# PROJECT : IASD DATA WRANGLING, DATA QUALITY

## GAIN: MISSING DATA IMPUTATION USING GENERATIVE ADVERSARIAL NETS

BY JINSUNG YOON, JAMES JORDON MIHAELA VAN DER SCHAAR

**Yassine Abou Hadid & Ismail El Hadrami**
Master IASD, Université Paris-Dauphine, France
yassine.abou-hadid@dauphine.eu
ismail.el-hadrami@dauphine.eu

June 1, 2023

**ABSTRACT**

In this report, a reproduction of results is presented of a state-of-the-art model handling missing data, GAIN.
GAIN uses a variant of the GAN architecture, in which the generator (G) fills in the missing values, while the discriminator (D) has as input the the filled vector and finds which of these elements are filled or observed. To ensure that the model is trained, a hint vector is used to highlight partial information about the missing elements of the input vector. In addition, we implement modifications of this model. At the same time, we extend GAIN to image data studying its performance with various modifications and using convolutional levels.
Our code can be found **here**

.

## 1 Introduction

The presence of missing values in datasets is one of the most common problems encountered in the machine learning field, and can negatively affect machine learning models (Yoon et al. [2018]).
Data may be missing for various reasons that can be categorized in three different ways: the data is missing completely at random (MCAR) if the missingness occurs entirely at random (there is no dependency on any of the variables), the data is missing at random (MAR) if the missingness depends only on the observed variables, the data is missing not at random (MNAR) if the missingness is neither MCAR nor MAR (more specifically, the data is MNAR if the missingness depends on both observed variables and the unobserved variables; thus, missingness cannot be fully accounted for by the observed variables). MCAR constitute the category of missing data dealt with

in the GAIN paper.
The need to manage this problem has led to the proposal and implementation of various techniques (imputation methods), in order to create complete datasets. A relatively promising imputation technique is based on the Generative Adversarial Networks (GAN) (Goodfellow et al. [2014]).
We focused on the GAIN model to theoretically describe how it works. We tested some changes by interfering with the structure of the network, such as the depth of the neural network, the type of the hidden layers (use of convolution layers), and in changing the activation functions. Besides, we studied the above methodologies on the Spam (Hopkins et al. [1999]), Letter (Hopkins et al. [1991]), Credit Yeh and Lie [2009], News (Fernandes et al. [2015]), Breast (Wolberg et al. [1995]) datasets, while we also experimented on image data such as MNIST (mni) and Chest x-ray (Pneumonia) (Kermany et al. [2018]). The changes we implemented are detailed in Section 4.

## 2 Related work

In the field of research on imputation methods, the algorithms used could be divided into discriminative and generative imputation models. Generative models are models that generate data, such as images or text. They use probability to learn the underlying structure of data, while discriminative models are models that focus on making predictions based on existing input data.

Algorithms such as MissForest (Stekhoven and Buhlmann [2011]), Mice (van Buuren and Groothuis-Oudshoorn [2011]) and Matrix Completion (Mazumder et al. [2010]) belong to the class of discriminative models. While the second category includes algorithms such as Expectation-Maximization (EM), Denoising Auto-Encoders (DAE) (Costa et al. [2018]) and GAN, which is the basis of the model presented.

GAIN Yoon et al. [2018] is a generative method that achieves without a complete dataset the retrieval of incomplete data, avoiding sub assumptions about distributions and with a fairly good accuracy rate in MAR cases. It has simple operation, as it makes use of a simple GAN and by modifying the input data, it overcomes previous state-of-the-art methods.

## 3 Theoretical framework

### 3.1 Generative Adversarial Models (GAN)

To be able to analyze the GAIN method, we must first describe the architecture of GANs, which train two neural network models simultaneously through a competitive process. Specifically, they exploit a generative model and a discriminative model which compete with each other, and this competition contributes to the improvement of both. As mentioned above, the term 'generative' describes a class of statistical models which are capable of generating new data by learning the distribution of a sample (e.g., the training set). In contrast, discriminative models undertake to separate data into classes. In other words, the generative model finds the (joint) distribution $P(Q, Y)$

(or $P(Q)$), where $Q$ is the sample data and $Y$ their labels, while the discriminative models find the distribution $P(Y|X)$.
The generator accepts a random input and based on it generates a pseudo-sample. The input may follow any distribution, e.g., the normal distribution, or any other. At the same time, a real sample, and together with the generator's pseudo-sample, is given to the discriminator. He in turn will check the proximity of the two sample images and decide that the pseudo-sample is real if it calculates high proximity.

