LCT-GAN
Improving the efficiency of tabular data synthesis via latent embeddings

Viktor Velev

Supervisor(s): Zilong Zhao, Lydia Chen
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

Abstract

In the past decade data-driven approaches have been at the core of many business and research models. In critical domains such as healthcare and banking, data privacy issues are very stringent. Synthetic tabular data is an emerging solution to privacy guarantee concerns. Generative Adversarial Networks (GANs) are one of the emerging solutions for synthesizing data. However in order to capture all relevant relationships between columns, tabular data needs to be numerically encoded. As columns might be of different types, this is a challenging task as proven by recent approaches [24]. Throughout this paper, we focus on the dimensionality explosion problem, which leads to high-dimensional datasets alongside computational overhead and increase in training time. We introduce a novel synthesis pipeline - LCT-GAN - an improvement to the current state-of-the-art in tabular data synthesis CTAB-GAN [24]. Our approach addresses the dimensionality explosion problem by introducing a low-dimensional embedding step via an autoencoder prior to training. It is then combined with a novel conditional GAN architecture, operating in latent space. After thorough evaluation, we observe that our solution achieves more than 30% improvement in certain statistical metrics in comparsion to CTAB-GAN, accompanied by 5 fold decrease in size and 150 times speedup in training time for a single epoch. We successfully show that it is possible to embed data using autoencoders, and that GANs are able to learn complex relationships in latent space in the context of tabular data.

1 Introduction

Data-driven approaches have been at the core of many business and research models for a while now. Grocery stores optimise their stock based on data gathered, tied to unique customers through discount cards. Banks assess risks, trust and creditworthiness of loan applicants by means of analysis on big data. Hospitals gather numerous data points for each patient - radiological images, insurance claims and medical records - which are later on used to improve their processes, and simplify management [6]. Solving complex problems in various fields, happens through data-hungry methods such as machine learning. In critical domains such as healthcare and banking, data privacy issues are very stringent, as large amounts of sensitive data is involved, and stored in a distributed manner [12].

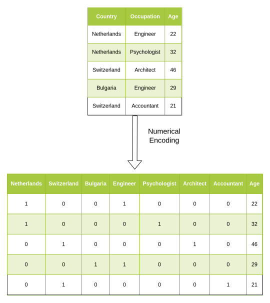
While data sharing is crucial for knowledge development, privacy concerns and strict regulation (e.g., European General Data Protection Regulation (GDPR)) unfortunately limit its full effectiveness. Thus, to effectively utilize specific methods, privacy concerns and regulatory constraints need to be addressed. The most common method used in practice is the anonymization of identifiable data, however most similar approaches are susceptible to de-anonymization attacks [2]. As a result of that, synthetic tabular data emerges as an alternative to enable data sharing while fulfilling regulatory and privacy constraints [8].

Synthetic tabular data is an emerging solution to privacy guarantee concerns [16]. Synthetic data statistically resembles real data and can comply with a multitude of strict regulations due to its synthetic nature. Most commonly datasets are organized in tables and populated with both continuous and categorical variables, or a mix of the two. Generative Adversarial Networks (GANs) are one of the emerging solutions to learning a data distribution. GANs are first trained on a real dataset, trying to learn it's multi-dimensional probability density function, subsequently sampling from it, thus generating new samples. The sampled data should be resembling statistically the original dataset. However, when GANs are used in the context of tabular data, a multitude of additional challenges come up.

1.1 Challenges of Tabular Data

Tabular data consists of many different types of variables: numerical, textual, categorical are at the core of every data table. In order to build machine learning algorithms around tabular data, it needs to be encoded to numerical space, whilst capturing all important features and not introducing unwanted relationships. There exist various encoding schemes, they range from straight-forward, to very challenging, as shown in [24] [16] [3].

Categorical columns are amongst the biggest challenges to encode. Simplistic approaches such as label encoding - assigning a number for each category, resulting in a single column - introduces unwanted relationships between categories. Another embedding solution might be using word2vec [15], however that would introduce a layer of complexity, incapable of handling different languages and out-of-vocabulary words (OOV). A workaround is to use one-hot-encoding. This way it captures the categorical nature, without any unwanted relationships. However it requires to create new columns corresponding to the number of all possible categories in the column. Potentially this might result in a table with exponentially increased number of columns - dimensionality explosion. As a consequence this leads to increased use of computational resource when training.



Country	Occupation	Age
Netherlands	Engineer	22
Netherlands	Psychologist	32
Switzerland	Architect	46
Bulgaria	Engineer	29
Switzerland	Accountant	21

Netherlands	Switzerland	Bulgaria	Engineer	Psychologist	Architect	Accountant	Age
1	0	0	1	0	0	0	22
1	0	0	0	1	0	0	32
0	1	0	0	0	1	0	46
0	0	1	1	0	0	0	29
0	1	0	0	0	0	1	21

Figure 1: Visualized dimensionality explosion of using one-hot encoding for categorical variables.

