Sequential Data Imputation with Evolving Generative Adversarial Networks

Haripriya Chakraborty
Ph.D. Program in Computer Science
City University of New York
The Graduate Center
New York, United States
Haripriya.Chakraborty@csi.cuny.edu

Priyanka Samanta
Ph.D. Program in Computer Science
City University of New York,
The Graduate Center
New York, United States
psamanta@gradcenter.cuny.edu

Liang Zhao
Computer Science Department
City University of New York
Lehman College
Bronx, United States
Liang.Zhao1@lehman.cuny.edu

***Abstract*—Data imputation is an important and widely researched problem in data analysis with applications to a variety of problem domains. Existing literature explores different methods to estimate missing values in a dataset. In this paper, we analyze the task of imputing multiple incomplete numerical datasets that arrive sequentially, each with an overlapping yet distinguishable set of features. We propose a novel procedure based on the Generative Adversarial Network (GAN) architecture, called EvoGAN, which uses transfer learning to build a unified model to perform imputation for all datasets in the sequence. Our methods exploit overlap between datasets and utilize the knowledge gathered from previous rounds. We validate our model on several sequences of datasets and find compelling results for compression rate and acceleration which outperform the state-of-the-art imputation models that are based on an input of a single dataset.**

***Index Terms*—Data Imputation, Missing Values, Generative Adversarial Network, Machine Learning, Deep Learning, Neural Network**

I. Introduction

The explosive growth in the volume of big data in recent years has fueled machine learning and statistical analysis in many scientific disciplines including astronomy, medicine, genomics, and information sciences [1]. Due to various factors such as incomplete measurements, entry mistakes, loss of records, or privacy concerns, the problem of missing data arises in many occasions, creating a very serious issue when it is not an option to simply discard the incomplete data. To adjust for the missing values, it has become generally accepted practice to use statistical methods for data imputation. For example, all three of the largest federally funded health surveys in the US - the Behavioral Risk Factor Surveillance System, the National Health and Nutrition Examination Survey, and the National Health Interview Survey - have employed item-specific formulas to estimate missing information on health risks and health behaviors [2], [3]. However, not all imputation methods are created equal. Therefore, it is crucial to develop imputation methods that can accurately grasp the relationships among variables across different datasets, since improper imputations may introduce systematic biases to the analysis, potentially causing immense harm to society.

This work was supported by NSF Grants CCF–1733834 and PSC CUNY Award 62413-00 50. All authors contributed equally.

Recently, researchers have studied the missing data imputation problem using Generative Adversarial Networks (GANs) [4]. A GAN model not only trains a generator network that predicts the missing values but also trains a discriminator network to provide good measurements on the imputation accuracy. This approach does not depend on any assumptions about the distribution of the data, and therefore it is favorable in many applications where the distribution of the data is unknown. Due to the immense flexibility of deep neural networks, GAN has shown impressive imputation results on images [5], numerical data [6], categorical data [7], and time-series data [8].

Transfer learning is a process by which a model can use information gathered from a previous task and utilize it to improve its performance a new but related task. The motivation for this is that, in real life, people constantly reuse knowledge already gathered from past experiences to find better solutions for new problems. For instance, someone who already possesses the ability to play the mandolin might have an easier time learning how to play the banjo than someone who has no prior experience playing an instrument. Transfer learning has been successfully applied to a variety of problems including image classification [9], text classification [10], biomedical imaging [11], and speech processing [12]. We explore the use of transfer learning in data imputation when there is a task involving multiple datasets that have some shared features.

In this work, we tackle the imputation task where the imputation model needs to be trained on a sequence of related yet unique datasets. This situation happens when the set of relevant variables evolves dynamically. For example, streaming services regularly add and remove TV shows and movies, which results in user rating datasets that are generated during different time periods having different sets of features. Similar issues occur in survey data [13] and observational data [14], where the set of survey questions or observable features change over time. While a portion of each dataset arriving at a given time may have significant overlap with datasets that have appeared before it, the features unique to each dataset prevent traditional models from fully exploring the relationships between different datasets. A naïve way to tackle this problem would be to train a new GAN from scratch

for each new dataset. However, it is rather inefficient in terms of storage and computational cost. Also, each model is only trained on one specific dataset, which does not make use of any information contained in other related datasets. It is also often infeasible to simply merge all these datasets. Our work is motivated by the fact that it is worthwhile to have a framework that does not require all the data to be available for conclusions to be derived. Sometimes, there might be a need to get some intermediate results rather than waiting for all the data to be seen. Also the merged data may get too large for storage and computationally expensive to process in one instance. Moreover, as we mentioned before, in real-life situations, data can be generated continuously which leads to new features being added and deleted periodically. For all these reasons, it is vital to find a smarter solution. In this paper we propose EvoGAN, a novel procedure that produces efficient models that learn at each instance and aggregate information learned at each step with previous steps to evolve over time. EvoGAN also employs model compression techniques to tackle redundancy, reduce network complexity, and improve generalization. To our knowledge, EvoGAN is the first imputation method that exploits the overlap of these datasets while also accommodating features unique to each dataset.