## 3.2 GAIN

GAIN networks follow, as mentioned, roughly the same training process, but additionally undertake to recover the missing data from the original sample (imputation). In the GAIN there is similarly the generator (G) that observes some data from a real vector data, imputes the missing data based on the data it has observed and extracts as output a complete data vector. The discriminator (D) then takes the full vector output by G and decides which elements were present in the first place and which were produced by imputation.
The random matrix and the mask matrix are both used to create a "masked" version of the original input. The random matrix is used to generate noise which will fill in areas of the image that have been removed by the mask matrix, thus creating an incomplete version of the original input for training purposes. The mask matrix defines which parts of the input should be kept intact and which should be replaced with noise from the random matrix. This masked version will then be fed into the generator.
It is of obvious importance to ensure that D forces G to learn the true distribution. To this end, D is additionally given as input a vector called the hint vector, which contains partial information about the missing elements of the input vector. D thus has a direction on how to check the elements of the input vector, paying most attention on some specific elements that the hint vector points to him.

To achieve this, the authors define two different loss functions.
The first, $\mathcal{L}_G : \{0,1\}^d \times [0,1]^d \times \{0,1\}^d \to \mathbb{R}$, is given by

$$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = -\sum_{i:b_i=0} (1 - m_i) \log(\hat{m}_i)$$

And the second, $\mathcal{L}_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, by

$$\mathcal{L}_M(\mathrm{x}, \mathrm{x}') = \sum_{i=1}^{d} m_i L_M(x_i, x_i'),$$

where

$$L_M(x_i, x_i') = \begin{cases} (x_i' - x_i)^2, & \text{if } x_i \text{ is continuous} \\ -x_i \log(x_i'), & \text{if } x_i \text{ is binary} \end{cases}$$

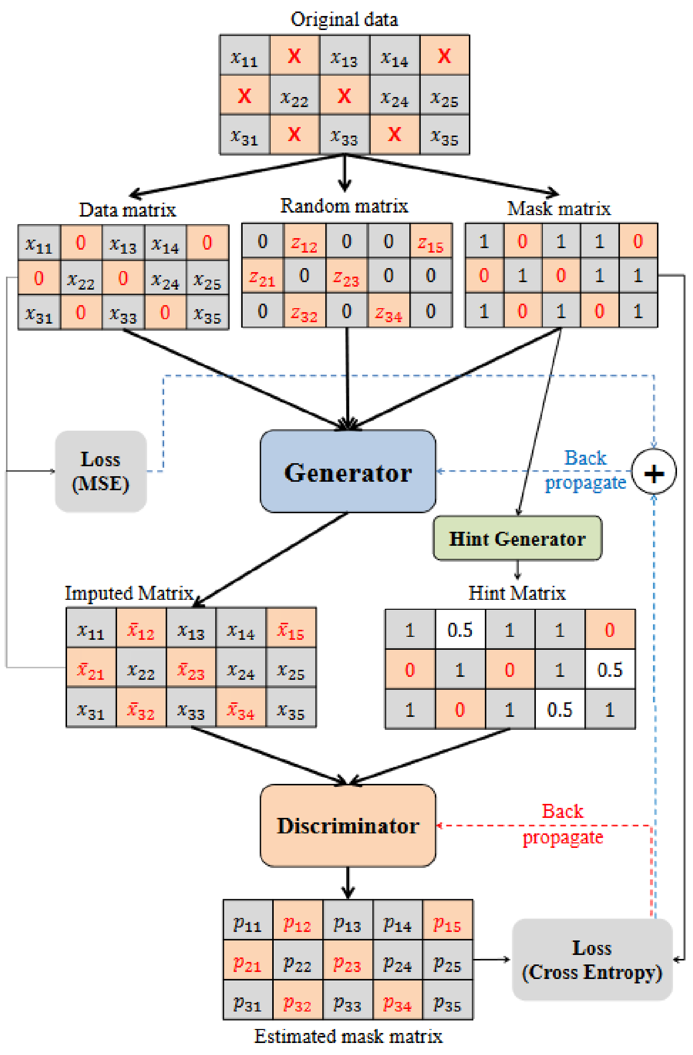

Figure 1: The architecture of GAIN

The loss function $\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}})$ is minimized when the imputed values $\hat{m}_i$ are close to 1 for those $i$ such that $m_i = 0$. This means that it is better if $D$ cannot recognize these values as imputed, as they will be falsely classified as observed. Similarly, $\mathcal{L}_M(\mathbf{x}, \tilde{\mathbf{x}})$ is decreased when the reconstructed features produced by $G$, which were previously observed, are similar to the original observed features.

$G$ is then trained to minimize the weighted sum of the two losses as follows:

$$\min_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\tilde{\mathbf{x}}(j), \hat{\mathbf{x}}(j))$$

where $\alpha$ is a hyperparameter.

