Diffusion Models for Tabular Synthetic Data Generation

Sharmin Akhter
Supervisors: Dr. Alex Bihlo and Dr. Terrence Tricco
Master of Data Science
Memorial University of Newfoundland

August 30, 2023

Contents

1 Introduction

The growing significance of synthetic data within the realm of Machine Learning systems is undeniable. Obtaining real-world data can often pose challenges in terms of availability, time, and resource requirements. Moreover, concerns related to privacy and legal limitations might restrict access to certain types of real data. Synthetic data emerges as a crucial solution in such scenarios, offering a viable alternative that closely mimics real-world measurements.

In the context of this capstone project, our central focus revolved around Generative Models, with a specific emphasis on the diffusion model. The primary drive behind this endeavor was to gain practical expertise in utilizing TensorFlow for real-world applications. Concurrently, the project aimed to deepen comprehension of data manipulation techniques and the intricate nuances of diffusion models. At its core, this project aimed to apply these newly acquired skills to the creation of synthetic tabular data.

However, the impact of this project transcended personal technical growth. A substantial objective was to contribute to the broader realm of generative modeling by adapting the diffusion model to the distinctive requirements of tabular data. This report serves as a comprehensive documentation of the entire project journey. It delves into the evolution, challenges faced, methodologies embraced, and ultimate outcomes accomplished. Encompassing crucial elements such as an extensive literature review, meticulous methods employed, and the results attained, this report provides an in-depth exploration of the project's intricacies. Undertaking this venture was not just about refining technical prowess but also about making a meaningful contribution to the progressive domain of generative modeling, particularly in the context of tabular data adaptation.

The machine learning community is actively engaged in exploring generative models tailored for tabular problems[[1–3]]. This research avenue has gained traction due to the substantial demand for high-quality synthetic data to address various tabular tasks. Unlike domains like vision or NLP that benefit from abundant Internet data, tabular datasets often encounter limitations in size. Moreover, synthetic datasets provide the advantage of circumventing concerns related to GDPR, as they do not contain actual user data. This characteristic enables their public sharing without compromising user anonymity.

2 Literature Review

2.1 Generative AI Model

Generative AI refers to a subset of artificial intelligence focused on creating models that can generate new content, such as images, text, music, and more, resembling human-created data. Unlike traditional AI, which typically solves well-defined problems, generative AI deals with the creative aspect of generating novel and diverse outputs that capture the patterns and characteristics of the training data.

During the training period, generative AI models learn the underlying patterns in a given dataset. They then use this learned information to create new data that resembles the original examples. This process involves understanding the chances of different things happening in the data and using those chances to come up with new examples. How well this is done and how the new examples are made depend on how the model is built.

