SpatialGAIN: Spatial Generative Adversarial Imputation Networks and its Application to Zero-Shot Pixel Imputation

Justin Young
Stanford University
450 Serra Mall, Stanford, CA
justiny@stanford.edu

Abstract

In this paper we introduce SpatialGAIN (Spatial Generative Adversarial Imputation Networks). We consider a setting where pixels in an image are corrupted. We repurpose the generative adversarial network (GAN) framework to impute these pixels of interest. This work builds upon the generative adversarial imputation networks (GAIN) literature first introduced by Yoon et al. (2018) [1]. We provide two main contributions: (1) we introduce a new architecture for pixel imputation that is shown to outperform the baseline GAIN method on downsampled ImageNet data; (2) we propose a zero-shot imputation technique as an alternative to traditional statistical estimators like matrix completion by using SpatialGAIN.

1. Introduction

The need for image imputation arises when images are of low quality (i.e. upscaling) or if pixels are missing. As many downstream tasks require complete or high-resolution images, it is of substantive interest to impute their values but doing so remains a non-trivial task. Leading approaches have become increasingly sophisticated in exploiting the image structure to infer missing pixels, but the majority of methods have only focused on the setting where values are missing at random. In addition, almost all methods require many training images, as the models intuitively rely on the structure learned from other images to impute the missing pixels of the image at hand.

Our primary goal is to investigate pixel imputation in the zero-shot setting where missing data has a particular pattern. Namely we artificially corrupt rows of pixels such that a long sequence (e.g. last ten pixels) is missing. This serves mainly as a first pass to better understand the underlying challenges before allowing for richer patterns of missing data. More generally, such patterned missingness may occur naturally whenever higher resolution is needed.[1] For example, physicians may require finer detail when examining specific regions in a given medical image (e.g. meniscus in a knee MRI). Especially in settings where no (or very few) additional training images are available, finding generic tools to impute these missing pixels poses an interesting challenge.

In order to explore this goal, we first must introduce a new architecture, called SpatialGAIN (Spatial Generative Adversarial Imputation Networks). We consider a setting where pixels in an image are corrupted, and we leverage the adversarial GAN structure to impute these pixels of interest. This work builds upon the GAIN framework first introduced by Yoon et al. (2018) [1]. We believe that the adversarial structure, coupled with a convolutional architecture, will allow the model to learn a deep understanding of images and the spatial dependencies between pixels. Introducing this architecture and comparing it to the baseline GAIN comprises what we will refer to as **Experiment 1**. We then switch gears and discuss the zero-shot setting by fixing a single image. In **Experiment 2**, we induce localized missingness to this image and cut up the remaining image into subsets that act as training data to apply SpatialGAIN to the target area of interest. In either experiment, the (test) input to our algorithm is an image that is partially corrupted. We then use a our SpatialGAIN method to output a completed image.

On a high-level our contributions can be summarized as follows:

[1] We note that pixel imputation and super-resolution are distinctly different challenges. While the former deals with missing data, the latter often details with some form of learnable up-scaling. However, these two fields are closely tied together and tackle similar problems. In a real-world setting where a particular region of a medical image is blurred, combining both techniques may be a fruitful line of research. For instance, while missing pixel imputation operates under the premise of using surrounding spatial information to impute pixels with *no* information, one could imagine we could reap performance gains on imputing low-resolution regions by using the information in those regions. Thus, the work here provides a preliminary step in exploring these issues.

1. We provide a flexible PyTorch Module based API [2] implementation of Yoon et al. (2018).

2. We extend Yoon et al. (2018) to allow for CNNs (i.e. SpatialGAIN).

3. As a first-pass baseline, we compare SpatialGAIN's performance on ImageNet data against GAIN and show that it outperforms it.

4. We apply SpatialGAIN to the zero-shot setting by artificially constructing a training data set by taking random crops of an image of interest.

We note that this last zero-shot exercise seems to be quite distinct, in that **Experiment 2** may seem unrelated to **Experiment 1**. Namely, zero-shot imputation is usually left to statistical methods due to the obvious lack of training data. However, performing better with a deep-learning method would be a novel approach that may lead to other interesting questions. We believe it is a natural experiment and one that allows for deeper insight into the new proposed architecture. Good imputation results here take a first pass at understanding the inner workings of the proposed SpatialGAIN method. In some sense, we want to determine whether SpatialGAIN is learning (i) what an object actually looks like from training data; (ii) how to fill in missing pixels from adjacent values; (iii) the extent to which both are occurring. As a related goal, we artificially construct this zero-shot setting to also study different patterns of missing data. Data can be missing completely at random (MCAR) or missing not at random (MNAR), aand if imputation is successful in one but not the other, that also tells us something about the successes or limitations of SpatialGAIN.