We test our procedure on different datasets and different settings and find that our best models are consistently efficient and achieve drastic compression rates with high imputation accuracy. We find that our model works for sequences of datasets that vary in length and for various degrees of missingness. We use data from Walmart and S&P 500 to show the value of our model in some real-world scenarios.

II. Related Literature

The problem of missing data has been considered for many years [15]. Substantial work has been done on various approaches to tackle the problem of missing data. Some approaches involve statistical methods like regression techniques or mean imputation [16], [17]. Some other approaches propose imputation techniques based on neural networks that learn the distribution of the training dataset to impute data [18]–[20]. There also exist state-of-the-art discriminative imputation methods [21]. In recent years, some work has been done on using generative adversarial networks (GAN) to perform data imputation [5]–[8]. Our work expands on this very idea.

Some of the work on data imputation is specific to problem domains. For instance, some are directed towards imputation algorithms with applications to medical research, while others are directed towards text data [22]–[25]. Some works approach imputation problems as an instance of the classical matrix completion problems. Therefore, matrix factorization based techniques can be used for imputation [26], [27]. Results in this area require explicitly defined completion objectives using matrix-related properties such as eigenvalues or norms, and therefore these methods are only useful in instances where the optimal imputation corresponds to nice linear algebraic properties.

The idea of networks that change dynamically with time has also been explored [28], [29]. These results aim to handle the changes in data distributions over a fixed set of variables. Our work considers a fundamentally different problem where the set of variables also changes dynamically. Some other solutions aim to improve the model performance rather than adapting to new data formats and do not take advantage of information common to all datasets while we look to exploit these commonalities using transfer learning [30].

The problem of catastrophic forgetting deals with sequential learning tasks and some work has been done in this regard [31]. However, this work does not extend to imputation tasks and scenarios where the features of the input varies over time. Some work has been done on using transfer learning for handling missing data. However, these approaches are substantially different from ours. For instance, some work focuses on using evolutionary algorithms to produce a good configuration of the neural network model and using transfer learning to update their model while only imputing a single dataset [32]. Some other methods are more suited for data with a small number of features where imputation is not the main goal [33]. Moreover, their model cannot be applied to a situation with sequential data since their model uses all the features while our model does not require all the features.

III. Notations and Problem Formulation

In our work, we analyze the task of imputing missing data that arrive sequentially using generative adversarial networks (GAN). We assume that the data is missing under the MCAR assumption. Consider a (possibly infinite) set of variables $\mathbf{S} = \{X_1, X_2, ...\}$, where X_i takes values from space $\mathcal{X}_i$ for $i = 1, 2, ...$ We suppose that datasets $(\mathcal{D}_1, \mathcal{D}_2, ...)$ arrive sequentially, so D_i is the dataset that appears in round i. Suppose $\mathcal{D}_i$ contains a finite subset $\mathbf{S}_i$ of $\mathbf{S}$. For any two consecutive datasets $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, we assume that $\mathbf{S}_i \cap \mathbf{S}_{i+1} \neq \emptyset$, otherwise the knowledge in $\mathcal{D}_i$ won't be useful on $\mathcal{D}_{i+1}$.

For each i, a data instance of dataset $\mathcal{D}_i$ is the concatenation of two vectors $\mathbf{x}$ and $\mathbf{m}$, where $\mathbf{x} \in \prod_{X_j \in \mathbf{S}} \mathcal{X}_j$ is the data vector with values for each variable in $\mathbf{S}_i$, and $\mathbf{m} \in \{0, 1\}^{\|\mathbf{S}_i\|}$ is the mask vector representing the incompleteness of the data. The mask vector $\mathbf{m}$ is defined so that a value in $\mathbf{m}$ is 1 if and only if the value of its corresponding variable in $\mathbf{x}$ is missing. We use $\mathbf{x}_m$ (resp. $\mathbf{x}_{um}$) to represent the sub-vector of $\mathbf{x}$ consisting of missing variables (resp. observable variables).

The goal of the data imputation task is to predict the unobserved values in each data vector $\mathbf{x}$. Since in practice it is impossible to know those exact values that are missing, the best approach one can realistically expect is to learn the conditional distribution $P(\mathbf{X}_m | \mathbf{X}_{um} = \mathbf{x}_{um})$. The main challenge of this problem is that each $\mathbf{x}$ has its own set of masked variables and unmasked variables, resulting in possibly infinite combinations. Moreover, we would like to build a unified imputation model that handles the imputation of all datasets in the sequence. Moreover, since none of the datasets provides data vectors that covers the entire variable

set $\mathbf{S}$, the imputation model must be able to combine the knowledge learned from each individual dataset.