High-dimensional data introduces a lot of training overhead. In the case of tabular synthesis, there are categorical columns with hundreds or thousands of possible values, which results in immense amount of dimensions for each table, due to one-hot-encoding. Embeddings are techniques used to reduce dimensionality. Multiple embedding techniques are explored further in Section 2.

1.2 Research Questions and Contributions

In this paper, we are building upon the current state-of-the-art tabular data synthesis solution CTAB-GAN+, looking for an answer to the following research question: Are latent embeddings a viable solution to increase efficiency in tabular data synthesis?

We aim to address the dimensionality explosion problem and increase effectiveness of handling categorical variables and high-dimensional data. To such end, we explore a novel method of compressing data to improve the performance of generative tabular data synthesis solutions. We propose to use autoencoders as an embedding solution - reducing the dimensionality of tabular data. Specifically, we design a novel data synthesis pipeline, termed Latent Conditional Tabular GAN (LCT-GAN), including an adversarial network architecture and an autoencoder as preprocessing step.

The evaluation process consists of encoding multiple datasets, consequently comparing the original dataset with their decoded counterparts, using a multitude of metrics, such as Jensen-Shannon divergence (JSD) [5], Wasserstein distance (WD) [13] and difference in pair-wise correlation (Diff. Corr.) as seen in [24]. Moreover, performance of the novel data synthesis pipeline is further evaluated via machine learning based analysis on the generated synthetic data, observing and comparing metrics to the same machine learning performing on the original dataset.

Our extensive evaluation shows significant improvement in training. In some cases with high-dimensional data we see LCT-GAN outperforming CTAB-GAN within limited amount of training time, improving up to 30% machine learning utility metrics and up to 23 % statistical similarity. We show that autoencoders are a viable embedding solution, by proving that they are able to learn and reconstruct tabular data. Alongside that we demonstrate that LCT-GAN is able to learn in latent space. As a whole LCT-GAN reveals great promise, as a first iteration latent embedding solution in the field of tabular data synthesis.

2 Related Work

Related work in the field can be split into two categories: **Tabular data synthesis solutions** and **Deep latent solutions**. The first category goes through different solutions tackling the data synthesis problem with or without embedding solutions, whereas the second category lays out solutions in different fields, motivated by similar approaches of latent encoding of data prior fitting machine learning models.

2.1 Tabular data synthesis solutions

Throughout this research are building upon the current state-of-the-art in tabular data synthesis - CTAB-GAN+ [25]. It is a novel conditional tabular GAN architecture that can effectively model diverse data types. However one of the problems, not addressed by CTAB-GAN, is the dimensionality explosion problem leading to sub-optimal use of computational resources.

Another tabular GAN solution is CT-GAN [23], which leverages the usage of conditional vector to oversample the minority class to address imbalanced tabular data generation, alongside wasserstein distance and gradient penalty for stable training. Several other studies extend GANs to provide proper support for generation of categorical variables - MedGAN [4], CrGAN-CNet [17], TableGAN [19]. MedGAN has tried to integrate an autoencoder into the training process to improve statistical similarity. CrGAN deals additionally with missing values. TableGAN being one of the contenders for the state-of-the-art in statistical similarity, it is outperformed by CTAB-GAN in terms of privacy guarantees.

Although aforementioned algorithms can generate tabular data including some accommodation for categorical variables, they do not represent any efforts into mitigating the dimensionality explosion problem.

2.2 Deep latent solutions

One of our core contributions in this paper is the evaluation of embedding solutions. Deep latent solutions incorporate in some way or form a latent embedding for the use of speeding up training, enabling algebraic manipulation or data reconstruction.

Fourier descriptor [14] is used in image processing as an embedding, transforming an image to it's sine and cosine components.

A very well known technique for dimensionality reduction is PCA [7]. It provides a simple way of finding the best vectors in space, using which you can represent the most amount of the real data. This comes with a great trade-off between dimensions reduced and information loss.

Word2vec [15] is widely-spread in the field of Natural Language Processing (NLP). It provides efficient word associations and word embeddings. Using word2vec in the context of tabular data would ultimately require training a different model for each categorical column, as each embedding needs specific context. In addition to that it does not handle out-of-vocabulary words, e.g. unseen classes, so for example if we get a new person's phone number / address / nationality, which has not been before in the dataset, word2vec is not going to work.

Using latent representation from a trained autoencoder is not seen often. However it has been seen in the field of molecular generation (A de novo [20]) and point-cloud generation (lGANs in [1]). Their novel approaches achieve state-of-the-art level in their corresponding fields.

This paper is a thorough introduction of using autoencoders as an embedding solution in the field of tabular data synthesis.