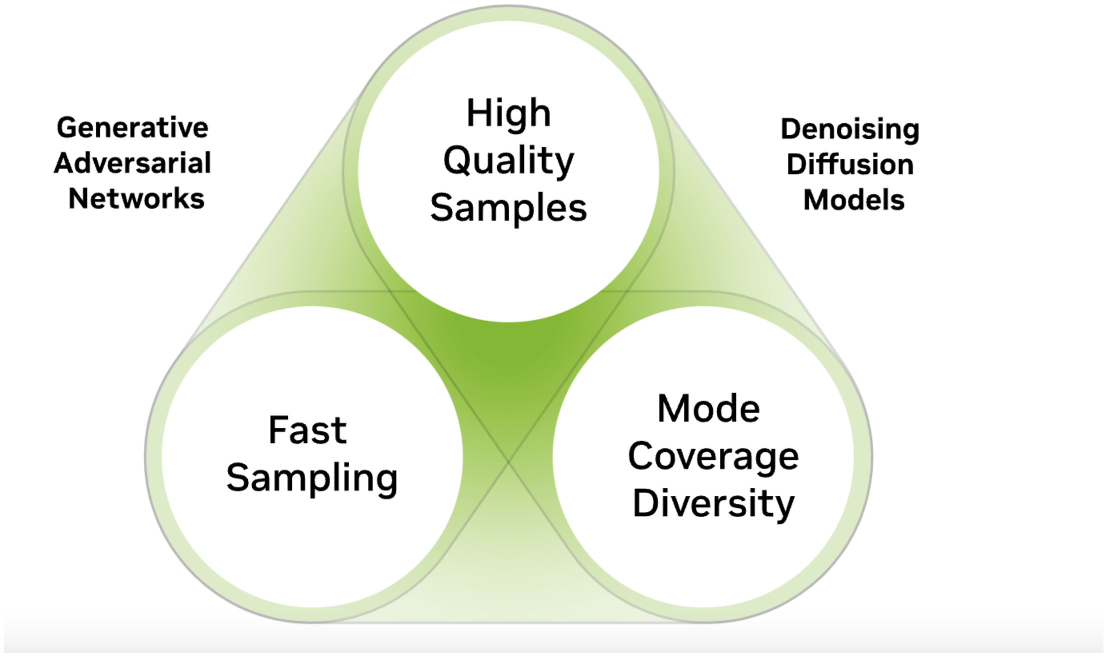


Figure 1: The three requirements of a successful generative AI model. Source

For instance, Generative Adversarial Networks (GANs) consist of a generator network that creates samples and a discriminator network that distinguishes between real and generated samples. Through iterative training, the generator improves its ability to produce samples that are increasingly difficult for the discriminator to distinguish from real data. This adversarial training process results in the generation of highly realistic data.

A proficient generative AI model built upon the diffusion model for synthetic data generation hinges on three essential components. First, it must excel in producing high-quality outputs, ensuring that user interactions are characterized by crisp speech and realistic images. Second, the model's ability to encapsulate diverse patterns, including those of minority groups, is crucial for reducing biases while maintaining output quality. Lastly, the speed at which the model operates is pivotal for its effective integration into interactive applications, such as real-time image editing within content creation processes.

2.2 Diffusion Models

Diffusion Models function as generative models, allowing them to produce data resembling their training dataset[[4–6]]. Essentially, these models operate by gradually introducing Gaussian noise to the training data, followed by learning to reverse this noise-induced transformation. Once trained, the Diffusion Model becomes capable of generating new data instances by applying its learned denoising process to randomly generated noise samples 2.

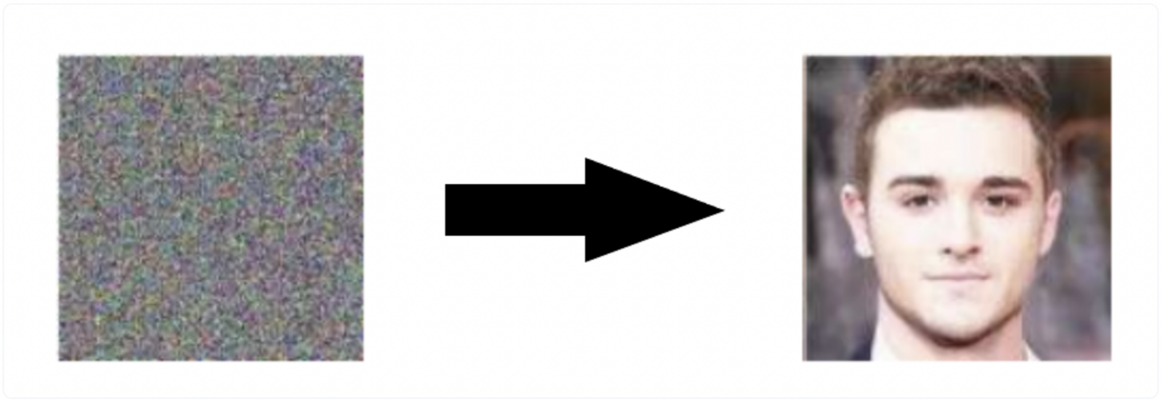


Figure 2: Diffusion Model to generate images from noise adopted from [4]

To elaborate, a Diffusion Model operates as a latent variable model utilizing a fixed Markov chain to project data onto a latent space. This chain incrementally introduces noise to the data, enabling an approximation of the posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, where $\mathbf{x}_1, ..., \mathbf{x}_T$ represent latent variables sharing the dimensionality of x_0. Illustrated in the accompanying figure 3 for image data, this Markov chain portrays the progressive noise addition until the image converges to Gaussian noise.