Related Work

Imputing missing values more generally is an important problem that has been long studied by researchers in statistics, computer science, and econometrics. For instance, when a program is administered without a randomized control trial, evaluating the causal average treatment effect requires knowledge of the counterfactual, which is fundamentally unknowable.[2] In other words, if a treatment is applied at time $t = t_0$ for unit i, we observe outcome $Y_{it_0}, \dots Y_{iT}$ but we never observe these outcomes had treatment not happened. Econometricians have consequently developed a wide range of analytic tools specifically for this setting ([3], [4]). Another approach arising in statistics is *matrix completion*, where we try to recover a matrix from a partially (potentially noisily) observed one ([5]). The popular use-case is recommender systems, where we observe users' ratings over certain movies but not all.

[2]Recall a difference in means estimator in the absence of a randomized control trial would be generically biased.

The above examples and corresponding literature inspire our approach to the task at hand for images. We recognize there are considerable differences between datasets and images: unlike image data, the local/global dependencies of datasets are often broken down only along the *horizontal* axis, ignoring more complex spatial dependencies. For instance, in panel data settings like the one above, often we consider variation along the time axis (horizontal), but the unit axis (vertical) is often ignored as observations are interchangeable. It is not immediately obvious whether these statistical approaches can be applied to image data as the underlying patterns are different. Indeed, CNNs have dominated image tasks in part due to the fact that convolving the filter with the image captures local spatial dependencies in a fundamentally different way. In this project we hope to explore how leading imputation models from computer science generalize to different types of data (i.e. images vs. panel data) and different patterns of missingness.

From the computer science literature, we will draw primarily on Yoon et al. (2018) [1], who consider modifying the generative adversarial network (GAN) framework to impute missing values in a zero-shot setting. From the econometrics/statistics literature, [6], [7], [8], [3], and [5] all provide regression-based or matrix completion-based methods for exploiting data dependencies along both matrix axes to impute missing values. In addition, [8] extend empirical and theoretical matrix completion results to this same missing data regime.

The vast majority of the literature described above yield statistical-based methods for imputation that do not have additional *training data*, which is fundamental to modern machine learning methods. Successful imputation methods in this literature have often been computationally prohibitive [9], but less intensive approaches like denoising autoencoders have been proven to show decent imputation results. Our focus for this paper is to provide a relatively computationally cheap alternative to such approaches that still produces decent imputation.

2. Methods I: SpatialGAIN

As one of our big contributions is introducing SpatialGAIN, and because the later results rest upon it, we devote this section to describing it. We first outline the framework of vanilla GAIN architecture and then discuss the differences in our proposed architecture.

Generative Adversarial Imputation Networks

Generative Adversarial Imputation Networks (GAIN) are first introduced by [1]. As our work builds upon the GAIN framework, we begin by outlining its basic set-up. GAIN considers a setting where the data arises from a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_d$. They consider an augmented space such that along each dimension we union

$\mathcal{X}_i \cup \{*\}$, where $\{*\}$ denotes a missing value.[3] The resuling space is thus $\tilde{\mathcal{X}} = \tilde{\mathcal{X}}_1 \times \ldots \times \tilde{\mathcal{X}}_d$. The authors introduce a mask vector $\boldsymbol{M} = (M_1, \ldots, M_d)$, which is a random variable where each $M_i \in \{0, 1\}$ governs whether a value is observed ($M_i = 1$) or not ($M_i = 0$).

The setting of interest is one where we have corrupted realizations of random variables $\boldsymbol{X} = (X_1, \ldots, X_d) \in \mathcal{X}$ distributed according to $P(\boldsymbol{X})$, so we actually observe data $\tilde{\boldsymbol{X}} = (\tilde{X}_1, \ldots, \tilde{X}_d) \in \tilde{\mathcal{X}}$, where each $\tilde{X}_i$ is given by

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

Thus, the data is generated from $\mathcal{X}$, corrupted deterministically by a realized $\boldsymbol{M} = \boldsymbol{m}$, and we observe $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ from which we can back out the realized missing vector $\boldsymbol{m}$. The dataset $\mathcal{D} = \{(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{m}^{(i)})\}_{i=1}^n$ contains i.i.d. copies from $\tilde{\boldsymbol{X}}$ and $\boldsymbol{M}$. GAIN imputes unobserved values by implicitly learning conditional distribution $P(\boldsymbol{X}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}^{(i)})$. By modeling the distribution and not simply a point estimate, this method naturally allows for multiple draws and inference.