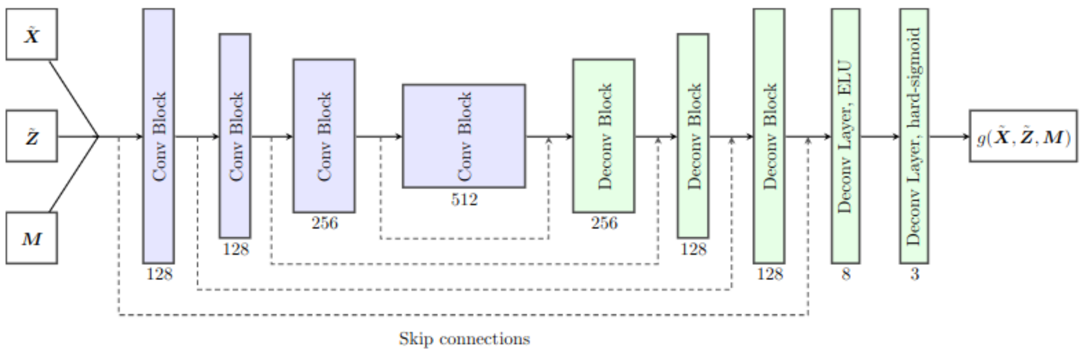

Figure 2: The architecture of the generator

## 3.3 WGAIN

The issues that come up when using GAIN are similar to the ones that arise when using GAN. Training GAIN can be slow and difficult to stabilize, and both algorithms can suffer from vanishing gradients and mode collapse, although it manifests itself differently. In GAN, mode collapse leads to G always producing the same output with little variety, whereas in GAIN it results in G always assigning the same values when given the same input and different noise values. To counter this, the idea emerged to combine the GAIN algorithm with the Wasserstein setting, resulting in something called Wasserstein Generative Adversarial Imputation Nets (WGAIN).

Both the generator and the critic networks are based on convolutional layers. The architecture of the generator g, as shown in Figure 2, is composed of building blocks of convolutional or deconvolutional layers with different dilation rates. Those building blocks are then combined in the encoder-decoder bottleneck topology with skip connections.

The skip connections enable the model to transfer high-precision characteristics from encoder layers to decoder layers (in reverse order), assisting the model in conveying the fine details in each depth more productively.

The critic part $f$ of the WGAIN model is a Lipschitz mapping $f : \mathcal{X} \times \{0,1\}^{m,n} \to \mathbb{R}$ represented by a deep convolutional network with norm restricted weights and fed by images and masks trained to maximize

$$\mathrm{E}_{\boldsymbol{X}\sim\mathrm{P}(\boldsymbol{X}),\boldsymbol{M}\sim\mathrm{P}(\boldsymbol{M})}\left(f(\boldsymbol{X},\boldsymbol{M}) - \mathrm{E}_{\boldsymbol{Z}\sim\mathrm{P}(\boldsymbol{Z})} f\left(\hat{\boldsymbol{X}}_{\boldsymbol{Z}},\boldsymbol{M}\right)\right)$$

which is estimated by sample means from mini-batches. This corresponds to the estimate of the expectation with respect to

## 4 Contribution

In this section we describe the experiments we performed to evaluate the performance of GAIN and present their results. The experiments were based on those applied in the original paper. In addition, we made certain changes to the standard architecture and extend the methodology to image datasets to see both graphically how well GAIN performs using different architectures, as discussed below. The code was implemented in python3 using PyTorch framework (in contrast to the paper authors who used TensorFlow). It can be accessed here: Project Repository.

## 4.1 Experiments reconstruction

### 4.1.1 RMSE with different settings

In this section, we implemented from scratch the experiments discussed in the paper to assess the GAIN model performance. The model consists of the input layer with an array of neurons composed of twice the number of features (due to the use of mask and hint vectors in the 2 models of GAIN), followed by two hidden layers and the output layer. This architecture is followed by the generator and the discriminator. We use the Rectified Linear Unit (ReLu) as the activation function of the hidden layers and Sigmoid for the output.

The operation of GAIN is based on three building blocks, the loss LG corresponding to the imputation of missing values from the generator network, the loss LM corresponding to the reconstruction of the original data from the generator network as well, and the hint vector which assists the discriminator network in the distinguishing between real and imputed data. Therefore, in order to evaluate the contribution of these elements, we excluded one or two of them at a time and measured the resulting RMSE. We used 5-fold cross validation and performed the experiment twice. Finally, we extracted the mean and the standard deviation of the RMSE results. The results are summarized in Table 1 As we observe, the lowest RMSE is achieved sometimes when all three

| **Algorithm** | **Breast** | **Spam** | **Letter** | **Credit** | **News** |
|---|---|---|---|---|---|
| **GAIN** | $0.1151 \pm 0.0002$ | $0.0577 \pm 0.0007$ | $0.13506 \pm 0.0050$ | $0.15052 \pm 0.0128$ | $0.20493 \pm 0.0056$ |
| GAIN w/o $\mathcal{L}_G$ | $0.10133$ $\pm0.0081$ | $0.0530$ $\pm0.00036$ | $0.1458$ $\pm0.0089$ | $0.18576$ $\pm0.0022$ | $0.2439$ $\pm0.0016$ |
| GAIN w/o $\mathcal{L}_M$ | $0.5174$ $\pm0.01692$ | $0.3388$ $\pm0.01379$ | $0.5292$ $\pm0.02202$ | $0.36634$ $\pm0.0966$ | $0.40017$ $\pm0.00747$ |
| GAIN w/o Hint | $0.1334$ $\pm0.0052$ | $0.0585$ $\pm0.0004$ | $0.1382$ $\pm0.0030$ | $0.18601$ $\pm0.00105$ | $0.20502$ $\pm0.00733$ |
| GAIN w/o Hint & $\mathcal{L}_M$ | $0.28847$ $\pm0.00473$ | $0.15308$ $\pm0.08163$ | $0.2197$ $\pm0.036927$ | $0.2197$ $\pm0.01212$ | $0.2353$ $\pm0.01729$ |