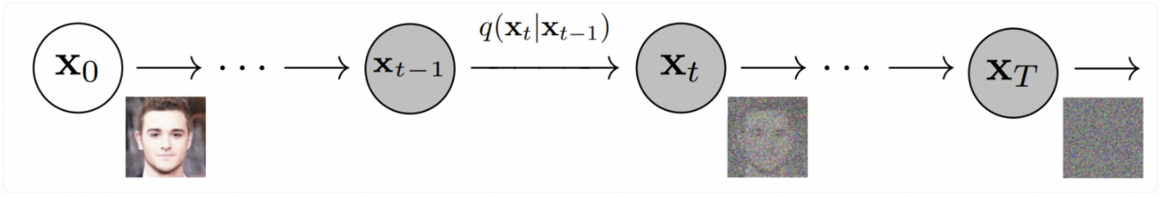


Figure 3: Markov Chain for Image Data adopted from [7]

The core training objective of a diffusion model lies in mastering the inverse process, denoted as $p_\theta(x_{t-1}|x_t)$. By retracing this chain in reverse, the model becomes capable of generating novel data instances shown in figure 4

In essence, a diffusion model operates through a dual process: initially introducing incremental Gaussian noise to a data point and subsequently acquiring the knowledge to revert the data to its original noise-free state. This acquired expertise empowers the model to create novel data instances by systematically applying the reverse process to randomly generated noise samples. Comprehensive insights and specifics about diffusion models can be found in the images documented by Ho et al. [7] and further elaborated upon in the work by Weng et al. [5].

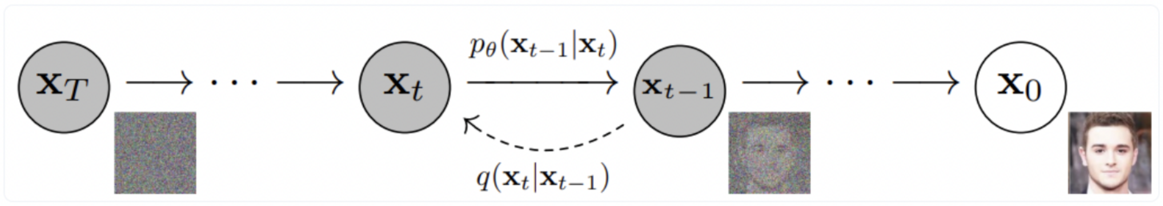

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 4: Reverse Process Image adopted from [7]

2.3 Diffusion Model from Image to Tabular Data

Adapting a diffusion model from image data to tabular data involves similar principles while considering the structured nature of tabular datasets. In the case of image data, the model progressively applies Gaussian noise to an initial image, learning to restore it to its original form. For tabular data, the process evolves to feature-level noise addition, where each feature value is perturbed by noise. Latent variables are introduced for each step, capturing the evolving data point. The model then aims to approximate the conditional distribution of the evolving data given the initial point, while training involves understanding the reverse process—estimating the probability of arriving at the previous data point from the current one. To generate new tabular data, random noise samples are iteratively denoised using the learned reverse process, offering a structured approach for creating synthetic tabular data while preserving the original dataset's characteristics [8].

2.4 TabDDPM

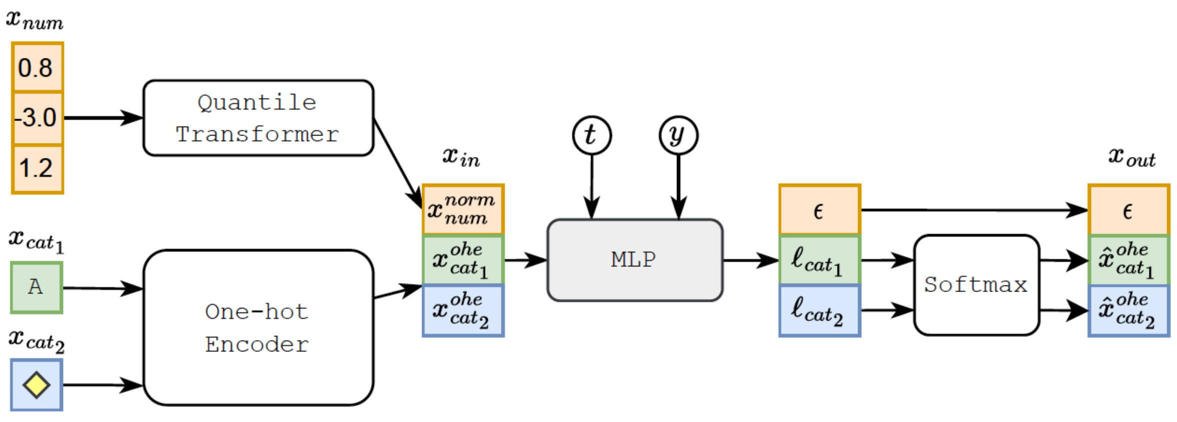


Figure 5: TabDDPM scheme for classification problems architecture adopted from [8]