Generator

GAIN builds upon the standard GAN structure but makes several key changes. With corrupted data $\tilde{\boldsymbol{X}}$ described above, noise $\boldsymbol{Z} \in [0, 1]^d$, and the mask $\boldsymbol{M} \in \{0, 1\}^d$, the generator $G : \tilde{\boldsymbol{X}} \times M \times Z \to \mathcal{X}$ implicitly learns the conditional distribution $P(\boldsymbol{X}|\tilde{\boldsymbol{X}})$ by imputing the missing entries from white noise.[4] In particular, the generator outputs a vector of imputations

$$\boldsymbol{X}^{imp} = G(\tilde{\boldsymbol{X}}, \boldsymbol{M}, (\boldsymbol{1} - \boldsymbol{M}) \odot \boldsymbol{Z})$$

Note this sample generates imputations for all data, including for observed data. The final output to the discriminator is given by the completed vector:

$$\boldsymbol{X}^{out} = \boldsymbol{M} \odot \tilde{\boldsymbol{X}} + (1 - \boldsymbol{M}) \odot \boldsymbol{X}^{imp}$$

Thus, $\boldsymbol{X}^{out}$ represents an example whose observed entries are kept in place and whose missing values are imputed.

Discriminator

Unlike the Vanilla GAN or DCGAN set-up, the discriminator must make multiple predictions for a single output. In particular, the discriminator takes in d-dimensional inputs and predicts entry-wise probabilities for each of the d features. The discriminator tries to learn the underlying mask matrix, predicting 1 when $M_i = 1$ and 0 when $M_i = 0$.

However, a simple discriminator $D : \mathcal{X} \to [0, 1]^d$ is insufficient for the generator to learn a unique conditional distribution. For theoretical and practical purposes, a hint vector $\boldsymbol{H} \in \mathcal{H}$ must be introduced that gives the discriminator additional information on where the 1s and 0s are in $\boldsymbol{M}$. Intuitively there is too much *freedom* in the generator's imputation, as the discriminator's loss is calculated over all entries and not simply the imputed ones. This is to say that errors on the "real" entries can wash out the signal obtained from errors on the "fake" entries, precluding G's convergence to the desired distribution determined by the underlying true data. Without a hint, the generator can easily fool the discriminator with a distribution that is very far from the true distribution.[5] From an implementation standpoint, I speculate giving this hint also speeds up convergence. As D must make fine-grained predictions in *every* entry, it is easy to surmise how the discriminator can be very weak in the absence of $\boldsymbol{h}$.

Formally the authors draw $\boldsymbol{h} \sim g(\boldsymbol{H}|\boldsymbol{M})$ from a prespecified conditional distribution $g(\cdot\,|\boldsymbol{M})$. In the paper they define random variable $\boldsymbol{B} \in \{0, 1\}^d$ as an all-ones vector with one zero entry that is uniformly chosen. The hint is generated by the function:[6]

$$\boldsymbol{H} = \boldsymbol{B} \odot \boldsymbol{M} + 0.5(\boldsymbol{1} - \boldsymbol{B})$$

Note that when $h_i = 1$, the hint perfectly reveals to the discriminator the veracity of the given entry, but when $h_i = 0.5$, no information is given. It is clear that $\boldsymbol{H}$ is a pivotal part of this architecture that controls the flow of information of $\boldsymbol{M}$ to the discriminator. Thought of another way, this hint mechanism can be seen as reducing the dimensionality the discriminator's problem. In whole the discriminator is given by $D : \mathcal{X} \times \mathcal{H}_{\boldsymbol{M}} \to [0, 1]^d$.

SpatialGAIN

With the basic structure outlined, we now turn to our proposed architecture, SpatialGAIN, which extends GAIN to image data. The major departure comes from the network architecture as well as the data masking and hinting processes.

[3] Formally this denotes a point not in any $\mathcal{X}_i$.

[4] The exact dimension of $\boldsymbol{Z}$ is not of considerable interest as the model learns the implicit map between $\boldsymbol{Z}$ and the data, and transformations of $\boldsymbol{Z}$ are common in GAN architectures. The authors let $\boldsymbol{Z} \in \mathbb{R}^d$ for simplicity. Note however that the noise fed into each training example ends up being $\|\boldsymbol{1} - M\|_1$-dimensional.

[5] This point is subtle but worth exploring more in future work. In particular, some theoretical work has been done on the distribution GANs learn (Arora et al. (2017)), and it has been shown that generically GANs need not converge to the true data distribution. It would be of considerable interest to examine whether convergence to the true data distribution is slower or otherwise not guaranteed when the discriminator's output is multidimensional like in the GAIN setting.