Table 1: Contribution of each loss function to the GAIN algorithm (Mean±Std of RMSE (Gain (%))).

losses are used. It is worth noting that the LM loss function seems to play a major role as, when is omitted, the RMSE increases by about 2 – 4 times, depending on the dataset (the increase varies naturally in different runs but remains within a relative range). The smallest contribution appears to be the LG function, as the RMSE increases relatively little when it is omitted.

### 4.1.2 RMSE vs parameters

For each of the 5 datasets, we measured the RMSE with respect to:

- the missing rate: the percentage of components missing from the vectors. (Figure 3)
- the data size: the size of a subset of the original dataset. (Figure 4)
- the hyperparameter $alpha$: hyperparameter appearing in the loss function. (Figure 5)
- the learning rate. (Figure 6)

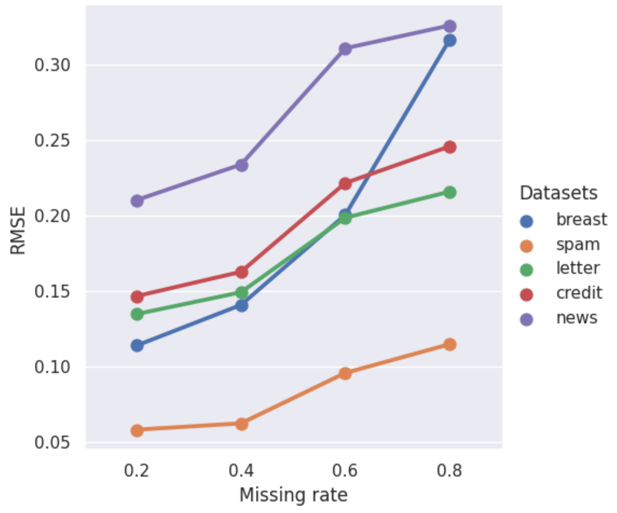

Figure 3: RMSE - Missing Rate.

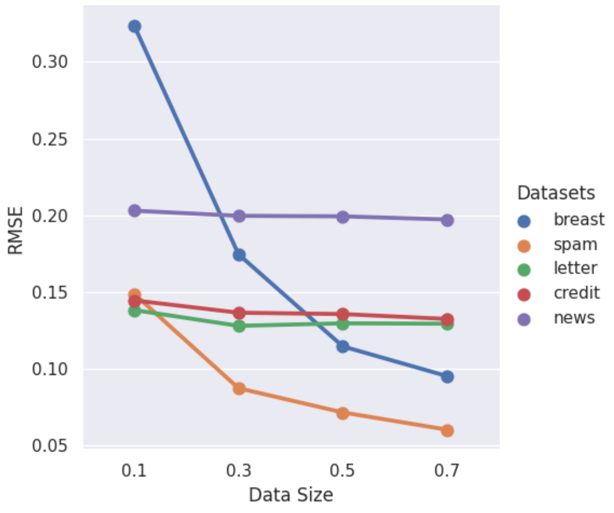

Figure 4: RMSE - Data Size.

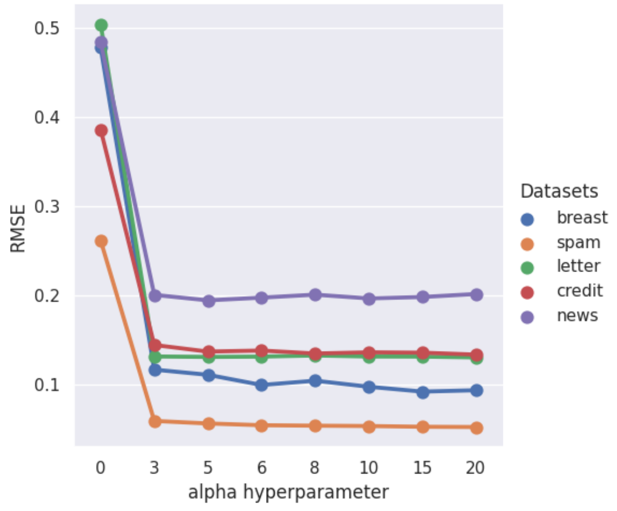

Figure 5: RMSE - hyperparameter.

