Meta-GAIN: Missing Data Imputation with Fast Domain Adaptation

Silvia Sapora

Abstract

Missing data imputation is often difficult in low-data settings, as models struggle to capture the underlying data distribution from only a few samples. We present Meta-GAIN, an extended version of the GAIN algorithm for data imputation. Our method employs meta-learning techniques so the model can integrate knowledge from a variety of domains. Meta-learning allows the model to execute accurate imputation while only needing a few training samples. This is particularly useful in medical contexts, where big datasets are rare but accurate and customizable imputation for each hospital is essential. Our model is able to adapt at test time, shifting its parameters to cater to a specific patient distribution. Our proposed implementation is also more stable, as it uses Wasserstein distance to ensure more stable training and prevent mode collapse, an issue commonly observed in the classic GAIN algorithm.

1. Introduction

Missing data is a pervasive issue and an obstacle to applying existing ML approaches and executing data-analysis on real-world datasets. Real data is often plagued by mistakes, corruption and inconsistencies that make training challenging and sometimes impossible. Recently, many have proposed solutions to this challenge(Smieja et al., 2018; Shen et al., 2020; Mattei & Frellsen, 2019).

In this paper, we will provide a critique and a proposed extension for GAIN (Yoon et al., 2018b), a GAN approach to impute missing data and present our extension: Meta-GAIN. The GAIN framework builds on top of the classic GAN architecture (Goodfellow et al., 2014) and is able to impute missing features accurately while not requiring complete data during training. In GAIN, the Generator tries to fool the Discriminator into thinking the imputed values are real, while the Discriminator tries to guess which values are real and which are imputed.

In each one of the following subsections, we will introduce a weakness, assumption or limitation of the original GAIN model, briefly describe how the related literature deals with the problem and then present our proposed solution to the issue(if any).

1.1. Missingness Mechanism

The underlying cause of data missingness can be categorized into three different mechanism(Rubin, 1987):

- Missing At Random(MAR): Missingness is caused fully by the observed variables in the dataset.
- Missing Completely At Random(MCAR): Missingness doesn't depend on any variable, observed or unobserved. It is a special case of MAR.
- Missing Not At Random(MNAR): Missingness depends on both observed and missing variables(including the variable itself).

GAIN provides a theoretical proof of its algorithm for the MCAR setting only, although it provides empirical results under MAR and MNAR. MCAR, as we mentioned before, is a stronger assumption than MAR and MNAR.

Mattei & Frellsen use IWVAE and Nazabal et al. use HI-VAE for data imputation. VAEs optimize for a lower bound on Maximum Likelihood Estimation, which only requires MAR assumption.

If we assume a MNAR missing mechanism, then the mechanism needs to be explicitly modelled and taken into account. Not doing so might lead to significant bias in the results.

Recent work from Ipsen et al. extends IWVAE to support MNAR data. The new model(called not-IWVAE) allows users to incorporate prior information about the type of missingness into the model.

Although VAE implementations stand on a stronger theoretical ground for MAR imputation, as they explicitly model the data distribution and have tractable likelihoods, we chose to try and extend the original GAN implementation, as it provides strong empirical performance even if it doesn't provide theoretical proofs for MAR and MNAR.

Partial code implementation available at https://github.com/SilviaSapora/Meta-GAIN

1.2. Performance

The original GAIN publication showed GAIN outperforming many state-of-the-art imputation methods, but this claim has been disputed. A few recent papers found GAIN performance to be quite unstable on some types of datasets(Nazabal et al., 2020; Mattei & Frellsen, 2019).

Li et al. observed GAIN predictions would become almost constant as the length of training increased. Wildly oscillating performance and identical outputs from the generator(given different latent space inputs) are two tell-tale signs of mode collapse.

In mode collapse, the Generator network only learns how to generate samples from a few modes of the true distribution and misses many other modes, even though the model is trained on representative samples of the distribution.