[6] Note this choice of $\boldsymbol{B}$ implicitly defines $g(\cdot\,|\boldsymbol{M})$

Model Architecture

The generator and discriminator architectures are inspired by DCGAN and U-Net. In contrast to the DCGAN and similar to the vanilla GAIN model, the discriminator does not simply output a unidimensional probability, but rather probabilities for all pixels across all channels, as it needs to classify which pixels are imputed and which are real. We find that the model performs decently well when the discriminator and generator share the same architecture. We consider two specifications, denoted as *Standard* or *Downsample/Upsample* architectures:

1. (Standard): Input, [Conv-LeakyReLu-BN]x3, Conv, Sigmoid[7]

2. (Downsample / Upsample): Input, [Conv-LeakyReLu-BN], [Conv-LeakyReLU-MaxPool-BN], [Conv-LeakyReLu-BN], [ConvTranspose-BN], Conv, Sigmoid

We use the Standard specification for the baseline tests of SpatialGAIN. As GAIN shows significant improvement over other leading methods like autoencoders and random forests in tabular data, it seems natural to compare SpatialGAIN to GAIN in both tabular and image data settings as a baseline sanity check. We defer the Downsample/Upsample architecture for the zero-shot setting described later.

Although our framework can deal with RGB channels, as a first pass we consider only grayscale images where $C = 1$. The rationale is that color channels are more likely than not to complicate the model, and because in many important settings salient to this research like medicine, grayscale images are the norm. It is however very straightforward to use the implementation with all color channels, but this is not the main focus of our paper here and we defer this to future work.

We extend the basic missingness framework to images in a straightforward fashion, making small changes. Notably, an altered *spatial mask* is needed with images. A naive approach would simply mask in the same way the vanilla GAIN does, without regard to the separate color channels. We take note of this and leverage the fact that a missing pixel is not missing simply across one or two channels but across all. Furthermore, we choose to depart from the paper's hint mechanism. We introduce a simple *spatial hint* mechanism that simply provides upwards of 90% of the mask matrix to the discriminator. This mimics the idea of $\boldsymbol{B}$ in the original GAIN set-up, but it gives even more information to the discriminator.[8] This may also be desirable given that in images, the number of possible predictions the discriminator must make increase at the rate of $O(n^2)$ versus $O(n)$. I also slowly decayed the hint percentage with training, as intuitively we speculate the discriminator becomes stronger over time and requires less hint information. This is akin to the idea of changing the rates at which D and G learn in order to stabilize training.

Training

Just like in the standard GAN, G and D play a minimax game, where D tries to maximize correct (multi-output) classification and G tries to minimize D's success in doing so. The objective of SpatialGAIN is the same of that of GAIN:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{X}^{out},\boldsymbol{M},\boldsymbol{H}} \Big[\boldsymbol{M}^T \log D(\boldsymbol{X}^{out}, \boldsymbol{H}) + (\mathbf{1} - \boldsymbol{M})^T \log(\mathbf{1} - D(\boldsymbol{X}^{out}, \boldsymbol{H})) \Big]$$

SpatialGAIN is then trained in the same way as GAIN, which in turn is very similar to the alternating fashion standard in GAN training. We defer these details to [1]. The discriminator's loss is straightforward, as it is simply multi-output classification loss. The generator has two losses. One is the standard generator GAN loss $\mathcal{L}_G$ that deals with how well the discriminator distinguishes G's outputs from real data, i.e. this loss concerns the mapping from noise to imputation. The second loss is a reconstruction loss $\mathcal{L}_M$. Recall the generator predicts the full data vector $\boldsymbol{x}^{imp}$, so it's predicted values should match up with the ground-truth whenever available. Thus, we can view both losses as arising from prediction errors associated with wherever $M_i = 0$ or $M_i = 1$ respectively. The weights on each loss are hyperparameters; without loss, we can normalized the weight on $\mathcal{L}_G$ to 1 and tune the weight α on $\mathcal{L}_M$. Lastly, unlike GAIN which samples mini-batches from the training data with replacement (i.e. no epochs), we instead opt to do so without replacement (i.e. with epochs) as doing so helps convergence [10].

3. Methods II: Zero-Shot Imputation

Now, as an exercise of our novel architecture, we apply SpatialGAIN to the zero-shot setting. This is our second main contribution; we design a deep-learning method based only on one image (no training data) using SpatialGAIN and assess how it performs relative to existing statistical methods like Matrix Completion.

As we will work in a zero-shot setting, it is natural to also consider different types of missingness, as this may give

[7] Note all data is normalized to $[0, 1]$ so using $\sigma(\cdot)$ as the activation is appropriate for both pixel output (generator) as well as probability classification (discriminator).

[8] Indeed, the existing code implementation online takes this approach as well, with the author noting that the written approach and this implementation yield similar results. Testing both, I see no systematic differences between the two.

insight as to what SpatialGAIN succeeds in and what it fails at. The MCAR setting is straightforward; for the MNAR setting, we corrupt 20% of the rows, and for corrupted rows we delete the last ten pixels.