In this section, we present the TabDDPM architecture and delve into its main hyperparameters, pivotal in influencing the model's efficacy [8]. TabDDPM employs multinomial diffusion to handle categorical and binary features, while Gaussian diffusion is utilized for numerical attributes. Specifically, for a tabular data instance x consisting of N_{num} numerical features and C categorical features, the model processes one-hot encoded categorical features and normalized numerical attributes. The input dimensionality x_0 is $(N_{num} + \sum K_i)$ with preprocessing involving Gaussian quantile transformation. Each categorical feature follows an independent forward diffusion process, while the reverse diffusion step is modeled by a multi-layer neural network, predicting Gaussian and multinomial diffusion outcomes. The TabDDPM model for the classification problems is schematically presented on figure 5. The details about TabDDPM architecture will be found in [8]

2.5 Process of Diffusion Model

Consider a diffusion process commencing at time = 0 and concluding at time = 1, referred to as the "diffusion time." For this purpose, We opt for a continuous approach to the diffusion time, which offers the flexibility to modify the number of sampling steps during inference, as opposed to discrete variations more typical in diffusion models.

An imperative component in this context is a function that furnishes noise and signal levels throughout the diffusion process, aligning with the prevailing diffusion time. This function, termed the "diffusion schedule" (accessible via `diffusion_schedule()`), produces two essential factors: the `noise_rate` and the `signal_rate`. These correspond respectively to $\sqrt{(}1 - \alpha)$and $\sqrt{(}\alpha)$ as featured in the DDIM paper [11]. The process of generating a noisy image entails a combination of weighted random noise and the training image. These components are aggregated in accordance with their respective rates.

It's noteworthy that the noise rates and signal rates are akin to the standard deviation of their counterparts within the noisy image, while the squares of these rates correspond to their variances (akin to power in signal processing). An important property is that the squared sum of these rates always equals 1, ensuring that noisy images consistently maintain a unit variance, similar to their unscaled constituents.

We will utilize a streamlined continuous variant of the cosine schedule [12] that finds widespread usage in existing literature. This schedule maintains symmetry and exhibits gradual changes at both the commencement and conclusion of the diffusion process.

2.6 Training Process

The training process for denoising diffusion models (outlined in the `train_step()` and denoise() functions) follows these steps: random diffusion times are uniformly sampled, and the training samples are combined with random Gaussian noises at rates aligned with the corresponding diffusion times. Subsequently, the model is trained to effectively separate the noisy sample into its constituent elements.

Typically, the neural network is instructed to forecast the unscaled noise component, which facilitates the computation of the predicted image component through utilization of the signal and noise rates. In this scenario, the mean squared error serves as the chosen loss function, demonstrating improved performance outcomes on this specific dataset.

2.7 Reverse Version

During the sampling process (depicted in `reverse_diffusion()`, at each iteration, we employ our network to segregate the previous estimation of the noisy sample into its underlying sample and noise components. Following this separation, we then reunite these components utilizing the signal and noise rates associated with the subsequent step.

The parallel concept is depicted in Equation 12 of [11].

The provided example specifically implements the deterministic sampling procedure outlined in DDIM, which aligns with the parameter $eta = 0$ in the paper. Alternatively, stochastic sampling can also be employed (resulting in the model becoming a Denoising Diffusion Probabilistic Model or DDPM [7]), wherein a portion of the predicted noise is substituted with either the same or an increased amount of random noise.

Notably, stochastic sampling can be implemented without necessitating network retraining (since both models are trained similarly). This approach has the potential to enhance sample quality; however, it typically requires more sampling steps.