3 Background

Throughout this research are be building upon CTAB-GAN+ [25], a novel conditional table GAN architecture that can effectively model diverse data types, including a mix of continuous and categorical variables. CTAB-GAN+ also addresses data imbalance and long tail issues, i.e., certain variables have drastic frequency differences across large values. One of the problems which is not addressed by CTAB-GAN+ is the dimensionality explosion problem. When training tabular GAN, data dimension is a limitation. For algorithms such as CTGAN and CTAB-GAN+, they use variational gaussian mixture to encode continuous columns, one-hot encoding to encode categorical column. Thus, if there is high dimensional categorical column, the final encoded data will encounter the dimensionality explosion problem.

3.1 Conditional Generative Adversarial Networks (CGAN)

GANs are a popular method to generate synthetic data first applied with great success to images and later adapted to tabular data. GANs leverage an adversarial game between a generator trying to synthesize realistic data and a discriminator trying to discern synthetic from real samples.

The conditional GAN, or CGAN for short, is a type of GAN that allows for the conditional generation of data. Data generation can be conditional on multitude of classes and or other data. It is appended to the noise as the input of the GAN and to the training data of the discriminator, allowing the GAN to condition itself. An illustrative diagram can be found in Figure 4

On top that, using Wasserstein Distance (WD) and replacing weight clipping with a constraint on the gradient norm of the critic to enforce Lipschitz continuity, introduced in WGAN-GP [9] allows for stable training and improved performance.

3.2 Autoencoders

Autoencoders [11] are an unsupervised learning technique in which we leverage neural networks to learn the most efficient latent representation of input data. Their architecture has an imposed bottleneck which forces a compressed knowledge representation of the original input. If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task. However, if some sort of structure exists in the data (i.e. correlations between input features), this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck. A diagram image can be seen in Figure 3.

3.3 Metrics of interest

Jensen-Shannon divergence (JSD)

JSD [5] provides a measure to quantify the difference between the probability mass distributions of individual categorical variables belonging to the real and synthetic datasets, respectively. This metric is symmetric and bounded between 0 and 1, allowing for easy interpretation of results.

Wasserstein Distance

Wasserstein [13] distance is used to capture how well the distribution of variables are emulated by synthetically produced datasets. In a similar manner this is used for the stable training of GANs [9] alongside a gradient penalty.

Difference in pair-wise correlation

In order to evaluate how well feature interactions are preserved, we first compute the pair-wise Pearson correlation matrix for the columns within real and synthetic datasets individually, ranging between $[-1, +1]$. Similarly, the Theil uncertainty coefficient is used to measure the correlation between any two categorical features, ranging between $[0, 1]$. And the correlation ratio between categorical and continuous variables is used, which falls in the range of $[0, 1]$.

Finally, the difference between pair-wise correlation matrices for real and synthetic datasets is computed. This helps us to conclude our evaluation of statistical similarity.

4 LCT-GAN

LCT-GAN is a tabular data synthesis pipeline, designed to overcome the dimensionality explosion problem. As all machine learning methods perform computations on numbers, we need to transfer tabular data to proper numerical space. Building upon CTAB-GAN, we have leveraged their tabular encoding tools, perserving information about multi-mode continuous distributions, categorical and mixed variables. Categorical variables are handled via one-hot-encoding, which results in highly dimensional data. This is where LCT-GAN diverges from CTAB-GAN as an approach. It introduces an additional embedding step, with the goal to mitigate the dimensionality explosion problem. This embedding step, significantly influences the training process later, which prompted for the creation of a novel conditional GAN architecture, operating on latent space.

LCT-GAN performs two steps - (i) tabular encoding & embedding and (ii) generation. The encoding embedding step has the goal to embed all tabular training data to a numerical representation (tabular encoding), consequently to a lower-dimensional space (embedding). The generation step happens with all data embedded in the aforementioned lower-dimensional space. Vectors from the lower-dimensional (latent) space are called latent vectors. They are fed as training data and also as the target of generation. As the data is in lower-dimensional space, it takes less effort and computational resources to efficiently learn it, so we can sample from it. Later on the sampled latent vectors are decoded corresponding to the means of embedding, e.g. in this case - via the decoder of an autoencoder.

4.1 Overview

All core components of LCT-GAN are illustrated in Figure 2. LCT-GAN consists of a tabular autoencoder and a latent conditional GAN. The autoencoder serves as an embedding solution. It maps from original dataset space to lower-dimensional latent space, consequently mapping from latent space, back to original dataset space. The tabular autoencoder, learns the latent representation of a given dataset, effectively learning how to compress and decompress data. As latent space is lower-dimensional compared to dimensionally exploded data, this requires the introduction of a conditional GAN operating in latent space. This GAN architecture operates on latent data, which results in a lot of saved computational resources at some cost of statistical similarity performance, due to lossy embedding.