Many have proposed solutions to the issue of mode collapse and GAN training instability(Srivastava et al., 2017; Metz et al., 2017). Arjovsky et al. present Wasserstein GAN(WGAN), a theoretically-backed solution with strong empirical results. We adopt WGAN to extend Meta-GAIN and ensure training is more stable than the original GAIN implementation.

1.3. Heterogeneous Datasets

It is especially important that a model is applicable to a heterogeneous dataset, as many real-world datasets contain variables with different types. For instance, in healthcare applications, a patient record may contain demographic information such as nationality(categorical), age (ordinal) and height(continuous).

GAIN only supports binary and continuous data(in the Supplementary Material performance on categorical data is also analyzed). In their work, Nazabal et al. propose a generative model that handles numerical (continuous real-valued and positive real-valued and discrete count data) and nominal data (categorical and ordinal data).

Applying the standard implementation of VAE to mixed type datasets can lead to poor results(Ma et al., 2020). This is because each feature type uses a different likelihood function, and contributions from different loss functions can lead to multiple training objectives, where some dimensions end up dominating the training. This justifies why Ivanov et al.'s version of GAIN performs better than the original(particularly with respect to the addition of weights 0.8 and 0.2 to cross-entropy reconstruction for binary and categorical features respectively). We adopt Ivanov et al.'s proposal and add this tweak to our generator's loss function. Although this is not a general solution to the issue, it shows promise for improving performance on the benchmarks.

2. Background

In this section, we introduce a few concepts and related work necessary to understand our Meta-GAIN extension.

2.1. Few-Shot Learning

Learning effectively from low-data settings is crucial in fields where large datasets are often not available(e.g. counterfactual estimation and survival analysis(Lee et al., 2018)). In the original paper, GAIN performs worse than other benchmarks in low-data settings, limiting its applicability to settings where data is rare, expensive or challenging to collect. Particularly, this severely restricts the applicability of GAIN to fields such as the medical one, where it is often critical to perform accurately using limited data from a single hospital.

Standard learning approaches and models(such as GAIN) assume the train and test set to be drawn from the same distribution $p(\mathcal{D})$. Since the learner doesn't know $p(\mathcal{D})$, the true error is not directly available to the learner. One useful approximation to the true error is the training error. Trying to minimize the training error is a process called Empirical Risk Minimization, and assumes the datapoints of the training set S are i.i.d. sampled from the true distribution $p(\mathcal{D})$(Shalev-Shwartz & Ben-David, 2014). In the case of distribution shift, this assumption doesn't hold anymore, and the performance of our model starts to rapidly deteriorate.

Across hospitals, a distribution shift is present as different hospitals cater to different subsets of the population. One hospital might be in an area where young people live, which, needless to say, will have dramatically different characteristics compared to older people. Trying to train a model using both datasets will likely severely bias the model, causing inaccurate predictions in both hospitals. A significant amount of research exists to use various transfer learning techniques to be able to leverage data across different hospitals and medical studies(Yoon et al., 2018a; Bellot & van der Schaar, 2019). There are two main challenges in this approach: heterogeneous domains(not to be confused with heterogeneous datasets) and domain shift. Heterogeneous transfer learning attempts to utilize datasets that have different feature spaces. In hospitals, this might be caused by different protocols and procedures for record-keeping. Domain shift refers to the issue(described above) of distribution shift.

Our posed method, Meta-GAIN, allows for the model to be pre-trained on data from a variety of domains and only needs a few examples to be able to adapt to the specific settings it needs to cater for. Although few-shot learning generally refers to the model's ability to predict accurately with only a handful of samples, in this case, we think more samples($\sim$100) will be necessary for accurate adaptation.

2.2. Domain Adaptation

Domain Adaptation tries to learn a model that can adapt to perform accurately across domains despite domain shift. Domain Adaptation first trains a model on a set of labelled training data from a source domain, then tries to adapt it to perform well on a target domain with a different distribution and little or no labelled training data(Ben-David et al., 2010).