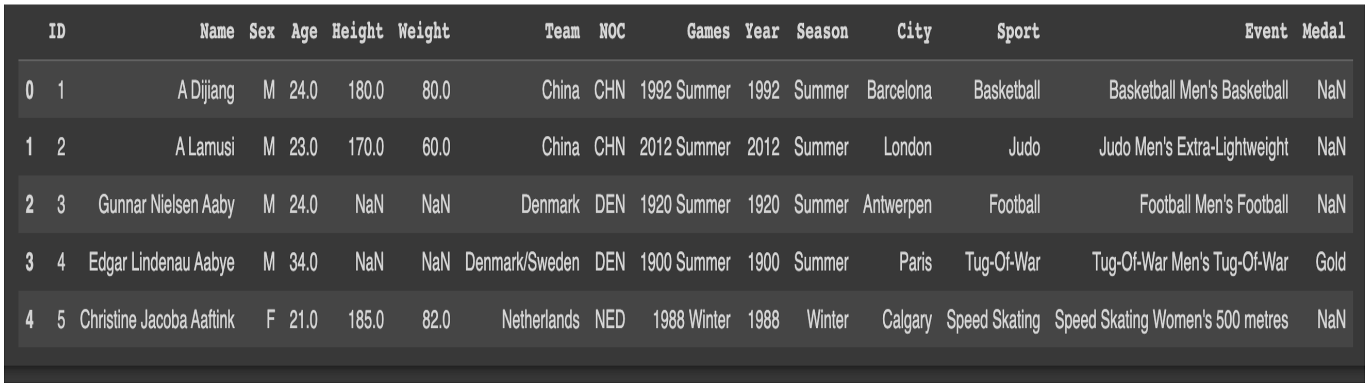
2.8 Generate Data

The implemented diffusion model encompasses various essential steps to generate synthetic data. The process begins with perturbing a given data point using incremental Gaussian noise, followed by learning the inverse transformation to restore the data to its noise-free state. The diffusion process is characterized by a continuous diffusion time, offering flexibility in adjusting the number of sampling steps during inference.

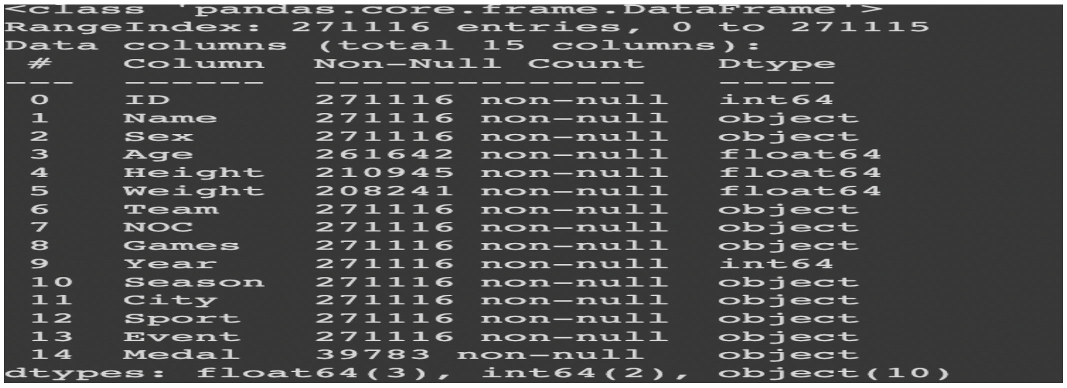
A cosine-based diffusion schedule is adopted to guide noise and signal levels throughout the process. Training involves mixing training images with random Gaussian noise according to the diffusion times, then training the model to disentangle the noisy image into its underlying components. This involves predicting the unscaled noise component, from which the sample component can be reconstructed. Sampling during inference involves iteratively separating noisy sample into original sample and noise components using the model and recombining them with the signal and noise rates of subsequent steps. Optionally, stochastic sampling, akin to Denoising Diffusion Probabilistic Models (DDPMs), can be employed to further enhance sample quality. Finally, the model's performance can be visually evaluated by generating synthetic data for assessment during training epochs.

3 Dataset

The dataset contains information about athletes who participated in Olympic events up until the 2016 games. It comprises 271,116 rows and 15 columns, with each row representing an athlete's involvement in a particular Olympic event referred to as "athlete-events." This dataset was selected due to its substantial size. An additional reason for choosing the Olympic Dataset is that its features are not interdependent. To illustrate, it is evident from the dataset features that attributes like height and weight exhibit a robust correlation. The particulars of the dataset are outlined below.