Zero-Shot SpatialGAIN

We consider the setting where an image is corrupted in a localized region, e.g. some 64×64 crop of the given image, denoted as $\boldsymbol{A}$. Then, we create a training set of data by taking 64×64 random crops of the same image, but with no overlap with $\boldsymbol{A}$. Formerly, we collect a set of "training images" $\{\boldsymbol{T}_k\}_{k=1}^n$ such that $\cup_{k=1}^n \boldsymbol{T}_k \cap A = \emptyset$. Intuitively, we will mask out random values of all T_k and use these as training data in order to train our SpatialGAIN, in an analogous fashion to the original GAIN. We do not allow for overlap of the target region $\boldsymbol{A}$ because in real settings, the corrupted parts of the image are unobserved and cannot be used for training. Next, we make an important observation: because different parts of the images could lead to very different subsets – i.e. the training data is likely not to be i.i.d. – we employ a simple nearest-neighbors type algorithm to prune away non-suitable training images. In particular we calculate the average pixel-wise Euclidean distance between observed pixels of $\boldsymbol{A}$ and those of all other T_k. Denote the mean and standard deviations of these average distances as $\bar{d}, \sigma$. With a hyperparameter $\beta \in \mathbb{R}$, we then discard training examples whose mean pixel distance from $\boldsymbol{A}$ is too far. Formally, our final set $\{\boldsymbol{T}_k\}$ consists of images such that

$$\forall \boldsymbol{T} \in \{\boldsymbol{T}_k\} : \quad d(\boldsymbol{A}, \boldsymbol{T}) \leq \bar{d} - \beta\sigma$$

We note that there is a significant trade-off between the size of local missingness and the overall image dimension. In particular, if $\boldsymbol{A}$ has dimensions $H_A \times W_A$, and the overall image has dimensions $H \times W$, an upper bound on the number of training examples can easily be derived as $n_{train} \leq (H - 2H_A + 1) + (W - 2W_A + 1) + (H - 2H_A + 1)(W - 2W_A + 1)$. As this tradeoff can only be exacerbated by the pruning procedure, we focus on relatively small localized missingness to mitigate small data issues, i.e. we focus on the case where $H_A \ll H$.

We also note that we modify D and G to downsample and upsample the images. The idea behind this is that the model may benefit by having a larger receptive field. There may be global attributes in the photo that may help in reconstructing any given localized region.

Matrix Completion Baseline

As a natural comparison, we consider how matrix completion – arguably the most prominent approach in statistics – performs on reconstructing corrupted image data in our zero-shot setting. Given an unknown "true" matrix $Z \in \mathbb{R}^{m \times n}$, an missing index matrix $\mathcal{O} = [\mathcal{O}_{ij}] \in \{0, 1\}^{m \times n}$, and data $X \in \{\mathbb{R} \cup \{*\}\}^{m \times n}$ with $\{*\}$ representing a missing value, a large literature has studied estimating Z from X. The observed values of X are interpretable as noisy analogues of Z, e.g. measurement error, such that for all (i, j) such that $\mathcal{O}_{ij} = 1$, $X_{ij} = Z_{ij} + \epsilon_{ij}$. The key assumption here is that the latent, unobservable Z is assumed to be low-rank. Intuitively, a potentially large number of missing values requires low-rank in order to accurately reconstruct the underlying matrix. In practice, we solve the following convex optimization problem:

$$\min_Z \frac{1}{2} \sum_{(i,j) \in \mathcal{O}} (L_{ij} - Z_{ij})^2 + \lambda \|Z\|_*$$

Here, $\|\cdot\|_*$ denotes the nuclear norm. This is akin to LASSO as it induces a sparse representation associated with a low-rank matrix.

4. Dataset and Features

Our main data comes from a downsampled version of ImageNet. In particular, we use 160×160 images for the SpatialGAIN baseline experiments and 320×320 images for the zero-shot experiments. These downsampled versions have the shortest side resized to their respective dimensions and preserve their aspect ratios. We pull this data from the library *Imagenette* put together by fast.ai [11]. The rationale is that this is a new algorithm, and testing this on a smaller data allows for faster iteration. Imagenette is a subsample of 10 easily classified classes from ImageNet. In particular we randomly choose to focus on class 0, or "tench" (a type of fish).

In the first experiment of testing SpatialGAIN's capabilities, we use two datasets to compare GAIN against SpatialGAIN. The first is UCI's Spambase dataset, which contains 57 continuous and categorical features and 4601 observations that have all been classified as spam emails. This is one of the datasets used in the original GAIN paper and thus provides a nice baseline with which we can compare our results. The second dataset comes from Imagenette. Our training set contains of 1080 160×160 images, and our test set contains 270 images of the same size. All images contain either a fish or someone holding a fish. As mentioned earlier, we only consider grayscale images in this paper for simplicity, so there are no color channels. We note that all data is already normalized to $[0, 1]$.

In the second experiment of imputing localized missingness in the zero-shot setting, we focus on a randomly chosen photo from the same class of photos. That is, we fix one image. However, because larger images are needed (see discussion earlier), we pull this image from the 320×320 library of Imagenette.