In our approach, the goal at training time is to learn common distribution properties that are valid across domains. At test time, our model will be fine-tuned on a small data sample from the target domain to cater to the specific target distribution.

2.3. Meta-Learning

Meta-Learning, or learning to learn, is a paradigm that allows machine learning systems to learn over multiple learning episodes, effectively allowing the models to *learn how to generalize*(Thrun & Pratt, 1998).

Model-Agnostic Meta-Learning (MAML) is a recent gradient-based method for fast adaptation to a variety of tasks (Finn et al., 2017). MAML doesn't introduce any extra parameters to the model, as other Meta-Learning approaches do(Ravi & Larochelle, 2017; Andrychowicz et al., 2016). The goal of the MAML learning process is to maximize the sensitivity of the parameters to the loss function of new tasks: with high sensitivity, small local changes to the parameters can lead to large improvements in task loss.

We adapt the MAML algorithm to the domain adaptation case, where instead of tasks we have domains, and rather than task adaptation, we want MAML to quickly adapt to different distribution shifts. This isn't the first time Meta-Learning has been used to tackle generalization problems(Li et al., 2017)(Dou et al., 2019), but the use of meta-learning to address the issue of data imputation is novel.

3. Problem Formalism

Our goal is to build a model that can learn from different domains to adapt and achieve accurate imputation on any target domain. We will reuse most of the symbolic notation and problem formalism of the original GAIN paper, but we will introduce a few additions to adapt the notation to the multi-domain setting.

The data is provided is made up of N domains $\{\mathcal{D}_i\}_{i=1}^{N}$. Each domain is made up of M i.i.d. datapoints $\mathcal{D}_i = \{\tilde{\mathbf{x}}_j, \mathbf{m}_j\}_{j=1}^{M}$ sampled from the domain distribution $p(\mathcal{D}_i)$. For each domain $\mathcal{D}_i$ we will divide the given datapoints into a source and a query set, denoted as S_i^S and S_i^Q respectively.

A space $\mathcal{X} = X_1 \times ... \times X_{d_1}$ is given, where each X_i can be binary or continuous. We introduce the symbol $*$ to represent a missing feature. We create and extended space $\tilde{\mathcal{X}} = \mathcal{X} \cup \{*\}$ to include the missing value $*$. $\tilde{\mathbf{x}}_{ij} \in \tilde{\mathcal{X}}$ is a feature vector. Any arbitrary number of values can be missing from $\tilde{\mathbf{x}}_{ij}$. Each feature $k \in \{1, ..., d_1\}$ is either observed or missing.

The vector $\mathbf{m}_{ij} \in \{0, 1\}^{d_1}$ indicates if a feature is missing so that, for each feature $k \in \{1, ..., d_1\}$:

$$\tilde{\mathcal{X}}_i = \begin{cases} x_{ijk} \text{ if } m_{ijk} = 0 \\ * \text{ if } m_{ijk} = 1 \end{cases}$$

The goal is to impute the value of $\tilde{\mathbf{x}}$ that correspond to a 0(i.e. the feature is missing) in the $\mathbf{m}$ vector. From now on we will generally drop the i domain index from vectors, as the domain will be clear from the context.

4. Proposed Method

In this section, we will discuss our proposed Meta-GAIN implementation. In 4.1 we will describe how we changed GAN training to be more stable. In 4.2 we will present the meta-learning extension of GAIN.