	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #    Column   Non-Null Count    Dtype
---   ------   --------------    -----
 0    ID       271116 non-null   int64
 1    Name     271116 non-null   object
 2    Sex      271116 non-null   object
 3    Age      261642 non-null   float64
 4    Height   210945 non-null   float64
 5    Weight   208241 non-null   float64
 6    Team     271116 non-null   object
 7    NOC      271116 non-null   object
 8    Games    271116 non-null   object
 9    Year     271116 non-null   int64
 10   Season   271116 non-null   object
 11   City     271116 non-null   object
 12   Sport    271116 non-null   object
 13   Event    271116 non-null   object
 14   Medal    39783 non-null    object
dtypes: float64(3), int64(2), object(10)
```

Figure 6: Data Information

When implementing the diffusion model on tabular data, we focus solely on the continuous attributes present in the Olympic Athletes dataset. These attributes include Age, Height, Weight, and Year.

4 Method

4.1 Hyperparameters

Hyperparameters play a crucial role in the diffusion model, as evidenced by their significant impact on the model's effectiveness in our experiments. The principal hyperparameters, along with their recommended search spaces, are outlined in Table 1. We advise utilizing these values. Further details on the tuning process are elaborated in the experimental section.

Parameter	**Value**
Dataset Name	"df2"
Dataset Repetitions	5
Number of Epochs	5
Sample Size	4
Plot Diffusion Steps	20
Min Signal Rate	0.02
Max Signal Rate	0.95
Embedding Dimensions	32
Max Embedding Frequency	1000.0
Widths	[32, 64, 96, 128]
Block Depth	2
Batch Size	32
Exponential Moving Average (EMA)	0.999
Learning Rate	1e-3
Weight Decay	1e-4

Table 1: Configuration Parameters

4.2 Network Architecture

In our specific scenario of handling tabular data, we're crafting a tailored neural network architecture focused on denoising tasks. This involves a well-thought-out choice: we're adopting a Multi-Layer Perceptron (MLP) structure.The rationale for selecting an MLP (Multilayer Perceptron) is rooted in a recent publication, TabDDPM [8], where MLPs were chosen as the standard architecture for tabular datasets. Furthermore, MLPs demonstrate proficiency in handling nonlinear datasets, a quality that aligns with our situation, given that the Olympic dataset displays nonlinear characteristics. Within this setup, we're carefully considering two main inputs essential for denoising: the noisy data points and the inherent noise variances that define their elements.

The importance of these variance values is significant, as they drive various operations depending on the varying levels of noise in the data. This adaptive strategy ensures that the network's denoising approach is in sync with different noise levels, leading to accurate results.

To further enhance the network's ability to handle subtle noise variations, we introduce a creative concept called "sinusoidal embeddings" [[9, 10]]. These work like special markers to help the network detect different noise patterns. These embeddings are similar to the markers used in other architectures like transformers and NeRF, which are used to understand data.

The inclusion of these "sinusoidal embeddings" is achieved without using a Lambda layer. Instead, we've seamlessly integrated them into the network. This choice boosts the network's capability to manage and counteract varying noise levels, resulting in denoising that's both accurate and high in quality.

In summary, the synthesis of our architectural choices and strategic considerations yields an MLP design adept at addressing denoising challenges within the realm of tabular data. This architecture is meticulously customized to our distinct context, incorporating vital elements such as sample size and hidden units. Through this thoughtfully engineered design, we embark on a quest to harness neural networks' capabilities for resilient denoising, with the potential to yield accurate and trustworthy outcomes. Regrettably, the anticipated results were not attained; the quality of the generated data points diverges significantly from that of the actual real-world data.

4.3 Model Summary

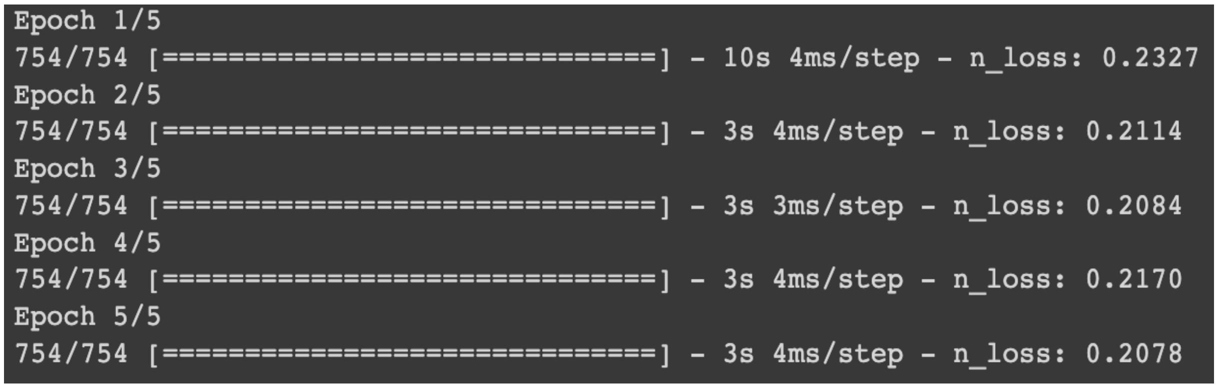
The "MLP" model we've developed is a neural network designed to carry out denoising tasks. Its architecture encompasses a sequence of layers, intricately designed to process input data and noise variances, ultimately producing refined denoised outcomes. The model's journey begins by accepting data points through the "input 1" layer, which are then flattened for efficient processing. Concurrently, noise variances are absorbed via the "input 2" layer. These inputs are intelligently merged through the "concatenate" layer, enabling the model to leverage both data and noise insights.

The heart of the model resides within its Dense layers: "dense," "dense 1," "dense 2," and "dense 3." Each layer progressively refines the input, uncovering hidden patterns and relationships. The culmination of these layers results in an output layer with 4 units, representing the denoised results. Throughout its architecture, the model dynamically learns 42,948 parameters during training, facilitating its ability to understand data intricacies and deliver precise denoising outcomes.

In essence, the "mlp" model embodies the fusion of data-driven insights, noise adaptation, and neural network power, all orchestrated to achieve unparalleled denoising accuracy and quality.

5 Results

5.1 Training Process(Epochs)