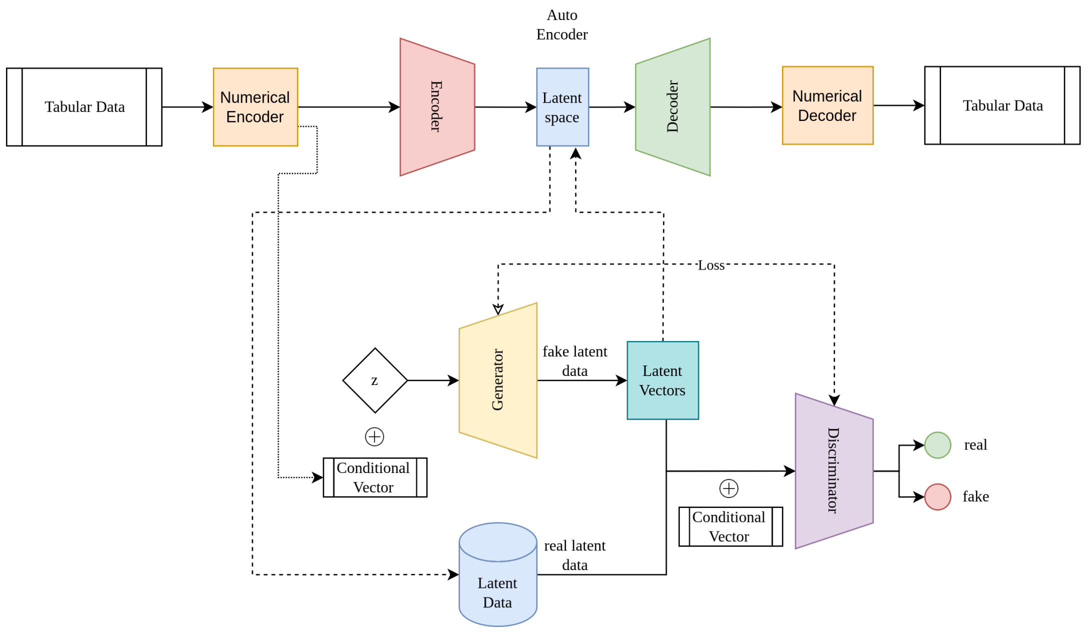


Figure 2: High-level architecture overview.

4.2 Encoding

Before we jump to generative models, we need an efficient way to encode and decode tabular data to numerical data. For this initial encoding, LCT-GAN uses the Mixed-type encoder introduced in CTAB-GAN+ [25].

Tabular data is encoded variable by variable. There are three types of variables continuous, mixed and categorical. Mixed variables are defined as mixed if they contain both continuous and categorical values, or continuous with missing values.

Continuous values are seen as a Gaussian mixture. However first one needs to estimate the number of modes in a gaussian mixture, which is why Variational Gaussian Mixture Model is used (VGMM) [18] to estimate the number of modes and fit a gaussian mixture. The resulting gaussian mixture can be expressed as:

$$M = \sum_{k=1}^{3} \omega_k \mathcal{N}(\mu_k, \sigma_k) \tag{1}$$

Where k is the number of modes, $\mathcal{N}$ is a normal distribution, ω_k, μ_k and σ_k are the weight, mean and standard deviation of each mode.

Categorical values are encoded via a one-hot encoded vector. Missing values are treated as a separate class, thus additional bit in the one-hot encoded vector is added. This is where most of the dimensionality in the training data is introduced, and what we are trying to mitigate, via an additional embedding.

There are additional implementation details expressed in CTAB-GAN+ [25] paper.

4.3 Embedding

Efficient two-way embeddings are a widely respected challenge. LCT-GAN makes use of an autoencoder (AE) to translate the data from a high-dimensional numerical space, to a very low-dimensional latent space.

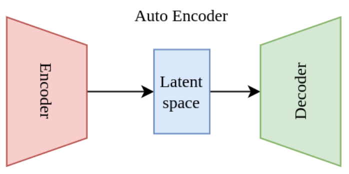


Figure 3: Architecture diagram of an autoencoder

The autoencoder consists of two distinct parts - encoder and decoder. The encoder's output is a low dimensional, latent vector. The goal for the decoder is to learn how to decode this latent vector. The resulting architecture is as shown in Figure 3. Encoder learns how to compress data, and the decoder learns how to decompress it.

The AE in LCT-GAN uses dense linear layers for both the encoder and decoder. The decoder is a mirrored architecture of the encoder. Mean-Squared Error (MSE) is used as a loss function, signalling the differences between the input and the output.

To evaluate whether we are correctly decoding, we compute Jensen-Shannon Divergence, Wasserstein Distance and Pair-wise correlation difference between the original and the decoded dataset. After we confirm that the autoencoder is fully trained, we use it's encoder to embed the high-dimensional numerical tabular data, to a latent vector of size 32 - rapidly decreasing the size of the data that is to be trained with.

4.4 Generation

Once we have the latent embeddings, we proceed to the generation phase. GANs as the leading architecture in generative applications, are at the core of LCT-GAN.

We are employing a conditional generative adversarial network. As we are working with latent data, we can afford to build the discriminator and generator as linear dense layers, instead of complicated convolutional and residual architectures. This further improves the training speed and use of computational resources.