4.1. Stabilizing Training

We adapt our implementation to follow the recommendations made by Arjovsky et al. in their paper. We start by defining a new vector $\mathbf{c}$ as

$$\mathbf{c}_j = \mathbf{b}_j + \mathbf{m}_j$$

We can then change the Discriminator D_ϕ loss to

$$\mathcal{L}_D(\mathbf{m}_j, \hat{\mathbf{m}}_j, \mathbf{b}_j) = \frac{1}{d_o} \sum_{k:c_{jk}=1} \hat{m}_{jk} - \frac{1}{d_m} \sum_{k:c_{jk}=0} \hat{m}_{jk},$$

$$\text{where } d_o = \| \mathbb{I}(\mathbf{c}_j = \mathbf{1}) \|_1 \text{ and } d_m = \| \mathbb{I}(\mathbf{c}_j = \mathbf{0}) \|_1$$

Where $\mathbb{I}$ represents the indicator function so that, for each element in the two vectors

$$\mathbb{I}(x_{jk} = y_{jk}) = \begin{cases} 0 \text{ if } x_{jk} = y_{jk} \\ 1 \text{ if } x_{jk} \text{ != } y_{jk} \end{cases}$$

We also replace the Adam optimization used in the original GAIN implementation with RMSProp(Tieleman & Hinton, 2012) and add parameter clipping.

Finally, the Discriminator executes multiple optimization steps s_D (Arjovsky et al. suggest 5) for each optimization step the Generator executes. All the modifications can be observed in Algorithm 1.

4.2. Meta-GAIN

The Meta-GAIN algorithm, described in detail in Algorithm 1, adapts the MAML(Finn et al., 2017) framework to

the GAIN use case. The Generator and Discriminator are trained adversarially and independently. We summarize our loss functions and optimization goals below.

$$\mathcal{L}_D(\mathbf{m}_j, \hat{\mathbf{m}}_j, \mathbf{b}_j) = \frac{1}{d_o} \sum_{k:c_{jk}=1} \hat{m}_{jk} - \frac{1}{d_m} \sum_{k:c_{jk}=0} \hat{m}_{jk},$$

$$\text{where } d_o = \| \mathbb{I}(\mathbf{c}_j = \mathbf{1}) \|_1 \text{ and } d_m = \| \mathbb{I}(\mathbf{c}_j = \mathbf{0}) \|_1$$

$$\mathcal{L}_G(\mathbf{m}_j, \hat{\mathbf{m}}_j, \mathbf{b}_j) = -\frac{1}{d_m} \sum_{k:c_{jk}=0} \hat{m}_{jk}$$

$$\text{where } d_m = \| \mathbb{I}(\mathbf{c}_j = \mathbf{0}) \|_1$$

$$\mathcal{L}_M(\mathbf{x}_j, \mathbf{x}'_j) = \sum_{k=1}^{d} m_{jk} L_M(x_{jk}, x'_{jk})$$

$$L_M = \begin{cases} 0.2(x'_{jk} - x_{jk})^2 \text{ if } x_{jk} \text{ is continuous} \\ 0.8(-x_{jk} \log(x'_{jk})) \text{ if } x_{jk} \text{ is binary} \end{cases}$$

The goal of our meta-learning algorithm is to train the model so it can quickly adapt to a new domain using only a few datapoints and training iterations. To accomplish this, we split our process into two parts: a meta-training task and a meta-testing task. Given a batch of domains $\{\mathcal{D}_i\}_{i=1}^N$ drawn from $p(\mathcal{D})$, we can divide each domain in a set of datapoints S_i^S and another set S_i^Q, called source set and query set respectively. The two sets are disjoint($S_i^S \cap S_i^Q = \emptyset$).

Let's go through the optimization procedure of the Generator G_ϕ. The meta-train task tries to optimize in two separate loops: an inner loop, that performs a gradient update according to the source set S_i^S, and an outer loop, that performs the gradient update according to the performance on the query set S_i^Q.

More specifically, in the inner loop of the meta-train task, we first sample n i.i.d. samples from the domain to form the source set S_i^S, then we calculate a gradient update step as follows

$$\phi'_i = \phi - \alpha \nabla_\phi \mathcal{L}_G(S_i^S; \phi) + \mathcal{L}_M(S_i^S; \phi)$$

Where $\mathcal{L}(S_i^S; \phi)$ indicates the loss function is calculated on the elements of the source set S_i^S with parameters ϕ. This update can be repeated multiple times for one source set S_i^S. We will calculate the updated parameters ϕ'_i for each one of the domains $\mathcal{D}_i$.