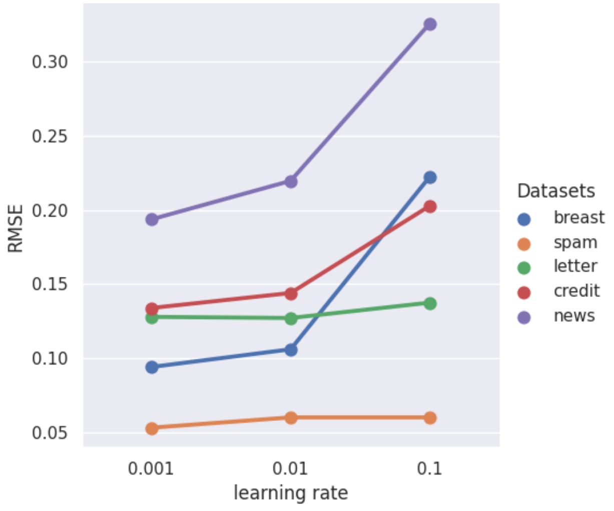

Figure 6: RMSE - Learning rate.

Figure 3 shows the RMSE - missing rate plot, in which we observe the trend in the RMSE to increase as the missing rate increases. When we experiment with subsets of the sample (Figure 4), we observe that in general the RMSE increases as the subset size decreases. The letter, credit and news datasets show the smallest increase in RMSE, possibly due to their large initial sizes, in the sense that both subsets of the initial sets are sufficient for training. A large increase was experienced in the RMSE of breast, of the smallest data set. In spam the RMSE decreased by about 35%.

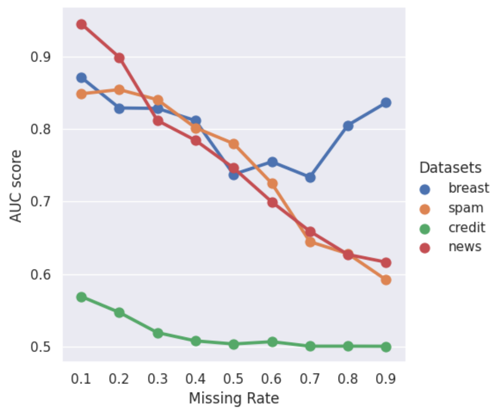

Figure 7: AUROC vs Missing Rate.

Figure 5 captures the change in RMSE with respect to the hyperparameter alpha in the loss function LM. We essentially relied on this plot to trap alpha in our code. Figure 6 describes the learning rate in the spam dataset with the change in RMSE with respect to the training epochs - from 100 to 10,000.

#### 4.1.3 Prediction quality post-imptaitons

We evaluate the predictive ability on the imputed data with the Area Under the Receiver Operating Characteristic Curve (AUROC) as a metric.

Figure 7 shows the AUROC metric versus missing rate for the datasets mentioned using the logistic regression algorithm. We observe that in general, as the noise in the sets increases datasets, the lower the predictive ability, with one exception in the breast dataset, which in the end shows an increase.

### 4.2 GAIN architecture improvement: GAIN-V2

We reproduce the standard implementation of the GAIN framework by adding another hidden layer. We also use *tanh* as the activation function for the hidden layers.

| **Algorithm** | **Breast** | **Spam** | **Letter** | **Credit** | **News** |
|---|---|---|---|---|---|
| **GAIN-V2** | $0.08687 \pm 0.003$ | $0.05156 \pm 0.0006$ | $0.1207 \pm 0.002$ | $0.1405 \pm 0.0067$ | $0.1978 \pm 0.0018$ |
| GAIN-V2 w/o $\mathcal{L}_G$ | 0.08689 $\pm 0.0063$ | 0.0518 $\pm 0.0009$ | 0.1223 $\pm 0.0020$ | 0.1791 $\pm 0.0044$ | 0.2246 $\pm 0.0028$ |
| GAIN-V2 w/o $\mathcal{L}_M$ | 0.3760 $\pm 0.0216$ | 0.1195 $\pm 0.0417$ | 0.3186 $\pm 0.0154$ | 0.2405 $\pm 0.0273$ | 0.3710 $\pm 0.0334$ |
| GAIN-V2 w/o Hint | 0.0982 $\pm 0.0086$ | 0.0514 $\pm 0.0011$ | 0.1216 $\pm 0.0021$ | 0.1520 $\pm 0.0031$ | 0.2002 $\pm 0.0059$ |
| GAIN-V2 w/o Hint & $\mathcal{L}_M$ | 0.2955 $\pm 0.0122$ | 0.0762 $\pm 0.0120$ | 0.3269 $\pm 0.0479$ | 0.2054 $\pm 0.0164$ | 0.2640 $\pm 0.0305$ |

Table 2: GAIN-V2 (Mean±Std (Gain (%))).

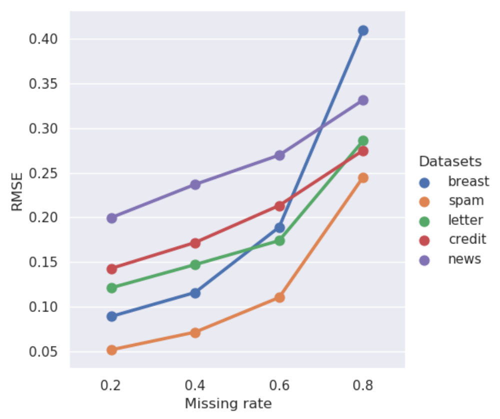

Figure 8: RMSE - Missing Rate.