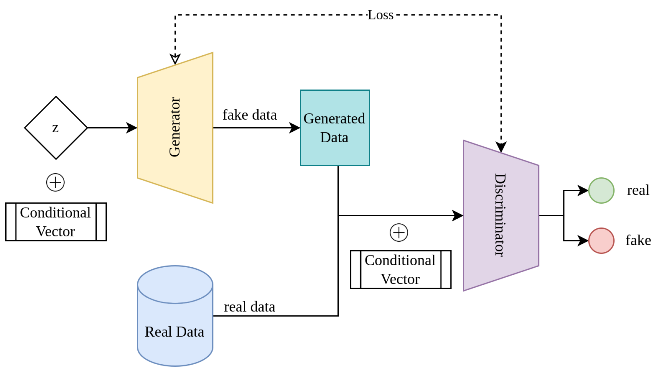


Figure 4: Architecture diagram of a Conditional GAN

Similarly to CTAB-GAN+ we are using conditional vectors to counter imbalanced training datasets. Whenever real data is sampled, we use conditional vectors to classify and rebalance the training data. The conditional vector is a bit vector given by the concatenation of all mode one-hot encodings for all types of variables. Each conditional vector specifies a single mode or category.

4.5 Architectures

In this subsection, we will outline the final Autoencoder and LCT-GAN neural network architectures, used in the computations of the results section. Most design choices are made, taking performance into account, which is why linear layers persevere as an approach over more complex ideas.

Autoencoder

Autoencoder is composed of simple linear dense layers, semantically split into two - an encoder and decoder. The decoder is symmetrical in terms of layer sizes. There are 6 densely connected layers, connected sequentially with sizes in the form of $m \times n$ where m is the dimension of the input and n the output of a given layer:

$$len(x) \times 128, 128 \times 64, \mathbf{64} \times \mathbf{64}, 64 \times 128, 128 \times len(x')$$

Where $len(x)$ is the length of the input, $len(x')$ is the length of the reconstructed input where $len(x) = len(x')$, as we want x and x' to be as close as possible to each-other. Between each layer there is non-linearity introduced in the form of LeakyReLU in order to avoid the vanishing gradient problem as much as possible [22].

Network size is static across input and output dimension, as all datasets used in the evaluation, were similar in size, the only exception is Loan, where the bottleneck (bolded) is changed to 32, to improve perfomance, as the training data size is 59

Latent Conditional GAN

Generative Adversarial Networks are generally composed of two networks. A Generator and a Discriminator. The goal of the generator is to try and learn the probability distribution of a dataset through the guidance of a Discriminator. The Discriminator is a classifier, trying to predict whether a given data sample is from the original data distribution or not, this loss is later propagated to the generator, guiding him to improve. The goal of the two networks is to reach Nash equilibrium.

We can condition GANs to generate a specific class of data, by introducing a conditional vector. The construction of this conditional vector is leveraged from the current state-of-the-art in tabular data synthesis CTAB-GAN. As seen in Figure 4, this conditional vector is fed once alongside the noise vector and another time when training the discriminator. It helps the GAN differentiate between different data features, and generate them on demand.

As we are operating in latent space, there is no need for complex neural network architectures, we should be able to learn the space through very simple dense layers.

The **Generator** consists of 4 layers structured in the following way, represented in the form of $m \times n$ where m is the dimension of the input and n the output of a given layer:

$$len(z) \times 16, 16 \times 32, 32 \times 64, 64 \times len(l)$$

where z is the noise vector input to the generator, and l is the desired generated latent vector.

The first three layers are combined with LeakyReLU with value of 0.1 and Batch Normalization with a value of 0.8. The final dense layer is succeeded by a tahn non-linearity. The tanh activation function results in higher values of gradient during training and higher updates in the weights of the network.

The **Discriminator** also consists of 4 linear dense layers structured the following way:

$$len(l) \times 64, 64 \times 32, 32 \times 16, 16 \times 1$$

Each layer is succeeded by a LeakyReLU non-linearity, with a value of 0.2. The final output layer has a Sigmoid activation layer, to ensure the outputs are in the probability range, as the discriminator is a classifier.

4.6 Training process

Training GANs is notoriously hard. Autoencoders are adding an additional layer of complexity on top of that, as latent space is specific to an autoencoder. Meaning that we cannot expect good performance out of an autoencoder and GAN algorithm, not trained in sync with each-other (e.g. training an autoencoder to decode latent space l_1, while GAN is trained on latent data from latent space l_2 from another autoencoder with different architecture and parameters). This made the exploration of hyperparameter tuning a challenging task, alongside the mode collapse problem, which was very apparent in some cases.

Latent data seemed to be within a random scale, which was hard for a GAN to learn, which is why we normalized the data from $[-1, +1]$, which increased variety and mitigated, to some extent, mode collapse.