IV. EvoGAN: A Progressively Trained Imputation Model

In this section, we present the procedure of constructing EvoGAN, an imputation model progressively trained on a list of datasets arriving sequentially. Here we assume that dataset $\mathcal{D}_k$ arrives in round k, and shares some overlapping variables with the previous dataset $\mathcal{D}_{k-1}$, but each set may have other variables that are unique to it. We moved the description of EvoGAN$_1$ to the supplementary material, since it simply requires adopting an existing imputation GAN network. Here we focus on how the EvoGAN model at round k evolves to adapt to the the next dataset $\mathcal{D}_{k+1}$ at round $k+1$.

Here EvoGAN is being called a model that imputes progressively, but EvoGAN$_k$ is also being called a model that imputes on one part of the dataset. Perhaps, EvoGAN$_k$ can be defined as the version of the model in round k.

A. Constructing EvoGAN$_{k+1}$ from EvoGAN$_k$

Due to the existence of overlapping variables between $\mathcal{D}_k$ and $\mathcal{D}_{k+1}$, EvoGAN$_{k+1}$ is constructed upon EvoGAN$_k$. This approach ensures that the model can transfer the knowledge learned by EvoGAN$_k$ on those overlapping variables to EvoGAN$_{k+1}$. The challenges that we consider are two-fold:

1) How to make the model in round k preserve features learned in round $k-1$, while having enough flexibility to fit the new dataset?
2) How to modify its input / output structure to handle the newly included variables and removed variables?

To the best of our knowledge, EvoGAN is the first machine learning procedure that tackles the challenges mentioned above.

Next, we will describe how EvoGAN is modified in each round in terms of generator and discriminator, weight freezing schemes, model compression, and objective function. The overall architecture is illustrated in Figure 1

Expanded Generator: We build the generator G_{k+1} of EvoGAN$_{k+1}$ by first recreating the neural network structure of generator G_k from EvoGAN$_k$. All weights and biases are preserved. Then we expand each hidden layer by adding a reasonable amount of nodes that are fully connected to the previous layer. At this stage, if all the newly-added weights are initialized to zero, then G_k and G_{k+1} represent the same function. Similar to practices in transfer learning, we will freeze all or some of the weights inherited from G_k during training. The reader may refer to the Experiments section for the performance of different freezing schemes.

Next, we modify its input layer and output layer to fit the new sets of inputs and outputs. Let S_{rm}, S_{sh}, S_{new} represent $S_k - S_{k+1}, S_k \cap S_{k+1}, S_{k+1} - S_k$, namely the set of removed variables, shared variables, and new variables respectively. Since the input of G_{k+1} is a vector $(\mathbf{x}, \mathbf{m})$ only containing information of S_{sh} and S_{new}, we append an i.i.d. random noise vector $\mathbf{z}_{aug} \in \mathbb{R}^{\|S_{rm}\|}$ to $\mathbf{x}$, and a "completely masked" vector

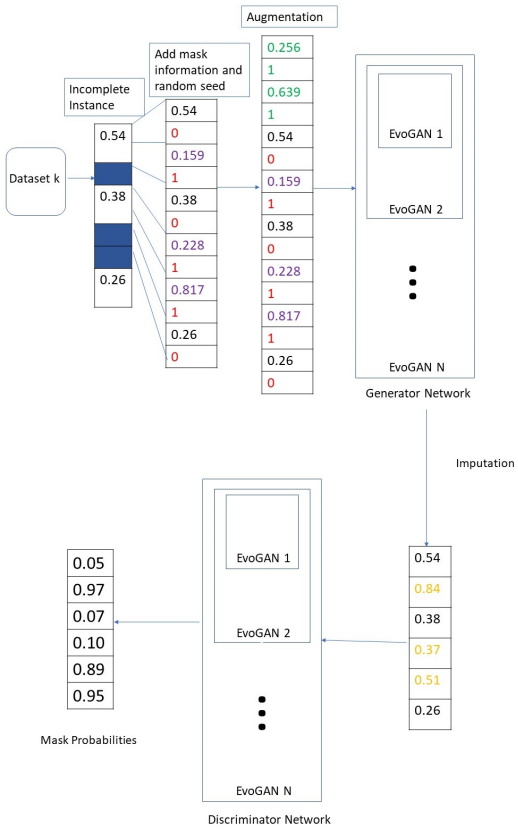

Fig. 1: The overall structure for EvoGAN for sequential data imputation. The input vector of the generator is augmented to contain random seed for the input nodes corresponding to features removed from the previous dataset.