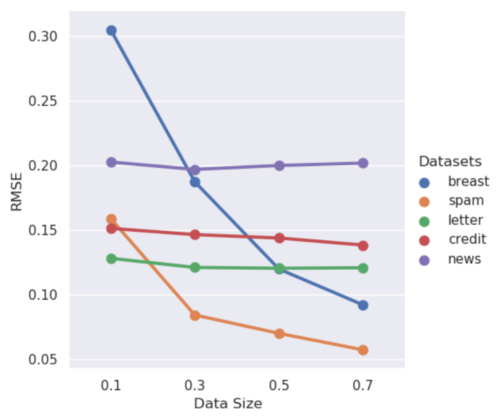

Figure 9: RMSE - Data Size.

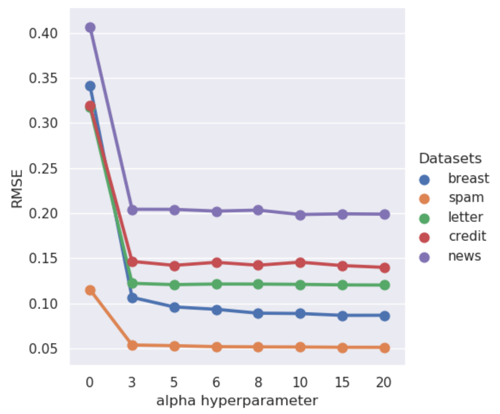

Figure 10: RMSE - hyperparameter.

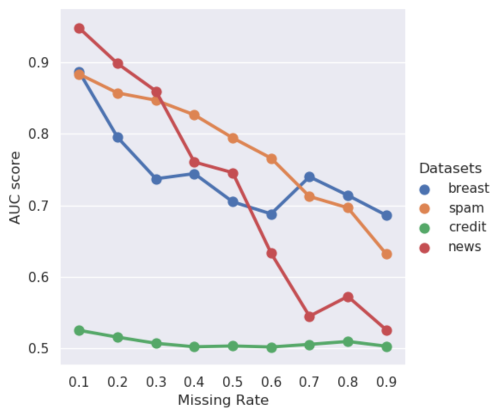

Figure 11: AUROC vs Missing Rate.

Compared to the corresponding results of the standard architecture, GAIN-V2 achieved a lower RMSE when all loss functions were included. Figure 8 shows the increase in RMSE in relation to the increase in the missing rate. We can observe that the changes are milder in the cases of large datasets, credit and news, smaller in the case of spam and letter, and more rapid-explosive in the case of breast, the smaller dataset. Compared to the corresponding graph in the standard architecture, in this case we observe greater uniformity and stability in the change in RMSE. The RMSE-data size plots of the Figure 9 and RMSE-hyperparameter alpha in Figure 10 are very similar to those of the standard architecture. Finally, Figure 11 corresponds to Figure 8 of the standard architecture and concerns the metric AUROC vs missing rate. With the exception of breast, which in the case of the architecture that we tested ends up with a downward trend while in the model it ends up with an upward trend, the rest of the datasets move with similar in terms of the slope of the AUROC descent.

## 4.3 Extending GAIN to Image Data

### 4.3.1 WGAIN performance

We extended GAIN to image data with WGAIN. With this method, we are able to use the power of generative models to fill in the missing values in an image dataset. We tested the approach on the CelebA dataset and obtained promising results. The model was able to accurately impute missing information while maintaining realistic facial features of each individual face without blurring or distorting them. The resulting images had very high fidelity and were visually appealing, demonstrating that WGAIN is capable of producing highly accurate reconstructions from incomplete datasets.

In our attempt to effectively extend the GAIN method to image data, we have tested several variations of the model architecture, such as using convolutional

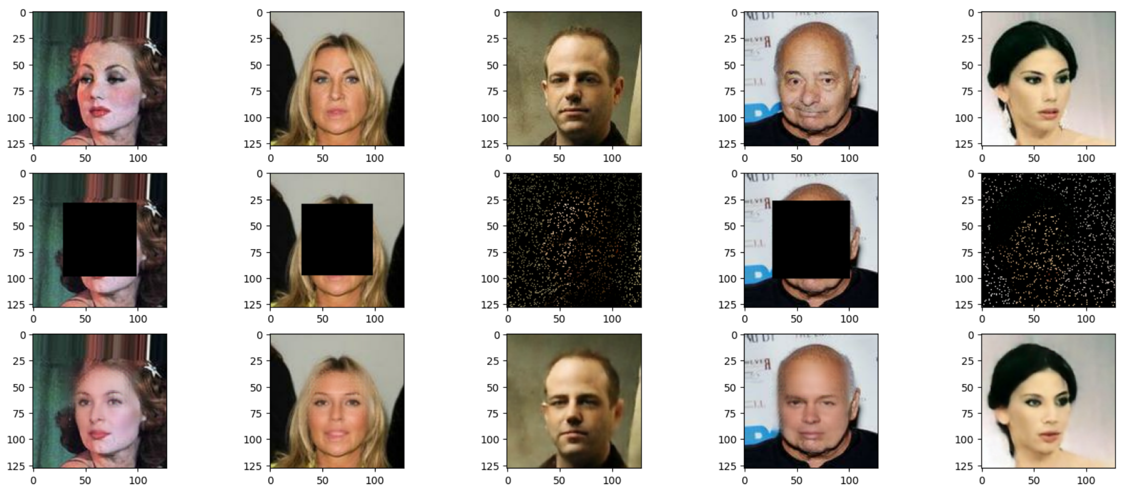

Figure 12: WGAIN results. First line represent the input image, the second one simulates missingness and third one is the reconstruction via WGAIN