In the outer loop of the meta-training step, we evaluate each one of the updated weights ϕ'_i on the corresponding query set S_i^Q for the domain $\mathcal{D}_i$. This will help the model understand and measure how well it is adapting to each domain, allowing it to adjust its weights accordingly.

Algorithm 1 Meta-GAIN pseudo-code

Input: Distribution over domains $p(\mathcal{D})$
Initialize: $\theta, \phi, \alpha, \beta, \gamma, c, s_D$
repeat
  (1) Discriminator Optimization
  for t = 1, ... , s_D **do**
    Sample a batch of domains $\mathcal{D}_i \sim p(\mathcal{D})$.
    for all $\mathcal{D}_i$ **do**
      Sample source and query set from $\mathcal{D}_i$
      $S_i^S = \{(\tilde{\mathbf{x}}_j^S, \mathbf{m}_j^S)\}_{j=1}^n$
      $S_i^Q = \{(\tilde{\mathbf{x}}_j^Q, \mathbf{m}_j^Q)\}_{j=1}^m$
      for j = 1, ..., n **do**
        $\bar{\mathbf{x}}_j \leftarrow G_\phi(\tilde{\mathbf{x}}_j, \mathbf{m}_j, \mathbf{z}_j)$
        $\hat{\mathbf{x}}_j \leftarrow \mathbf{m}_j \odot \tilde{\mathbf{x}}_j + (\mathbf{1} - \mathbf{m}_j) \odot \bar{\mathbf{x}}_j$
        $\mathbf{h}_j = \mathbf{b}_j \odot \mathbf{m}_j + 0.5(\mathbf{1} - \mathbf{b}_j)$
      end for
      (1.1) Perform update on source set S_i^S
      $g_\theta \leftarrow \nabla_\theta \sum_{j=1}^n \mathcal{L}_D(\mathbf{m}_j, D_\theta(\hat{\mathbf{x}}_j, \mathbf{h}_j), \mathbf{b}_j)$
      $\theta'_i \leftarrow \theta + \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
      $\theta'_i \leftarrow \text{clip}(\theta'_i, -c, c)$
    end for
  end for
  (1.2) Perform meta-update on query set S_i^Q **of each** $\mathcal{D}_i$
  $g_\theta \leftarrow \nabla_\theta \sum_i \sum_{j=1}^m \mathcal{L}_D(\mathbf{m}_j, D_{\theta'}(\hat{\mathbf{x}}_j, \mathbf{h}_j), \mathbf{b}_j)$
  $\theta \leftarrow \theta + \beta \cdot \text{RMSProp}(\theta, g_\theta)$
  $\theta \leftarrow \text{clip}(\theta, -c, c)$
  (2) Generator Optimization
  Sample a batch of domains $\mathcal{D}_i \sim p(\mathcal{D})$.
  for all $\mathcal{D}_i$ **do**
    Sample source and query set from $\mathcal{D}_i$
    $S_i^S = \{(\tilde{\mathbf{x}}_j^S, \mathbf{m}_j^S)\}_{j=1}^n$
    $S_i^Q = \{(\tilde{\mathbf{x}}_j^Q, \mathbf{m}_j^Q)\}_{j=1}^m$
    for j = 1, ..., n **do**
      $\mathbf{h}_j = \mathbf{b}_j \odot \mathbf{m}_j + 0.5(\mathbf{1} - \mathbf{b}_j)$
    end for
    (2.1) Perform update on source set S_i^S **with fixed** D
    $g_\phi \leftarrow \nabla_\phi \sum_{j=1}^n \mathcal{L}_G(\mathbf{m}_j, \hat{\mathbf{m}}_j, \mathbf{b}_j; \phi) + \gamma \mathcal{L}_M(\mathbf{x}_j, \tilde{\mathbf{x}}_j; \phi)$
    $\phi'_i \leftarrow \phi + \alpha \cdot \text{RMSProp}(\phi, g_\phi)$
    $\phi'_i \leftarrow \text{clip}(\phi'_i, -c, c)$
  end for
  (2.2) Perform meta-update on query set S_i^Q **of each** $\mathcal{D}_i$
  $g_\phi \leftarrow \nabla_\phi \sum_i \sum_{j=1}^m \mathcal{L}_G(\mathbf{m}_j, \hat{\mathbf{m}}_j, \mathbf{b}_j; \phi'_i) + \gamma \mathcal{L}_M(\mathbf{x}_j, \tilde{\mathbf{x}}_j; \phi'_i)$
  $\phi \leftarrow \phi + \beta \cdot \text{RMSProp}(\phi, g_\phi)$
  $\phi \leftarrow \text{clip}(\phi, -c, c)$
until train loss is converged