$\mathbf{m}_{aug} = (1, ..., 1)$ of length $\|S_{rm}\|$ to $\mathbf{m}$. Now the augmented input vector $(\mathbf{z}_{aug}, \mathbf{x}, \mathbf{m}_{aug}, \mathbf{m})$ has length $2(\|S_{rm}\|+\|S_{sh}\|+\|S_{new}\|) = 2\|S_k\| + 2\|S_{new}\|$. We then add $2\|S_{new}\|$ nodes to the input layer of G_{k+1}. Similarly, we remove the nodes corresponding to the removed variables in the output layer, and append $\|S_{new}\|$ new nodes to it, so that the output vector is a vector of length $\|S_{sh}\| + \|S_{new}\| = \|S_{k+1}\|$.

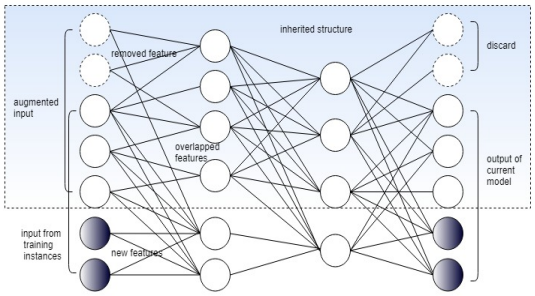

Fig. 2: Illustration of the network structure for the generator/discriminator network of EvoGAN$_k$ for $k \geq 2$.

Expanded Discriminator: Similar to G_{k+1}, the discrimi-

nator D_{k+1} is also generated by expanding the structure of D_k with additional nodes to the input layer, hidden layers, and the output layer. Note that the reason for adding new nodes to hidden layers is to grant additional learning potential since most of the inherited weights won't be updated during training. The input and output layer should be modified according to the change of variables in order to maintain the correspondence between nodes and variables. The construction of the generator/discriminator network is illustrated in Figure 2.

Weight Freezing: Weight freezing and pruning redundant weights have been shown to be effective methods of model compression. These are valuable methods of reducing network complexity and overfitting without affecting the performance of the network [34], [35]. We propose to "freeze" the inherited weights to preserve the knowledge learned from the previous round, so that those weights are not updated during the back-propagation of the current training round [36]. If all inherited weights are frozen, however, one may need to add a considerable amount of new nodes to add enough flexibility to adapt to the new dataset. Our experiments suggest that, in some instances, an acceptable trade-off is to freeze the lower layers and let the higher layers be completely trainable.

Model Compression: Deep neural networks contain a large number of parameters, making them prone to model overfitting with limited training data. This results not only in unnecessary computational costs but also limits the generalizability of the model which can be detrimental to sequential learning. This is an important motivation for the use of pruning techniques. Many methods have been proposed to address this issue of overfitting, such as pruning and soft weight-sharing [37]. In particular, pruning tend to significantly increase the sparsity of the model by setting many parameters to zero. This approach has a multiplicative effect on EvoGAN since a sparser network will not only reduce the model size in one round but will also improve the model sparsity of all subsequent rounds. We refer to the experiments section for evidence of this accumulated effect of pruning.

Objective Function: The objective of EvoGAN is straightforward: the generator generates values for the masked variables in a data vector, aiming at preventing the discriminator from correctly distinguishing between real values and fake values while the discriminator tries not to be fooled by the fake values filled by the generator. Although detailed settings may vary, the objective function for the generator should penalize the model depending on how much the generated values are different from the masked values, and the objective function for the discriminator should punish the model depending on the predictions on the mask vector.

V. Experiments on UCI Datasets

A. Benchmarks, Datasets, Evaluation Criteria

We test the following six variations of EvoGAN in our experiments.

- EvoGAN-Complete: This refers to the model under *complete freezing scheme* without $L1$ regularization.
- EvoGAN-Partial: This refers to the model under *partial freezing scheme* without $L1$ regularization.
- EvoGAN-Complete-L1: This refers to the model under *complete freezing scheme* with $L1$ regularization.
- EvoGAN-Partial-L1: This refers to the model under *partial freezing scheme* with $L1$ regularization.
- EvoGAN-Complete-L1+Prune: This refers to the model under *complete freezing scheme* with $L1$ regularization along with an additional pruning mechanism.
- EvoGAN-Partial-L1+Prune: This refers to the model under *partial freezing scheme* with $L1$ regularization along with pruning.

For our baseline, we consider the case when one of the state-of-the-art models for data imputation is trained from scratch on each dataset in the sequence. For this purpose, we chose the GAIN network [6] and the MisGAN network [38] as the baseline model. To compare with EvoGAN$_k$, on the k-th dataset D_k is used to train the baseline models, since they cannot make use of earlier datasets. We replicated the network structure used in the original papers with only necessary changes to adapt to the new datasets. Detailed implementation information can be found in the supplementary material.