```
Epoch 1/5
754/754 [==============================] - 10s 4ms/step - n_loss: 0.2327
Epoch 2/5
754/754 [==============================] - 3s 4ms/step - n_loss: 0.2114
Epoch 3/5
754/754 [==============================] - 3s 3ms/step - n_loss: 0.2084
Epoch 4/5
754/754 [==============================] - 3s 4ms/step - n_loss: 0.2170
Epoch 5/5
754/754 [==============================] - 3s 4ms/step - n_loss: 0.2078
```

Figure 7: Training Process

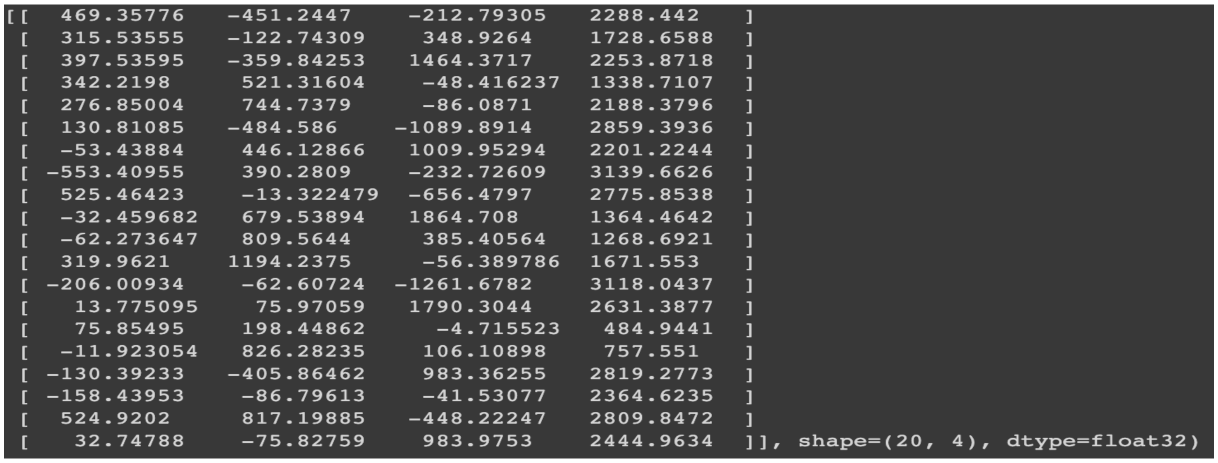
The model is trained over multiple epochs to improve its denoising capabilities. Each epoch represents a full iteration through the training data. The displayed output in figure 7 provides information about the progress of the training process. For instance, "Epoch 1/5" indicates the first epoch out of a total of five. The "`n_loss`" value, which represents the normalized loss, starts at 0.2327 in the first epoch. As the training progresses, the normalized loss decreases, indicating that the model is learning to reconstruct clean data from noisy input.

5.2 Generating Synthetic Data

Upon completing the training phase, the trained model is employed to produce synthetic data. This generated synthetic data comprises numerous data points, each characterized by four numerical attributes. These attributes pertain to distinct features of the synthetic data points. The output, presented in figure 8, illustrates the generated synthetic data, highlighting a deviation from the likeness of actual data points. Regrettably, the generated values for attributes such as height, weight, age, and year display negativity, which contradicts their expected non-negative nature. This outcome falls short of our anticipated results; however, it doesn't diminish the technical skills acquired during the capstone project.

In essence, this process demonstrates the efficacy of the diffusion model in removing noise from data. However, in practice, the synthetic data generated did not closely resemble the characteristics of the real data as expected. This approach is valuable when real data is scarce or constrained by privacy concerns, as synthetic data can serve as a viable alternative for analysis, insights, and predictive tasks.

The TensorFlow code utilized in this project has been tailored from the source provided in the reference [13]. This adaptation has been instrumental in shaping the structure and