Overall, in the outer loop, we are trying to optimize for

$$\min_{\phi} \sum_{i} \mathcal{L}(S_i^Q; \phi_i')$$

Overall, we first meta-train our Discriminator(inner and outer loop) over a batch of domains $\{\mathcal{D}_i\}_{i=1}^N$. We repeat this process 5 times(parameter s_D in the pseudo-code) to improve stability. We then fix the parameters of the Discriminator and optimize the Generator. Assuming domains at meta-training time and meta-test time display a similar kind of distribution shift, imputation performance should achieve good results also at meta-test time.

5. Experiments

Our experiments are designed to answer the following questions:

1. Can Meta-GAIN enable fast adaptation to new domains? How many samples from each domain does Meta-GAIN need to achieve accurate performance?
2. How does Meta-GAIN perform compared to GAIN? We compare performance across a variety of settings(i.e. varying amounts of data, varying missing rates, varying number of dimensions).
3. Does Meta-GAIN perform better when few samples are presented from many domains, or when we have many samples from a few domains?
4. Does the Wasserstein distance help avoid mode collapse and does it make training more stable?
5. How does Meta-GAIN perform when domains are unbalanced(i.e. some domains have many and others few datapoints)
6. Can Meta-GAIN be used in an online-learning context where it is constantly updating and learning from the data it is provided?
7. How does Meta-GAIN perform when the goal is to perform prediction on the imputed dataset?

We will evaluate Meta-GAIN against other benchmarks using RMSE, AUROC and proportion of falsely classified entries (PFC) metrics, along with their standard deviation across 10 trials.

Performance will also be compared across different types of missing data mechanisms(MCAR, MAR, MNAR).

We will compare Meta-GAIN performance against the following benchmarks: GAIN, MICE(van Buuren & Groothuis-Oudshoorn, 2011), MissForest(Stekhoven & Bühlmann, 2011), Auto-Encoder, MisGAN(Li et al., 2019), MIWAE(Mattei & Frellsen, 2019) and HI-VAE(Nazabal et al., 2020).

We first evaluate Meta-GAIN on the Omniglot(Lake et al., 2015) dataset. We will then test in on Rotated and Translated MNIST datasets(LeCun & Cortes, 2010), as well as the UCI datasets(Dua & Graff, 2017) GAIN originally experimented on(Breast, Spam, Letter, Credit, and News).

In Ivanov et al.'s experiments, one of the most compelling examples of GAIN weaknesses is the horizontal line mask test over MNIST(i.e. only one horizontal line is visible, the rest of the image is missing). The original GAIN implementation fails to learn conditional distribution for the horizontal mask, while other implementations, such as VAEAC and Universal Marginalizer (Douglas et al., 2017) succeed. We will replicate this experiment on our Meta-GAIN model to verify if the issue persists.

6. Conclusions

Our main contribution is the Meta-GAIN algorithm, a novel version of GAIN that relies on meta-learning to be effective in low-data settings. Our approach has several benefits. It doesn't add any additional parameters to the main model while making it much more flexible and adaptable to many different contexts. Our model can integrate knowledge from different domains while adapting its prediction to a specific one with only a few examples.