Techincal limitations were imposed during the training process by hardware. An Intel i7-8550U mobile CPU was used to measure the training time of each algorithm and compare it, how it enhances the current state-of-the-art CTAB-GAN. Further improvements to the algorithm implementation such as proper GPU support, might further emphasise the illustrated differences in the 'Results' section.

LCT-GAN has implemented Wasserstein Distance and Gradient Penalty as introduced in WGAN-GP [9] to make training more stable.

5 Experiments and Evaluation

The evaluation of LCT-GAN is divided into two parts: evaluating performance of the embedding solution, evaluating generative performance combined with the use of computational resources (time spent training). As we are targeting efficiency, the core metric we are looking at is training time, associated with performance. We ran experiments, comparing, given a fixed amount of time, which algorithm performs better, outlining the gains in efficiency, achieved by LCT-GAN.

There are two types of performance assessment performed. Statistical similarity comparison, and a machine learning based analysis.

In order to ensure thorough evaluation, we test on 4 widely used machine learning datasets: Adult, Credit, Covertype and Load. All code, necessary to reproduce the outlined experiments can be found on https://github.com/VikVelev/LCT-GAN

5.1 Experimental Setup

Statistical similarity comparison of synthetic datasets through Jensen-Shannon Divergence and Wasserstein Distance is performed. These metrics are used to quantitatively measure the statistical similarity of real and synthetic data.

Machine Learning based analysis is used to contrast synthetic and real data by comparing the performance of the most widely used machine learning algorithms: Decision Trees, SVM, Random Forest, Logistic Regression and Multi-Layer Perceptron.

First off the data, set for comparsion, is split into training and test sets. Consequently, synthetic data is generated from the training dataset. Afterwards both the synthetic data and original are separately used to train the aforementioned machine learning algorithms, and afterwards both are evaluated on the original test set. Machine Learning performance is measured via accuracy, F1-score and AUROC.

5.2 Results

Autoencoder
is at the core of our generative pipeline - it lays the foundations for LCT-GAN, building the latent space which the GAN is trained on. The autoencoder needs to be able to encode a datum x and successfully decode it to x', with minimal difference between x and x'. In order to make sure our embedding solution works, we ran series of experiments, verifying the performance of the decoder by observing the statistical similarities between original datasets and the decoded dataset.

Dataset	Avg. WD	Avg. JSD	Diff Corr
Adult	0.01788	0.08821	0.32489
Credit	0.00966	0.00373	1.92776
Covertype	0.03044	0.00310	1.73784
Loan	0.03524	0.00283	0.21129

Table 1: Statistical similarity difference between real data and decoded data. Demonstrating the decoding performance of our embedding solution and the learned latent space. (500 epochs, batch size of 512)

In Table 1, we can see that, even with some difference, the autoencoder is able to learn how to encode and decode tabular data correctly, even with simple Dense / Linear layers as an architecture. This shows great promise in autoencoders as an embedding. For the sake of comparsion, the time it took to train these autoencoders is 30 minutes each. The performance of our autoencoder also servers as an upper bound of results, when it comes to generated data.

LCT-GAN efficiency
Table 5 illustrates how LCT-GAN compares in efficiency to CTAB-GAN. We have measured, given limited training time, all relevant metrics to statistical similarity, as a way to compare convergence and computational resources spent per unit of performance. We chose 15, 30 and 60 minutes as the time windows to simulate limited training resources (e.g. cloud workstations, which are billed by the minute or hour).

We can see an increase in efficiency when it comes in time per epoch is accompanied by a degraded performance in some cases. Some of that degradation might be due to lost information As we can see Jensen-Shannon Divergence and the rest of the statistical metrics are getting better after each time-period, showing that LCT-GAN is able to learn through multiple abstraction layers in latent space.

Looking at each dataset performance more closely we can see that for datasets with integer continuous columns or a lot of categorical columns, such as Adult and Covertype, LCT-GAN outeperforms CTAB-GAN for the given training time. For example training on the dataset Covertype we can see about **20%** improvement in machine learning accuracy for 15 minutes training, **25%** for 30 minutes and **33%** for 60 minutes. However datasets with a lot of real continuous columns such as Credit, seems to be a challenge for LCT-GAN to learn effectively and additional tweaking might prove to be useful.

All in all using latent space as an intermediary representation for tabular data is showing great promise. LCT-GAN as a novel approach in this context, is able to learn going through datasets about **150 times** faster, as they are in latent space, compressing them more than **5 times** (e.g. 331 vs. 64 dimensions per row) in size. When accounting for the training of autoencoder, LCT-GAN still performs faster, however needs more epochs to learn the relationships in latent space.