### 4.3.2 Adapting GAIN architecture

In our attempt to effectively extend the GAIN to image data, we have tested several variations of the model architecture. In addition to adapting GAIN and GAIN-V2 seen previously, we propose three new variations: GAIN-FAST, GAIN-CNN and GAIN-DEEP. For the GAIN-FAST model, we have been inspired by the Auto-Encoder (AE) architecture, which is quite efficient in learning features and finding specific transformations. The idea for GAIN-CNN is based on related research, in which it is observed that the neural network can learn the features of an image much better using convolution. For the GAIN-DEEP model, we used four hidden layers. Further details (number of layers, parameters, activation functions…) about each of the introduced architectures can be found in project notebook. For the experiments, we used two image datasets: MNIST and PNEUMONIA (chest X-ray).

### 4.3.3 Imputation visualisation - RMSE results

In Figure 13 we have in the first line a subset of image data from the MNIST dataset with 20% noise. Then we filled, in the following lines respectively, the missing values using GAIN, GAIN-V2, GAIN-FAST, GAIN-CNN and GAIN-DEEP. Graphically, it can be seen that GAIN-CNN and GAIN-DEEP produce sharper images.

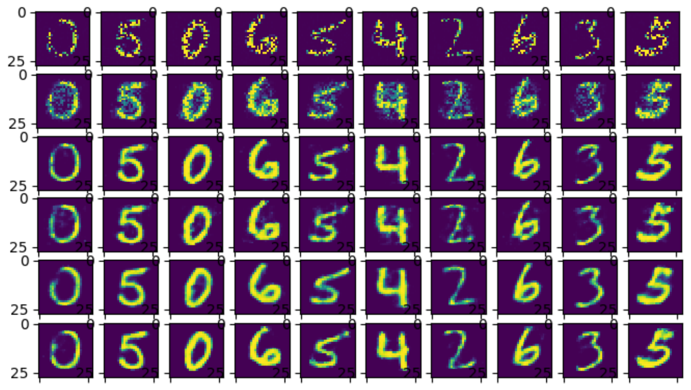

Figure 13: Imputations on MNIST dataset.

Figure 14 shows in the first line a subset of actual images from Pneumonia dataset. This is followed in the second line by these images with 20% noise. Finally, in the 3rd and 4th lines respectively, we have filled in missing values with GAIN-FAST and GAIN-DEEP. We note that the 2 last lines are quite different from the first one, which corresponds to the actual images.

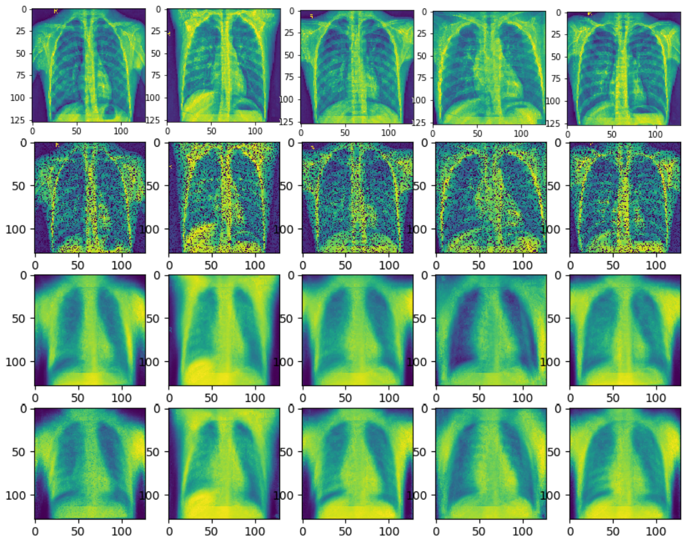

Figure 14: Imputations on PNEUMONIA dataset.

The RMSE results obtained on the MNIST and Pneumonia datasets, for all the different GAIN architectures, are presented in Table 3. We observe that the lowest RMSEs are obtained with GAIN-

DEEP and GAIN-CNN. It follows the GAIN-V2, and the GAIN-FAST in which the dimensions of the layers are gradually reduced. Finally, GAIN with the standard architecture gives the largest RMSE. All these results were obtained with missing rate of 0.2. We note that in the Pneumonia dataset, due to many features and lack of memory, we were able to test the GAIN, GAIN-V2 and GAIN-CNN networks.

| **Architecture** | **GAIN** | **GAIN-V2** | **GAIN-FAST** | **GAIN-CNN** | **GAIN-DEEP** |
|---|---|---|---|---|---|
| **MNIST** | 0.1836 | 0.1268 | 0.1179 | 0.1289 | 0.1195 |
| **Pneumonia** | - | - | 0.1068 | - | 0.1195 |

Table 3: RMSE scores on datasets with images for 0.2 missing rate.