We first validated different versions of EvoGAN on three datasets retrieved from UCI Machine Learning Repository [39]: *letter, credit, spam.*

To create a sequence of datasets, we picked each one of these three sets and subdivided them into subsets making sure that there was not only overlap between consecutive datasets in the sequence but that there also was addition and deletion of features from one set to another. In this manner, we subdivided each of the three original datasets to generate a sequence of smaller datasets.

For each of the datasets, we created n subsets with the following characteristics. Each subset has m_1 overlapping features with the previous subset and m_2 new features. If $m_1 = m_2$, then there is a 50% overlap between the previous subset and the new subset. For example, if Letter dataset is divided into $n = 4$ subset, then each subset will have 5000 rows and the columns from the original dataset that appear in each of the 4 subsets will be 1-6, 4-9, 7-12, and 10-15, respectively. Here $m_1 = m_2 = 3$.

Since we needed to start with a complete dataset to validate our methods, it was necessary for us to artificially determine the features included in each dataset. It must be emphasized that we did not choose them based on their significance or importance, but simply by position. Our initial tests were on sequences of four datasets, but we subsequently validated our model on longer sequences.

To evaluate the performance of our models, we use RMSE as a performance metric. In addition, we also use compression rate (CR) and speed-up to measure the quality of compression and acceleration respectively:

$$CR = \frac{\text{number of parameters in baseline model}}{\text{number of parameters in EvoGAN model}} \quad (1)$$

Model	Letters			Spam			Credit card		
	Speed-up	**CR**	**RMSE**	**Speed-up**	**CR**	**RMSE**	**Speed-up**	**CR**	**RMSE**
EvoGAN-Complete	1.412	3.27	0.163	1.278	3.77	0.072	1.201	3.48	0.153
EvoGAN-Partial	1.068	4.92	0.164	0.97	4.25	0.069	1.045	4.59	0.130
EvoGAN-Complete-L1	1.568	6.37	0.162	1.405	7.49	0.069	1.271	6.86	0.142
EvoGAN-Partial-L1	1.077	5.72	0.164	0.963	4.88	0.063	1.101	5.34	0.127
EvoGAN-Complete-L1+Prune	1.465	21.24	0.148	1.523	19.96	0.068	1.315	23.17	0.127
EvoGAN-Partial-L1+Prune	1.214	11.26	0.152	1.247	13.22	0.068	1.207	11.97	0.129
Baseline	1.000	1.00	0.156	1.000	1.00	0.075	1.000	1.00	0.135

TABLE I: Overall performance of EvoGAN with respect to baseline model on Letters, Spam and Credit Card datasets.

$$\text{Speed-up} = \frac{\text{running time of baseline model}}{\text{running time of EvoGAN model}} \tag{2}$$

We use the *masked* values from the data records to evaluate the performance of the models. These records are split evenly between the validation set and test set. The validation RMSE are used for model selection, and the test RMSE are reported in the tables and figures of this paper.

It is worth noting that our focus was not on fine-tuning the hyper-parameters to improve the performance; we simply chose some intuitive values and empirical results to pick the best settings.

B. Network Construction

For EvoGAN$_1$, we use the GAIN architecture for both the generator and discriminator. It has three fully connected layers; the first two use ReLU as activation functions and the last layer uses sigmoid activation function. L1 regularization was added to the loss function. For the generator, the input is the concatenation of mask, actual data, and random seed; for the discriminator, the input is the concatenation of the hint matrix and the imputed data. In the subsequent rounds, EvoGAN freezes most or all of the weights and biases from the previous round and introduces N new nodes at each stage to deal with the new dataset with some previously seen features. This process is the same for both the generator and the discriminator of EvoGAN. The newly added parameters are initialized to zero. This is repeated over many rounds.

C. Weight freezing, Regularization, and Pruning

Under the *complete freezing scheme*, all the weights post-training for the version of the model in the previous round are preserved and inherited by the model in the current round. Then, N extra nodes are added to each layer and only the weights corresponding to these new nodes are retrained and updated.

Under the *Partial freezing scheme*, all the weights post-training for the version of the model in the previous round are preserved and inherited by the model in the current round. Then, N extra nodes are added to each layer, but in this case, all the weights corresponding to the last layer are retrained and updated.

In addition to these weight freezing mechanisms, we consider L1-regularization and a pruning mechanism in which all edges corresponding to the smallest m % of weights in terms of magnitude are removed. This process is repeated in every round to ensure a lean model. Pruning methods are well-known for providing benefits of model compression while avoiding the problem of over-fitting.

VI. Results for GAIN-Based EvoGAN

A. Overall Performance

Our results for the performance of models in initial tests are outlined in Table I. For our initial tests, we randomly removed 20% of all data points. It is clear that models constructed according to the **EvoGAN-Complete-L1+Prune** scheme consistently improve upon the RMSE of baseline while also achieving impressive rate of compression as well as speed-up. In addition, models constructed according to the **EvoGAN-Partial-L1+Prune** scheme also achieve improvements in RMSE as well as substantial compression rates and speed-up.