5. Experiments & Results

Experiment Overview

We provide the experiments and results for our two major contributions. For clarity, we denote our main aforementioned experiments as **Experiment 1** and **Experiment 2**. At the end of this section we also provide a brief section on hyperparameter and training details.

Experiment 1

This experiment tests whether SpatialGAIN "works" in the most basic sense. As a baseline, we compare it to GAIN to see whether it achieves better performance on spatial data. Other natural baselines exist as well, e.g. convolutional autoencoders. At the moment, we choose to focus on GAIN as it is a state-of-the-art imputation tool, and any outperformance proves *some* level of success of our proposed method.

First, we naively reshape the Spambase dataset into 8×8 "images"[9] and apply SpatialGAIN to see how it performs relative to GAIN. Next, we turn to Imagenette data. We let SpatialGAIN run naturally while we naively flatten the training images to vectors and run GAIN on those. The experiments are performed multiple times and the test set RMSE averaged. The mean RMSE and standard deviation is given in Table 1.

	GAIN	SpatialGAIN
Spam	0.051 (0.002)	0.109 (0.022)
Imagenette	0.171 (0.002)	0.076 (0.005)

Table 1. Experiment 1 Results: RMSE on Test Set

As expected, GAIN performs better on tabular data while SpatialGAIN performs better on image data. Interestingly, we see significant degradation of GAIN on image data while that of SpatialGAIN on tabular data is relatively modest. This suggests that SpatialGAIN works and is able to learn spatial dependencies well enough to impute them. In Figure 1, we show the quality of imputation after 20% of the pixels were artificially corrupted. Qualitatively the reconstructed image looks adequate. In contrast, as we can see in Figure 3, Vanilla GAIN performs much worse. Note that we present a different image here so the reader can see different training examples. Qualitatively the results are robust across almost all test images. Lastly, as another an interesting visualization, Figure 2 shows us that SpatialGAIN suffers with borders which intuitively makes sense. It uses surrounding spatial information to determine missing pixels, and borders are naturally places where the surroundings contrast with one another.

[9]Note there is some padding that occurs as the original spam dataset only has 57 features

(a) Original (b) Imputed

Figure 1. SpatialGAIN: Original vs Imputed Test Image, 20% Corruption

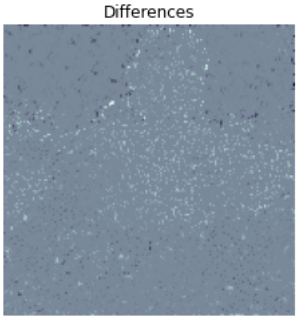


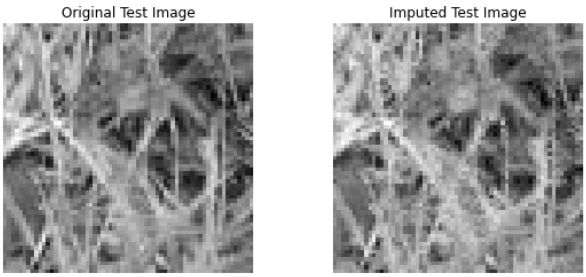
Figure 2. SpatialGAIN: Heat Map Showing $I_{original} - I_{imputed}$. The brighter the signal the higher the error.

(a) Original (b) Imputed

Figure 3. GAIN: Original vs Imputed Test Image, 20% Corruption

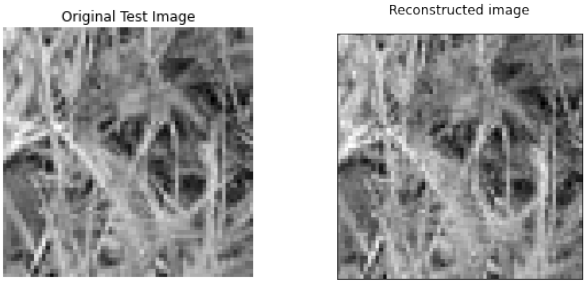
Experiment 2

This class of experiments can be best described as an application of SpatialGAIN to imputing data in the zero-shot setting. Armed with positive results from **Experiment 1**, it is reasonable to expect some success when we apply our proposed zero-shot imputation method. We test on both MCAR and MNAR data, as they both pose unique challenges. In particular, matrix completion is known to struggle with MNAR data, and thus it would be interesting to see whether SpatialGAIN can impute these adequately.



(a) Original (b) Imputed

Figure 4. Zero-Shot SpatialGAIN: Original vs Imputed Test Image, MCAR.



(a) Original (b) Imputed

Figure 5. Matrix Completion: Original vs Imputed Test Image, MCAR.

We consider two baselines. First, we modify an implementation of matrix completion that uses first-order optimization method FISTA (fast iterative shrinkage-threshold algorithm) to perform low-rank matrix completion to impute missing pixel values [12]. Second, we use a naive nearest neighbor algorithm to find the closest complete training image. The results are summarized in Table 2.

