of pixels in test data are dirty. We demonstrate the results in Figure 2.

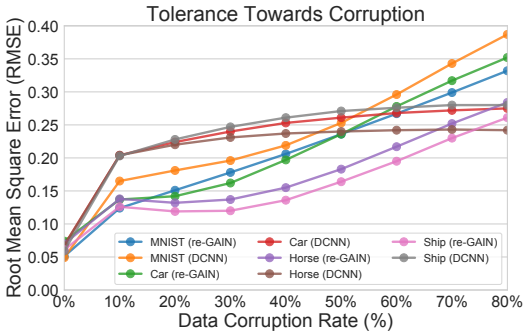

Figure 2: Tolerance Towards Corruption.

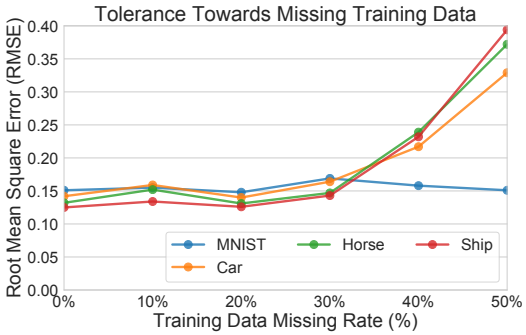

Figure 3: Tolerance Towards Missing Training Data.

As shown in the figure, our proposed system constantly outperforms the DCNN baseline until 70% of pixels are dirty. Even at 60% corruption, our proposed system beats the DCNN baseline by more than 4% RMSE on average. Our explanation for this observation is that if 70% of pixels are dirty, the image might have lost the core features used to encode the latent space representation for the given class. The latent space vector thus represents a point in a region that is far away from the distribution in the manifold, which we know nothing about. On the other hand, the DCNN baseline was trained to identify noise directly from the image space using convolutional kernels, which might be more sensitive to local dynamics.

Surprisingly, at the 10% corruption level, our proposed model generates a result that is even worse than the corrupt image. We attribute this phenomenon to two reasons: 1. We assume it is because of the metric we used to judge if a pixel is dirty or not: we count a pixel as dirty only if its value deviates > 20% from the original pixel value. Hence at the 10% corruption level, there might be only approximately 6% of pixels that fit this criterion. 2. The low precision rate indicates that the dirty data detection module will remove many more extra pixels than necessary. The error from imputing these accidentally removed pixels may exceed the error caused by the 6% of dirty pixels.

5.4 Training With Incomplete Data

In this section, we examine the system's performance when trained on incomplete data. Again, we adopt the standard setting where 20% of pixels in each image are randomly perturbed for testing. We investigate the system's performance when trained on datasets with 0%, 10%, 20%, 30%, 40%, and 50% missing data respectively, and we show the RMSE results of the system in Figure 3.

The results are quite different for the MNIST dataset and the CIFAR-10 dataset. For MNIST, the system's performance does not drop at all even if the data missing rate is high as 50%. Whereas for CIFAR-10, the system's performance degrades significantly when the data missing rate reaches the 40% mark. Upon further investigation, we found that the reason is due to GAIN's high RMSE (> 0.2) on CIFAR-10's training set when $> 40\%$ of pixels are missing in each sample. This is understandable because CIFAR-10 is a more complex and profound dataset than MNIST.

Overall, the system still has a very high tolerance to missing data in the training set.

5.5 Performance on Data Imputation

As the final experiment for this paper, we would like to study the proposed system's performance on the task of data imputation when images in the test set contain various degrees of corruption. We compare the performance of the rudimentary GAIN model with the proposed system which includes the dirty data detection module to visualize the improvement. We conduct our experiment on the MNIST dataset and the CIFAR-10 (ship) dataset, and we set the data missing rate to be 10%.

5.5.1 Robustness of GAIN Towards Corruption

GAIN estimates missing values based on observed data. However, it is yet unclear how well GAIN performs in the presence of noise. Is it still able to generate plausible results for missing values? In this subsection, we investigate the model's performance under the setting that an unknown subset of observed data is corrupted. The results are presented in Figure 4 and 5. As we can see, GAIN is not very robust to dirty pixels in test images, and using them as observed data for imputation can severely deteriorate the quality of generated outcomes.

5.5.2 Robustness of the Proposed System Towards Corruption

We now evaluate the performance of GAIN with the auxiliary of the dirty data detection module. Compared to the results in Section 5.5.1, the improvement indicates the extra robustness the proposed dirty data detection module provides to GAIN. We also present the results in Figure 4 and 5 for side-by-side comparisons.