We ran additional tests with longer sequences of datasets, various missing rates, and datasets with less overlap. Our models continued to outperform the baseline in all these cases. These results are discussed in more detail in the following subsections.

B. Effects of Regularization and Pruning

We ran our models under both weight freezing schemes with and without L_1 regularization and pruning. The results are detailed in Table I.

The models under the *complete freezing scheme* see a substantial difference with the addition of L_1 regularization. They consistently achieve higher compression rates than their counterparts without regularization. In addition, these models also see some improvement in RMSE as well as some acceleration in running time. With the addition of pruning, these models achieve further improvements in RMSE and dramatic changes in compression rate.

The models under the *partial freezing scheme* also see improvements in RMSE, compression rates, and speed-up with the addition of L_1 regularization and pruning. Most of these gains appear as improved compression rates and lower running-times.

We varied the values of the hyperparameters for L_1 regularization and the degree of pruning to experimentally pick the most suitable values for our models for each sequence of datasets.

C. Longer Sequences of Datasets

We divided the datasets into longer sequences of subsets and found that while various models of EvoGAN continue to achieve reasonable improvements in speed-up and

Model	Letters			Spam			Credit card		
	Speed-up	**CR**	**RMSE**	**Speed-up**	**CR**	**RMSE**	**Speed-up**	**CR**	**RMSE**
EvoGAN-Complete-L1	1.28	4.25	0.174	1.405	3.39	0.068	1.271	4.28	0.071
EvoGAN-Partial-L1	1.05	3.11	0.171	0.891	3.64	0.056	0.920	3.25	**0.059**
EvoGAN-Complete-L1+Prune	**1.568**	**14.0**	**0.145**	**1.781**	**19.04**	**0.055**	**1.762**	**13.08**	0.064
EvoGAN-Partial-L1+Prune	1.215	8.45	0.151	1.623	11.27	0.058	1.244	8.29	0.064
Baseline	1.000	1.00	0.178	1.000	1.00	0.076	1.000	1.00	0.069

TABLE II: Overall performance of EvoGAN with respect to baseline model on Letters, Spam and Credit Card datasets that have less overlap.

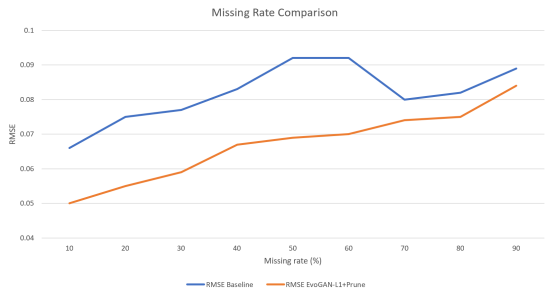

Fig. 3: RMSE for EvoGAN-complete-L1+prune and baseline model for datasets with different missing rates.

very high compression rates when compared to the baseline model, **EvoGAN-Complete-L1+Prune** and **EvoGAN-Partial-L1** models outperformed the baseline in every respect. We found that for longer sequences, as long as there was at least 50% overlap between consecutive datasets, the models still performed well.

For instance, when we divided the Letters dataset into a sequence of 6 smaller datasets with 50% overlap, we found that both **EvoGAN-Complete-L1+Prune** and **EvoGAN-Partial-L1+Prune** showed substantial reductions on the RMSE of the baseline while maintaining better running times and achieving significant compression rates. In this case, the best performer was **EvoGAN-Complete-L1+Prune** with a reduction in RMSE of over 18% while achieving a remarkable compression rate of almost 24. The results are detailed in Table III.

Model	Letters		
	Speed-up	**CR**	**RMSE**
EvoGAN-Complete-L1	1.39	9.56	0.176
EvoGAN-Partial-L1	1.06	9.93	0.174
EvoGAN-Complete-L1+Prune	**1.42**	**23.83**	**0.141**
EvoGAN-Partial-L1+Prune	1.36	11.72	0.143
Baseline	1.00	1.00	0.173

TABLE III: Overall performance of EvoGAN on sequence of 6 datasets.

Overall, we found that for longer sequences our models continued to show significant advantages in model compression and speed-up while maintaining low prediction error.

D. Missingness

Our initial results used a missing rate of 20%. We carried out more extensive experiments to examine the affect of missing rate and found that EvoGAN consistently outperformed the baseline. For instance, Figure 3 illustrates the effect of the missing rate on the performance of the best performing EvoGAN model on SPAM dataset. Our experiments showed that **EvoGAN-Complete-L1+Prune** performed better than the baseline and the other models in terms of RMSE while preserving the improvements in compression rate and speed-up. As is evident from the figure, this model is consistently better than the baseline for values of missing rate ranging from 20% to 90%.