As a summary - we successfull show that LCT-GAN is able to learn complex patterns through a latent embedding, con-

Approach	Training Time	Dataset	Accuracy Diff	AUC	F1-Score	Avg. WD	Avg. JSD	Diff Corr	s/epoch
CTAB-GAN	15 minutes	Adult	9.12887	0.23000	0.25636	0.10141	0.08764	2.47028	400.12
		Covertype	36.83585	0.39111	0.43352	0.02652	0.03939	5.18157	606.00
		Credit	2.96318	0.40695	0.38449	0.00942	0.16099	2.10457	542.17
		Loan	8.70000	0.11605	0.22446	0.03647	0.06017	2.07039	12.01
	30 minutes	Adult	8.46350	0.08763	0.04773	0.06141	0.10041	1.67055	406.23
		Covertype	37.77141	0.39083	0.42852	0.02393	0.03989	4.93570	612.76
		Credit	5.83141	0.36330	0.42172	0.00731	0.08544	2.12346	539.23
		Loan	6.95000	0.08072	0.22642	0.03948	0.06461	1.61641	12.23
	60 minutes	Adult	4.82751	0.08410	0.06189	0.03371	0.12482	1.63593	401.16
		Covertype	37.44809	0.38852	0.42228	0.02506	0.04036	4.86093	608.14
		Credit	15.28137	0.47540	0.43807	0.00675	0.10345	2.19736	546.11
		Loan	4.32500	0.07297	0.12701	0.03776	0.03568	0.96509	13.45
LCT-GAN	15 minutes	Adult	**6.87378**	**0.17121**	0.30094	**0.03519**	0.33090	2.55401	**2.75**
		Covertype	**29.15499**	**0.24819**	**0.29968**	0.08067	0.05759	5.38522	**2.93**
		Credit	-	-	-	0.05234	0.02938	5.21766	**3.30**
		Loan	**8.10000**	0.20059	0.33139	0.07520	0.05652	2.45248	**0.26**
	30 minutes	Adult	**7.73364**	0.20239	0.29372	**0.03690**	0.26007	1.77521	**2.90**
		Covertype	**27.52151**	**0.23035**	**0.32369**	0.05780	0.04830	**4.81472**	**2.95**
		Credit	-	-	-	0.05238	0.02938	5.21228	**3.37**
		Loan	7.25000	0.19262	0.36607	0.07062	0.19332	2.28565	**0.24**
	60 minutes	Adult	8.16101	0.17960	0.27275	0.03753	0.25199	**1.46671**	**2.83**
		Covertype	**24.45717**	**0.20414**	**0.27737**	0.05125	0.04768	**3.77253**	**2.93**
		Credit	-	-	-	0.03963	0.02938	5.19638	**3.16**
		Loan	6.70000	0.12376	0.26787	0.07341	0.09637	2.13683	**0.25**

Figure 5: Efficiency comparison, combining all aforementioned metrics alongside with training time on an i7-8550U CPU. All metrics are as differences to the original datasets, which is why lower is better across all columns. Accuracy, AUC and F1-score are averaged across all 5 machine learning algorithms. Dashes (-) mean that the specified metric could not be computed due to lack of data variety. Bolded values represent metrics, where LCT-GAN performs better than CTAB-GAN.

verging faster than CTAB-GAN in the case of big datasets with a lot of categorical and integer columns. Even though LCT-GAN does not outperform CTAB-GAN in the long run, it performs well as a first-iteration approach in applying deep latent solutions for tabular synthesis. LCT-GAN shows great promise and should continue to be improved in various ways discussed in Section 7.

6 Responsible Research

Data privacy is sought after as an essential part of human integrity. This section will go over potential ethical issues, and highlight important topics such as reproducability and scientific integrity.

Implications of privacy invasive business practices can be seen across different industries. If personal data is shared across different institutions it becomes inherently hard to secure and impose strict regulations. The data becomes prone to breach in any institution and people are the end-victims paying for the inconsiderations of businesses.

In order to make sure this research does not have any negative impact on data privacy preservation, additional metrics might be added, to make sure, that generated data points are sufficiently distanced from real ones, preventing de-annonymization attacks. Differential Privacy is a topic that is further explored in CTAB-GAN+ [25].

In order to make progress, we need to build upon previous work. For that to be possible we need to produce reproducible research. Reproducability is crucial for the field of machine learning and data science. Results in most papers are very hard to reproduce due to the non-deterministic statistical nature of the algorithms and computational resources. The progress of this research has been recorded in a GitHub repo alongside pickled binaries, with trained models, helping the reproducability of the results shown above. All code can be found here: https://github.com/VikVelev/LCT-GAN

In order to preserve scientific integrity, all sources and references have been thoroughly checked. Good research practices go a long way and we advice everyone to do the best they can in encouraging writing, reading and reproducing scientific research in a proper and responsible manner.

7 Shortcomings and Future Work

Even without ground-breaking results, embedding solutions in the field of tabular data synthesis show good promise. There are a lot of ways the approaches discussed in LCT-GAN can be improved and this section will outline some possible extensions and improvements of LCT-GAN.

One of the main shortcomings of LCT-GAN is that, given enough computational resources and training time, CTAB-GAN outperforms it on pure statistical similarity and machine learning performance. However due to technical limitations and time constraints the differences, sufficient experiments are yet to be run to estimate the difference in quality.