Furthermore, the use of the Wasserstein distance allows for more stable and easy training that helps to avoid mode collapse.

7. Future Work

This work presents many interesting future directions.

First, we would like to complete the code implementation of the proposed Meta-GAIN method and run experiments to evaluate Meta-GAIN performance. This would allow hyperparameter tuning for robust training across multiple tasks and a better understanding of the model's strengths and weaknesses.

Second, in this paper, we use MAML to achieve domain adaptation. While this is approach can improve model performance in low data settings, other more advanced meta-learning techniques exist specifically to address domain adaptation. An interesting approach is the one from (Zhang et al., 2020), where a meta-model is created to quickly adapt the prediction model using unlabelled examples. It especially focuses on how meta-learning can address distribution shift. The meta-model is creating a "context" vector $\mathbf{c}$ that feeds into the prediction model. The model can then use the vector $\mathbf{c}$ to infer additional information about the input

distribution.

Third, in this work, we couldn't properly address the issue of heterogeneous domains, and we assumed all the domains considered shared the same feature space. This is unfortunately often untrue, as hospitals have different record-keeping procedures and might record different types of information. Although our model could be used in this context if non-recorded features are considered as missing, this would not be an optimal solution. Adding an extra network to deal with domain heterogeneity and map all the feature spaces to the same latent space would be a compelling direction of research. Some research already exists in this area(Yoon et al., 2018a), but it would be interesting to have something designed with the meta-imputation use case in mind.

Fourth, the literature built around techniques to avoid mode collapse and make GAN training more stable is vast. Future work could explore this area further and maybe try to implement a Reconstructor network such as the one proposed by Srivastava et al. in their VEEGAN paper. A preliminary sketch of what the updated network would look like is provided in Figure 1.

Finally, if any negative transfer shows in the experiments(learning from multiple domains deteriorates the model performance in domains where data is abundant), multiple options could be explored. Wang et al. explain and analyze the problem in-depth in the case of modern multilingual models trained on multiple languages. Another fascinating proposal is the one from Yu et al., where detrimental gradient interference across task is solved with a form of gradient surgery: if the two gradients are conflicting a task's gradient is projected onto the other task's normal plane. In their experiments, this method leads to substantial gains in efficiency and performance.

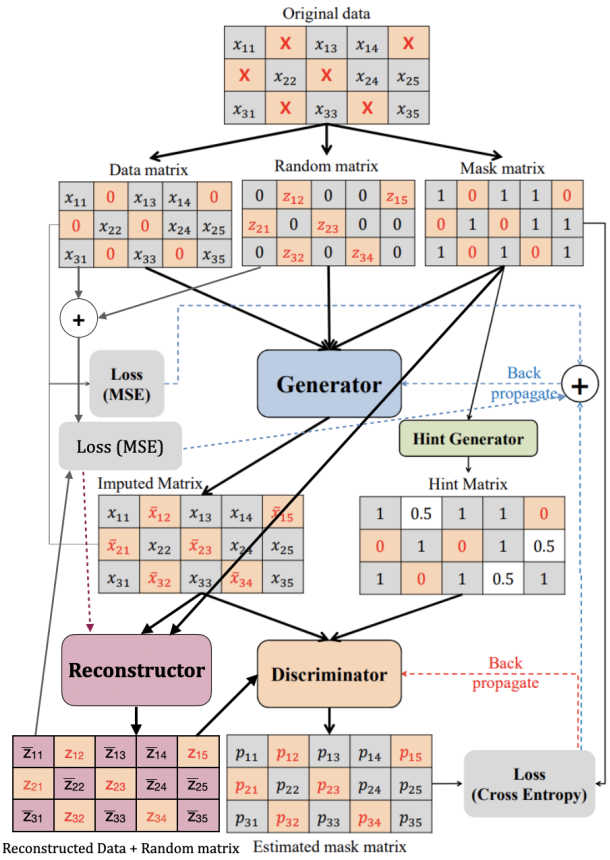

Figure 1. Sketch of proposed future extension for Meta-GAIN
A Reconstructor network is added to the architecture. The Reconstructor is trained to map the generated data back to the random matrix(generated from a Gaussian distribution) and approximately reverse the function of the Generator network. In the original implementation, the Reconstructor model is trained to map from true data back to the Gaussian distribution. Training the model this way encourages the generator network to map from the noise distribution to the entirety of the real data distribution, thus producing diverse samples and resolving mode collapse.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent, 2016.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