E. Sequences of datasets with less overlap

We varied the degree of overlap of features between two consecutive datasets and found interesting results. Table II details the results for the case when the overlap of features between two consecutive datasets is around 50%. We find that **EvoGAN-Complete-L1+Prune** and **EvoGAN-Partial-L1+Prune** models consistently outperformed the baseline in every metric. Overall, it is clear that these EvoGAN models are successful at exploiting the knowledge gathered over multiple rounds while also being more efficient than the baseline models.

VII. Results for MisGAN-Based EvoGAN

We also adapted the EvoGAN technique to MisGAN, so that EvoGAN_1 is now replaced with MisGAN structure introduced in [38]. We conducted tests on the MNIST dataset. We found that our EvoGAN provided image imputation comparable to the original EvoGAN while requiring much less parameters.

A. Experiment Design

We create two subsets from the original MNIST dataset. The first subset contains 6000 images, where each image is cropped so that only the upper-left 20×20 block is kept. The second subset contains the lower-right 20×20 of another 6000 images. EvoGAN_1 is trained only on the first dataset, while EvoGAN_2 is trained only on the second subset.

B. Network Structure

For EvoGAN_1, we use the basic MisGAN architecture for the generators, discriminators and imputer. Essentially, MisGAN has 3 network structures named ConvGenerator , ConvCritic and Imputer. ConvGenerator generates fake data and fake mask. ConvCritic is used for all three discriminators (imputer discriminator, mask discriminator and data discriminator). Imputer network contains generator for the imputed data. In EvoGAN_2 we used the same layers structure for all the generators, discriminators and imputer network, which consists

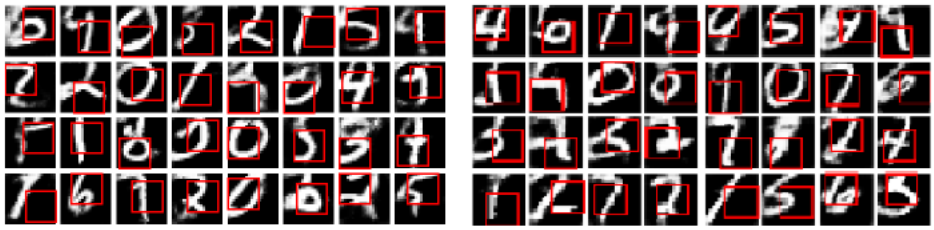

Fig. 4: Image Imputation Results for an EvoGAN model

Fig. 5: EvoGAN performance in real-world situations. Top row: Walmart sales data; Bottom row: S&P 500 data.

of three convolutional layers and a fully-connected layer. First 3 layers use ReLU as activation functions. For the last layer of the Imputer network, sigmoid activation function was used. In the subsequent rounds, EvoGAN freezes all of the weights and biases from the previous round and introduces 10% new nodes at each stage to adapt to the the new dataset with some previously seen features.

C. Results

Figure 4 displays a few randomly selected imputation results. The pixels outside each red box are considered missing from the original image, and the imputation model recreated this area based on their knowledge of the dataset. The imputation results on the second subset was shown for EvoGAN$_2$ and a MisGAN model trained from scratch. As showned in the figure, EvoGAN$_2$ was able to achieve comparable imputation performance on these masked images by largely re-using EvoGAN$_1$.

VIII. Simulating Real-World Situations

We simulated two real-world scenarios: sales revenue imputation and stock price imputation.

A. Imputation Results on Walmart Sales Data

We evaluated various EvoGAN models on each subset of Walmart data and compared them with baseline. We observed all the EvoGAN models performed well in terms of speed-up and compression rate while all but one of the models achieved lower RMSE than the baseline. Here we assumed 50% overlap between each subsets. Both **EvoGAN-Partial-L1** and **EvoGAN-Partial-L1+Prune** models achieve lower RMSE than the baseline. The**EvoGAN-Complete-L1+Prune** model is the best performer in terms of speed-up and compression rate. Figure 5 provides a comparison of the performances of the **EvoGAN-Partial-L1** model and baseline in terms of RMSE, speed-up and compression rate. As shown in the figure, EvoGAN consistently outperforms the baseline model in terms of RMSE, training time, and the number of parameters, with Subset 4 being the only instance in which the RMSE of the baseline is marginally lower than that of our model.

B. Imputation Results on S&P 500 Data

Figure 5 displays the comparison between EvoGAN and the baseline approach for each subset of the sequence. While

we achieved good results with different variants exhibiting different strengths, we focused on **EvoGAN-Partial-L1** as it was the best performer in terms of all three metrics. Overall, EvoGAN achieves faster training time, with far fewer parameters while maintaining good imputation accuracy when compared to the baseline.