There are not any obstacles in the pure mathematical sense, to build a close-to-perfect Autoencoder. Further experiments with the autoencoder architecture must be explored. For example a possible approach to discuss might be using convolutional layers instead of dense layers, alongside exploring solutions showing great promise in other fields such as residual layers [10] or randomly wired neural networks [21].

Given enough computational power, an exhaustive hyperparameter search is bound to improve the current results.

An additional way to improve LCT-GAN would be to delve into the neural network architectures behind the latent discriminator and generator. As of now all experiments are done with a single architecture, which might be hindering performance in certain bigger datasets. This could be mitigated by exploring different architectures and adapt the architectures based on the input size of the dataset.

Algebraic manipulation of data through it's latent space is an exciting field to explore as well. One could potentially build a powerful data imputation solutions based on interpolation or other algebraic operations in latent space of other latently similar (minimum distance between latent vectors) data.

8 Conclusion

High-dimensional data has proven to be a huge overhead when training machine learning algorithms. In this paper we took a closer look at using autoencoders for embedding high-dimensional tabular data. Subsequently using the embedded data as training data for a tabular synthesis pipeline called LCT-GAN. LCT-GAN is a complex architecture consisting of an autoencoder as an embedding step and a novel conditional GAN architecture, operating on latent (embedded) space.

Data synthesis solutions aim to generate data as close as possible to the original dataset. As a novel first-iteration approach, the ultimate goal of LCT-GAN is to show that latent embeddings are a viable technique. We successfully show that it is possible to embed data using autoencoders, and that GANs are able to learn complex relationships in latent space in the context of tabular data.

LCT-GAN was evaluated through a multitude of statistical similarity metrics, and machine learning based analysis. Consequently compared against the current state-of-the-art in tabular data synthesis CTAB-GAN. We observe that within less time and with less computational overhead than CTAB-GAN, in some cases we achieve more than **30% improvement** in relevant statistical metrics. This is accompanied by a **150x** speedup in training time for a single epoch in popular datasets and reducing the size of the data about **5 fold**.

Given enough computational resources, CTAB-GAN is able to learn more complex patterns and achieve better performance, however the application of latent embeddings in tabular data synthesis is yet to be explored fully. At the end, we discuss how LCT-GAN could be improved and point to multiple weaknesses which could be easily addressed as a first step to unravel the true potential of latent embeddings.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. 2017.

[2] Alexandros Bampoulidis and Mihai Lupu. An abstract view on the de-anonymization process. *CoRR*, abs/1902.09897, 2019.

[3] Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. A review of tabular data synthesis using gans on an ids dataset. *Information*, 12(9):375, Sep 2021.

[4] Edward Choi, Siddharth Biswal, Bradley A. Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. *CoRR*, abs/1703.06490, 2017.

[5] Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. Similarity-based methods for word sense disambiguation. *CoRR*, cmp-lg/9708010, 1997.

[6] Sabyasachi Dash, Sushil Kumar Shakyawar, Mohit Sharma, and Sandeep Kaushik. Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, 6(1):54, Jun 2019.

[7] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[8] Nils Gruschka, Vasileios Mavroeidis, Kamer Vishi, and Meiko Jensen. Privacy issues and data protection in big data: A case study analysis under gdpr. pages 5027–5033, 12 2018.

[9] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 3–10, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[12] M.K.Jayanthi Kannan and Harish Naik. Protecting confidential data over distributed storage in cloud. 02 2021.

[13] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1960.

[14] Iivari Kunttu, Leena Kunttu, Juhani Rauhamaa, and Ari Visa. Multiscale fourier descriptor for shape classification. pages 536 – 541, 10 2003.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[16] Alejandro Mottini, Alix Lheritier, and Rodrigo Acuna-Agost. Airline passenger name record generation using generative adversarial networks. *CoRR*, abs/1807.06657, 2018.

[17] Alejandro Mottini, Alix Lheritier, and Rodrigo Acuna-Agost. Airline passenger name record generation using generative adversarial networks. *CoRR*, abs/1807.06657, 2018.

[18] Nikolaos Nasios and Adrian Bors. Variational learning for gaussian mixtures. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, 39:849 – 862, 08 2006.

[19] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *CoRR*, abs/1806.03384, 2018.

[20] Oleksii Prykhodko, Simon Viet Johansson, Panagiotis-Christos Kotsias, Josep Arús-Pous, Esben Jannik Bjerrum, Ola Engkvist, and Hongming Chen. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):74, Dec 2019.

[21] Saining Xie, Alexander Kirillov, Ross B. Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. *CoRR*, abs/1904.01569, 2019.

[22] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[23] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. *CoRR*, abs/1907.00503, 2019.

[24] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. Ctab-gan: Effective table data synthesizing, 17–19 Nov 2021.

[25] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. Ctab-gan+: Enhancing tabular data synthesis, 2022.