As we can tell from the figures, our proposed system improves the performance of the rudimentary GAIN model by approximately 4% of RMSE on MNIST and 7% of RMSE on CIFAR-10 (ship) when the test data contain corrupt information. This final experiment is reflexive of the title of our paper, which warns that we should not always trust the observed data given to us.

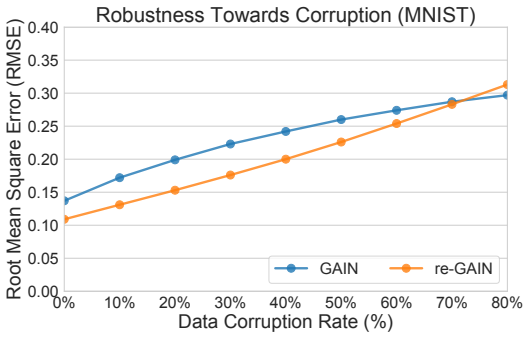

Figure 4: Robustness Towards Data Corruption in a 10% Missing Imputation Task (MNIST).

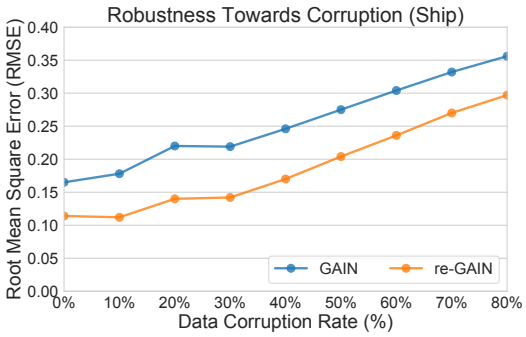

Figure 5: Robustness Towards Data Corruption in a 10% Missing Imputation Task (CIFAR-10 Ship).

6 Conclusion and Future Work

In this work, we propose a pipeline system to extend GAIN's applicability to more general settings. The newly added dirty data detection module provides an additional dirty vector to enable GAIN to perform corrupt data recovery on top of data imputation. We demonstrated empirically that the proposed system strengthened the rudimentary GAIN model in two aspects. It improves the robustness of GAIN towards corruption under the imputation setting. It also allows GAIN to catch and recover corrupt data automatically. Additionally, it does not introduce extra constraints in training. It demands the same training set as the rudimentary GAIN model, which can be incomplete but needs to be clean. We highlighted the importance of tolerance and robustness to corruption because noisy data are often unavoidable in the medical domain. Future work will look into methods to improve the overall performance of dirty data detection. We will also look into possibilities to relax constraints to permit the usage of corrupt data in training.

References

[1] Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018.

[2] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722, 2010.

[3] Craig K Enders. Missing not at random models for latent growth curve analyses. *Psychological methods*, 16(1):1, 2011.

[4] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[5] Kohki Kawane, Kou Motani, and Shigekazu Nagata. Dna degradation and its defects. *Cold Spring Harbor perspectives in biology*, 6(6):a016394, 2014.

[6] E. N. Manson, V. Atuwo-Ampoh, E. Fiagbedzi, J. H. Amuasi, J. J. Flether, and C. Schandorf. Image noise in radiography and tomography: Causes, effects and reduction techniques. *Current Trends in Clinical Medical Imaging*, 2(5):555620, 2019.

[7] Willi A Kalender, Robert Hebel, and Johannes Ebersberger. Reduction of ct artifacts caused by metallic implants. *Radiology*, 164(2):576–577, 1987.

[8] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[9] Freerk G Venhuizen, Bram van Ginneken, Bart Bloemen, Mark JJP van Grinsven, Rick Philipsen, Carel Hoyng, Thomas Theelen, and Clara I Sánchez. Automated age-related macular degeneration classification in oct using unsupervised feature learning. In *Medical Imaging 2015: Computer-Aided Diagnosis*, volume 9414, page 94141I. International Society for Optics and Photonics, 2015.

[10] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[11] Philipp Seeböck, Sebastian Waldstein, Sophie Klimscha, Bianca S Gerendas, René Donner, Thomas Schlegl, Ursula Schmidt-Erfurth, and Georg Langs. Identifying and categorizing anomalies in retinal imaging data. *arXiv preprint arXiv:1612.00686*, 2016.

[12] Thomas Schlegl, Sebastian M Waldstein, Wolf-Dieter Vogl, Ursula Schmidt-Erfurth, and Georg Langs. Predicting semantic descriptions from medical images with convolutional neural networks. In *International Conference on Information Processing in Medical Imaging*, pages 437–448. Springer, 2015.

[13] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection. *arXiv preprint arXiv:1906.11632*, 2019.

[14] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 2019.

[15] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[16] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer, 2018.