	SpatialGAIN	MC	NN
MCAR	0.090 (0.002)	0.054 (n/a)	0.143 (n/a)
MNAR	0.125 (0.013)	0.021 (n/a)	0.151 (n/a)

Table 2. Experiment 2 Results: RMSE on $\boldsymbol{A}$, MCAR and MNAR.

These results are for one particular image, where the localized missing subset $\boldsymbol{A}$ is the bottom-rightmost corner.[10] These errors suggest that this zero-shot method of using SpatialGAIN is not effective, but the reconstruction is still in the ballpark of reasonable. As an example, Figure 4 shows that the reconstruction qualitatively looks adequate and certainly better than a naive nearest neighbor approach, which fails the qualitative visual check. However, when compared with matrix completion methods as visualized in Figure 5, it is significantly poorer in quality. Intuitively this approach likely fails because there is not sufficient training data. Recall the underlying assumption of all ML models is that the train and test data come from the same distribution. Different subsets of an arbitrary image, even after pruning, are likely to be far too different for this method to work well.

Surprisingly, the matrix completion method for this particular image performs very well in the MNAR setting as well. We show the visualizations of MNAR performance in the Appendix in Figure 8. Based on the matrix completion literature, performing better in MNAR than MCAR is not expected. Indeed, as these results are only for one specific $\boldsymbol{A}$ within one specific training image, this pattern likely will not generalize. However, with some preliminary tests it does remain that matrix completion methods dominate SpatialGAIN at this zero-shot imputation task.

[10]This is arbitrarily chosen, but results are qualitatively similar if any other subset is chosen.

Hyperparameter and Training Details

For all experiments, we treat D and G symmetrically. We use the Adam optimizer and set the $lr = 0.001$, $(\beta_1, \beta_2) = (0.9, 0.999)$. For each experiment we set the mini-batch size at 64 or 128 depending on CUDA memory requirements. In addition, we employ the early-stopping criterion, with no experiment exceeding 15 epochs. We additionally re-run all experiments 5 times to obtain standard errors. We keep the default α weight on the reconstruction loss from the original GAIN paper: $\alpha = 10$. For missingness, we keep a constant 20% rate. Also, although He initialization is shown largely to work better in convolutional networks, Xavier initialization yielded lower RMSE consistently. Within the inner convolutional layers (recall D and G share the same structure), we chose channels of depth 64,32,16. This too was mostly guided by computational capacity. All tests were run on a Tesla V100 GPU. Loss curves for **Experiment 1** are shown in the Appendix.

6. Conclusion & Future Work

In this paper we have provided two main contributions. First, we introduced a novel adversarial architecture that imputes *image* data well relative to the state-of-the-art *tabular* baseline upon which it is based. In other words we have extended the GAIN model to image data by using convolutional networks to capture spatial dependencies.

However, a lot of work remains in checking the robustness of this model and understanding when and why it works well.

Training Limitations

Due to computing limitations, running these experiments more than 5 times for standard errors was cost-prohibitive.

In the future it would be of significant interest to explore these errors and determine how stable these point estimates are. In addition, little attention was paid to hyperparameter tuning in this study, as most parameters were kept the same from the original GAIN paper. This is unlikely to be the optimal scenario for our framework, as our architecture differs significantly.

Future Experiments

For our **Experiment 1**, we only used one class of data. Although it was randomly chosen and preliminary experiments into other classes suggest that our results are robust, a natural next step would be to explore the extent to which SpatialGAIN succeeds or fails on other classes. In addition, as this is a new method being proposed, many more baselines should be considered. Namely, a denoising convolutional autoencoder could provide a very natural baseline to SpatialGAIN, as the computational and data requirements are comparable. Exploring when an autoencoder outperforms the adversarial model and vice versa would be an interesting next step.

For our **Experiment 2**, future work should similarly check imputation across all different types of images. It is very likely that in images that have subsets that are "more i.i.d." we should expect to see greater success with our method. For instance, if given a simple image of a field or a fractal, it is easy to see how subsets are more similar to each other and can allow for more successful imputation.

Other Types of Data

As this project draws inspiration from a wide literature on imputation and can be extrapolated in theory to non-image data, as an alternative specification we could have considered a panel-data simulation. For a $N \times T$ panel data set with N observations and T time periods, we simulate a stochastic process given by:

$$Y_{i,t} = \rho Y_{i,t-1} + \delta_i + \gamma_t + \epsilon_{i,t}$$

We note that the underlying data structure is different between images and panel data: while images have a special spatial dependence along both rows and columns, the rows in panel data (i.e. observations) are not ordered. Theoretically one could employ some type of row clustering argument (e.g. nearest neighbor based on covariates) that could impose some local spatial dependence. This requires special care, but similar methods have been recently employed in the econometric literature [13]. This would take a first-pass into a larger literature on using GAN-type models in tabular data.