IX. Conclusion

In this paper, we have proposed a novel technique called EvoGAN that is useful for data imputation on multiple datasets that arrive sequentially. Our procedure leads to adaptable models that use transfer learning to take advantage of the overlap between datasets and evolve over many rounds to maintain the quality of imputation while preserving substantially lower network complexity when compared to various baseline models.

References

[1] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information sciences*, vol. 275, pp. 314–347, 2014.

[2] J. T. Massey, T. F. Moore, W. Tadros, and V. Parsons, "Design and estimation for the national health interview survey 1985-94." *VITAL AND HEALTH STATISTICS. SERIES 2: DATA EVALUATION AND METHODS RESEARCH*, vol. 110, pp. 1–33, 1989.

[3] CDC, "Behavioral risk factor surveillance system operational and user's guide," *Atlanta, GA: US Department of Health and Human Services, Centers for Disease Control and Prevention*, 2006.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] D. Lee, J. Kim, W.-J. Moon, and J. C. Ye, "Collagan: Collaborative gan for missing image data imputation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2487–2496.

[6] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*, 2018, pp. 5675–5684.

[7] R. D. Camino, C. Hammerschmidt *et al.*, "Generating multi-categorical samples with generative adversarial networks," *ICML Workshp*, 2018.

[8] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1596–1607.

[9] L. Duan, D. Xu, and I. Tsang, "Learning with augmented features for heterogeneous domain adaptation," *arXiv preprint arXiv:1206.4660*, 2012.

[10] C. B. Do and A. Y. Ng, "Transfer learning for text classification," in *Advances in Neural Information Processing Systems*, 2006, pp. 299–306.

[11] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[12] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2015, pp. 1225–1237.

[13] P. P. Reddy and M. M. Veloso, "Negotiated learning for smart grid agents: entity selection based on dynamic partially observable features," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[14] A. Gelman, G. King, and C. Liu, "Not asked and not answered: Multiple imputation for multiple surveys," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 846–857, 1998.

[15] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[16] R. J. Little, "Regression with missing x's: a review," *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 1227–1237, 1992.

[17] P. T. Von Hippel, "4. regression with missing ys: An improved strategy for analyzing multiply imputed data," *Sociological Methodology*, vol. 37, no. 1, pp. 83–117, 2007.

[18] I. A. Gheyas and L. S. Smith, "A neural network-based framework for the reconstruction of incomplete data sets," *Neurocomputing*, vol. 73, no. 16-18, pp. 3039–3065, 2010.

[19] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[20] W. Ren, X. Lian, and K. Ghazinour, "Efficient join processing over incomplete data streams," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 209–218.

[21] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[22] J. Barnard and X.-L. Meng, "Applications of multiple imputation in medical studies: from aids to nhanes," *Statistical methods in medical research*, vol. 8, no. 1, pp. 17–36, 1999.

[23] A. Mackinnon, "The use and reporting of multiple imputation in medical research–a review," *Journal of internal medicine*, vol. 268, no. 6, pp. 586–593, 2010.

[24] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.

[25] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: better text generation via filling in the_," *arXiv preprint arXiv:1801.07736*, 2018.

[26] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank svd via fast alternating least squares," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3367–3402, 2015.

[27] H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, "A tensor-based method for missing traffic data completion," *Transportation Research Part C: Emerging Technologies*, vol. 28, pp. 15–27, 2013.

[28] J. Yoon, J. Lee, E. Yang, and S. J. Hwang, "Lifelong learning with dynamically expandable network," in *International Conference on Learning Representations*. International Conference on Learning Representations, 2018.

[29] T. Ash, "Dynamic node creation in backpropagation networks," *Connection science*, vol. 1, no. 4, pp. 365–375, 1989.

[30] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 921–934, 2019.

[31] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[32] J. Gupta, S. Paul, and A. Ghosh, "A novel transfer learning-based missing value imputation on discipline diverse real test datasets—a comparative study with different machine learning algorithms," in *Emerging Technologies in Data Mining and Information Security*. Springer, 2019, pp. 815–826.

[33] D. Chen, S. Yang, and F. Zhou, "Transfer learning based fault diagnosis with missing data due to multi-rate sampling," *Sensors*, vol. 19, no. 8, p. 1826, 2019.

[34] S. Hosseini and C. Jutten, "Weight freezing in constructive neural networks: a novel approach," in *International Work-Conference on Artificial Neural Networks*. Springer, 1999, pp. 11–20.

[35] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[36] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International Conference on Artificial Neural Networks*. Springer, 2018, pp. 270–279.

[37] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural computation*, vol. 4, no. 4, pp. 473–493, 1992.

[38] S. C.-X. Li, B. Jiang, and B. Marlin, "Misgan: Learning from incomplete data with generative adversarial networks," *arXiv preprint arXiv:1902.09599*, 2019.

[39] M. Lichman *et al.*, "Uci machine learning repository," 2013.