Below are the results from the 2 datasets with images having entered 50% noise. The first two lines of Figure 15 show the data with noise, while the last two lines show the the data after imputation. Similarly, in Figure 16 the first line corresponds to data with noise while the last line corresponds to the data after imputation. The imputation was performed with the architecture GAIN-FAST.

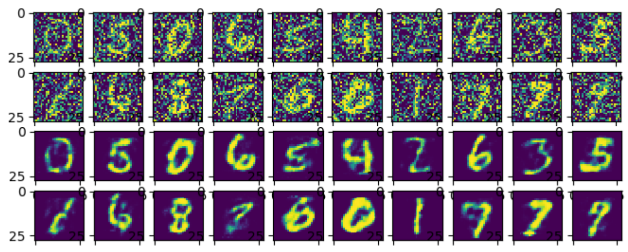

Figure 15: Imputation with 50% noise in MNIST dataset - RMSE=0.1630.

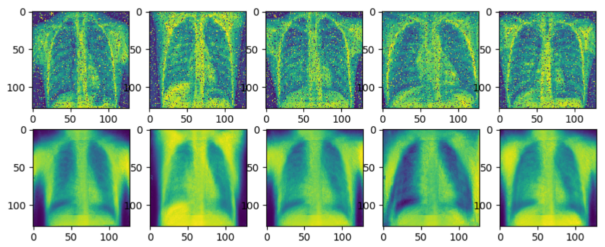

Figure 16: Imputation with 50% noise in the Pneumonia dataset - RMSE=0.1014.

## 4.4 Conclusion

In this report, we've done a replication of the results of the paper "GAIN: Missing Data Imputation using Generative Adversarial Nets" Yoon et al. [2018]. We also extend this methodology for image data sets such as MNIST, Chest X-Ray Images (Pneumonia) and CelebA. Subsequently, we propose alternative architectures to improve GAIN. The code can be found in this link.
We also wanted to extend our work and the GAIN method to time series, by leveraging some previous research (Luo et al. [2018]) that has been done to tackle this type of data or evaluating the GAIN method in a low-data regime since the original paper evaluated the GAIN method on datasets with a moderate amount of missing data, but did not investigate its performance on datasets with very few observed values. We could've evaluated the GAIN method in a low-data regime, such as by subsampling the available data or creating artificial datasets with varying levels of sparsity. But unfortunately, we couldn't produce results in time, and we chose to have quality results instead of effectuating a multitude of experiments.

## References

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Gain: Missing data imputation using generative adversarial nets, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. UCI machine learning repository, 1999. URL http://archive.ics.uci.edu/ml/datasets/spambase.

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. UCI machine learning repository, 1991. URL http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

Cheng Yeh and Che-hui Lie. UCI machine learning repository, 2009. URL https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients?fbclid=IwAR24zSxaMV6-JzPETXxGPmG3G7BQ_FIJ5jgCFK3DUyKSQ2ZdH1NXr64dDG4#.

Kelvin Fernandes, Pedro Vinagre, and Paulo Cortez. UCI machine learning repository, 2015. URL https://archive.ics.uci.edu/ml/datasets/online+news+popularity?fbclid=IwAR25BlNFWm5f3qPG8-9CahbrYwxl5zFuznD6sLa-8FJl6G2bWP-a17qB2qY.

William Wolberg, Nick Street, and Olvi Mangasarian. UCI machine learning repository, 1995. URL https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

URL http://yann.lecun.com/exdb/mnist/.

Daniel Kermany, Kang Zhang, and Michael Goldbaum. Labeled optical coherence tomography (oct) and chest x-ray images for classification, 2018. URL http://dx.doi.org/10.17632/rscbjbr9sj.2.

D. J. Stekhoven and P. Buhlmann. MissForest–non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, oct 2011. doi:10.1093/bioinformatics/btr597. URL https://doi.org/10.1093%2Fbioinformatics%2Fbtr597.

Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45:1–67, 2011. doi:10.18637/jss.v045.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v045i03.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

Adriana Costa, Miriam Santos, Jastin Soares, and Pedro Henriques Abreu. *Missing Data Imputation via Denoising Autoencoders: The Untold Story: 17th International Symposium, IDA 2018, ’s-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings*, pages 87–98. 01 2018. ISBN 978-3-030-01767-5. doi:10.1007/978-3-030-01768-2_8.

Yonghong Luo, Xiangrui Cai, Ying ZHANG, Jun Xu, and Yuan xiaojie. Multivariate time series imputation with generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/96b9bff013acedfb1d140579e2fbeb63-Paper.pdf`.