7. Contributions, Codebases, & Acknowledgements

This was a solo project. The vast majority of code for SpatialGAIN has been implemented from scratch in PyTorch. I note the original GAIN paper was written in TensorFlow [14], and an existing translation into PyTorch [15] exists online. However, I overwrote the large majority of this PyTorch translation from scratch and added model flexibility by writing with the PyTorch Module API. Most importantly, the bulk of the work here, i.e. modifying the architecture to allow for image data, is completely novel.

The code for matrix completion using FISTA was mostly borrowed [12], with small alterations to account for MNAR settings.

References

[1] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: missing data imputation using generative adversarial nets. *CoRR*, abs/1806.02920, 2018. 1, 2, 4

[2] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 2

[3] Jens Hainmueller Albert Abadie, Alexis Diamond. Synthetic control methods for comparative case studies: Estimating the effect of california's tobacco control program. *Journal of the American Statistical Association*, 105(490):493–505, 2010. 2

[4] Yudong Chen and Yuejie Chi. Harnessing structures in big data via guaranteed low-rank matrix estimation, 2018. 2

[5] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008. 2

[6] Jeffrey M. Wooldridge Guido W. Imbens. Recent developments in the econometrics of program evaluation. *Journal of Economic Literature*, 47(1):5–86, 2009. 2

[7] Nikolay Doudchenko and Guido Imbens. Balancing, regression, difference-in-differences and synthetic control methods: A synthesis. NBER Working Papers 22791, National Bureau of Economic Research, Inc, 2016. 2

[8] Susan Athey, Mohsen Bayati, Nikolay Doudchenko, Guido Imbens, and Khashayar Khosravi. Matrix completion methods for causal panel data models. *Journal of the American Statistical Association*, 116(536):1716–1730, may 2021. 2

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. 2

[10] Benjamin Recht and Christopher Re. Beneath the valley of the noncommutative arithmetic-geometric mean inequality: conjectures, case-studies, and consequences, 2012. 4

[11] Jeremy Howard et al. fastai. https://github.com/fastai/fastai, 2018. 5

[12] Ulrich Prestel. Matrix completion. https://github.com/uprestel/Matrix-Completion/, 2020. 7, 8

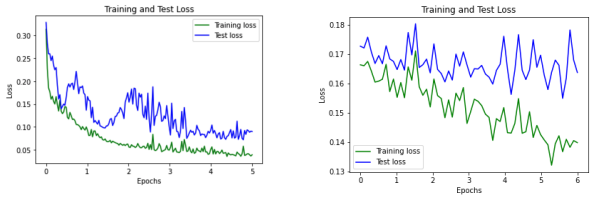
[13] Anish Agarwal, Munther Dahleh, Devavrat Shah, and Dennis Shen. Causal matrix completion, 2021. 8

[14] Jinsung Yoon. Codebase for generative adversarial imputation networks (gain). https://github.com/jsyoon0823/GAIN, 2020. 8

[15] Dhanajit Brahma. Generative adversarial imputation networks (gain) pytorch implementation. https://github.com/dhanajitb/GAIN-Pytorch, 2020. 8

Appendix

Loss curves for **Experiment 1** are shown in Figure 6. We note that loss curves for **Experiment 2** are not very informative and generally show flat losses, indicating that the model is not learning well in that setting.



(a) SpatialGAIN Loss (b) GAIN Loss

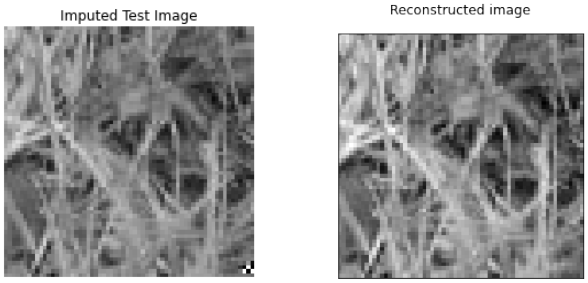
Figure 6. **Experiment 1**: Loss Curves for SpatialGAIN and GAIN on Imagenette (Class 0)

An additional example of SpatialGAIN imputation quality is given in Figure 7. We reiterate that the quality is consistent across the vast majority of test examples.

(a) Original (b) Imputed

Figure 7. **Experiment 1**: Original vs. Imputed Test Image, 20% Corruption

In Figure 8 we also provide additional visualization for **Experiment 2** into the MNAR reconstruction (both SpatialGAIN and matrix completion), for the same $\boldsymbol{A}$ described in the main text. As in the MCAR setting, matrix completion does a much better job.



(a) SpatialGAIN (b) Matrix Completion

Figure 8. Zero-Shot SpatialGAIN vs. Matrix Completion vs Imputed Test Image, MNAR.