Bellot, A. and van der Schaar, M. Boosting transfer learning with survival data from heterogeneous domains. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 57–65. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/bellot19a.html.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. A theory of learning from different domains. *Machine Learning*, 79:151–175, 05 2010. doi: 10.1007/s10994-009-5152-4.

Dou, Q., Castro, D. C., Kamnitsas, K., and Glocker, B. Domain generalization via model-agnostic learning of semantic features, 2019.

Douglas, L., Zarov, I., Gourgoulias, K., Lucas, C., Hart, C., Baker, A., Sahani, M., Perov, Y., and Johri, S. A universal marginalizer for amortized inference in generative models, 2017.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Ipsen, N. B., Mattei, P.-A., and Frellsen, J. not-miwae: Deep generative modelling with missing not at random data, 2020.

Ivanov, O., Figurnov, M., and Vetrov, D. Variational autoencoder with arbitrary conditioning, 2019.

Lake, B., Salakhutdinov, R., and Tenenbaum, J. Human-level concept learning through probabilistic program induction. *Science*, 350:1332 – 1338, 2015.

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

Lee, C., Zame, W., Yoon, J., and van der Schaar, M. Deephit: A deep learning approach to survival analysis with competing risks, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16160.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization, 2017.

Li, S. C.-X., Jiang, B., and Marlin, B. Misgan: Learning from incomplete data with generative adversarial networks, 2019.

Ma, C., Tschiatschek, S., Hernández-Lobato, J. M., Turner, R., and Zhang, C. Vaem: a deep generative model for heterogeneous mixed type data, 2020.

Mattei, P.-A. and Frellsen, J. Miwae: Deep generative modelling and imputation of incomplete data, 2019.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks, 2017.

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. Handling incomplete heterogeneous data using vaes, 2020.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Rubin, D. B. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Shen, L., Zhu, W., Wang, X., Xing, L., Pauly, J. M., Turkbey, B., Harmon, S. A., Sanford, T. H., Mehralivand, S., Choyke, P., Wood, B., and Xu, D. Multi-domain image completion for random missing input data, 2020.

Smieja, M., Struski, L., Tabor, J., Zielinski, B., and Spurek, P. Processing of missing data by neural networks. *CoRR*, abs/1805.07405, 2018. URL http://arxiv.org/abs/1805.07405.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning, 2017.

Stekhoven, D. J. and Bühlmann, P. MissForest—nonparametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 10 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr597. URL https://doi.org/10.1093/bioinformatics/btr597.

Thrun, S. and Pratt, L. *Learning to Learn: Introduction and Overview*, pp. 3–17. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

Tieleman, T. and Hinton, G. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude., 2012.

van Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software, Articles*, 45(3):1–67, 2011. ISSN 1548-7660. doi: 10.18637/jss.v045.i03. URL https://www.jstatsoft.org/v045/i03.

Wang, Z., Lipton, Z. C., and Tsvetkov, Y. On negative interference in multilingual models: Findings and a meta-learning treatment, 2020.

Yoon, J., Jordon, J., and van der Schaar, M. Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks, 2018a.

Yoon, J., Jordon, J., and van der Schaar, M. GAIN: missing data imputation using generative adversarial nets. *CoRR*, abs/1806.02920, 2018b. URL http://arxiv.org/abs/1806.02920.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning, 2020.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: A meta-learning approach for tackling group shift, 2020.