```
[[  469.35776   -451.2447    -212.79305   2288.442   ]
 [  315.53555   -122.74309    348.9264    1728.6588  ]
 [  397.53595   -359.84253   1464.3717    2253.8718  ]
 [  342.2198     521.31604    -48.416237  1338.7107  ]
 [  276.85004    744.7379     -86.0871    2188.3796  ]
 [  130.81085   -484.586    -1089.8914    2859.3936  ]
 [  -53.43884    446.12866   1009.95294   2201.2244  ]
 [ -553.40955    390.2809    -232.72609   3139.6626  ]
 [  525.46423    -13.322479  -656.4797    2775.8538  ]
 [  -32.459682   679.53894   1864.708     1364.4642  ]
 [  -62.273647   809.5644     385.40564   1268.6921  ]
 [  319.9621    1194.2375     -56.389786  1671.553   ]
 [ -206.00934    -62.60724  -1261.6782    3118.0437  ]
 [   13.775095    75.97059   1790.3044    2631.3877  ]
 [   75.85495    198.44862     -4.715523   484.9441  ]
 [  -11.923054   826.28235    106.10898    757.551   ]
 [ -130.39233   -405.86462    983.36255   2819.2773  ]
 [ -158.43953    -86.79613    -41.53077   2364.6235  ]
 [  524.9202     817.19885   -448.22247   2809.8472  ]
 [   32.74788    -75.82759    983.9753    2444.9634  ]], shape=(20, 4), dtype=float32)
```

Figure 8: Synthetic Data

functionality of our project's codebase. By drawing inspiration from the referenced source, we have effectively implemented and customized the diffusion model to suit our specific goals and requirements. This adaptation has played a pivotal role in our project's development, enabling us to explore the potential of the diffusion model for generating synthetic tabular data.

6 Conclusion

In conclusion, this capstone project has delved into the realm of generative models with a specific focus on the diffusion model, aiming to apply practical proficiency in TensorFlow to real-world scenarios. The project's central objective was not only to enhance technical skills but also to contribute to the field of generative modeling, particularly in adapting the diffusion model to the unique demands of tabular data generation.

The exploration began with a comprehensive literature review, elucidating the significance of generative AI models and the foundational principles of diffusion models. The diffusion model's adaptation from image to tabular data was carefully outlined, bridging the gap between structured and unstructured datasets. A notable highlight was the presentation of TabDDPM, a generative model tailored for tabular data tasks, offering insights into its architecture and potential applications.

The dataset utilized for experimentation contained information about Olympic athletes, setting the foundation for the methodology that follows. Subsequently, the customized network architecture was presented, shedding light on the essential components driving the denoising capabilities of the diffusion model. It's important to note that hyperparameter tuning was not attempted in our case. This could potentially be the next step in our pursuit to achieve the anticipated results.

In essence, this project has achieved its twin goals of enhancing technical acumen and contributing to generative modeling. By adapting the diffusion model to tabular data and documenting the journey comprehensively, it has laid the foundation for future advancements in the realm of synthetic data generation and generative modeling techniques.

In addition to the accomplishments of this project, there remains a promising avenue for future exploration – the implementation of the diffusion model on a check image dataset to generate synthetic data. While this project has concentrated on tabular data, the diffusion model's potential extends to diverse domains, including image data. By applying the diffusion model to a collection of check images, it becomes feasible to generate synthetic check images that closely mimic real-world samples.

References

[1] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling tabular data using conditional gan,” 2019.

[2] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, “Argmax flows and multinomial diffusion: Learning categorical distributions,” in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., vol. 34, pp. 12454–12465. Curran Associates, Inc., 2021. `https://proceedings.neurips.cc/paper_files/paper/2021/file/67d96d458abdef21792e6d8e590244e7-Paper.pdf`.

[3] Y. Zhang, N. A. Zaidi, J. Zhou, and G. Li, “Ganblr: A tabular data generation model,” in *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 181–190. 2021.

[4] “Introduction to diffusion models for machine learning.”.

[5] L. Weng, “What are diffusion models?,” *lilianweng.github.io* (Jul, 2021) . `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`.

[6] A. N. Karagiannakos, Sergios, “Diffusion models: toward state-of-the-art image generation,” *https://theaisummer.com/* (2022) .

[7] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” 2020.

[8] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, “Tabddpm: Modelling tabular data with diffusion models,” 2022.

[9] P. Dufter, M. Schmitt, and H. Schütze, “Position information in transformers: An overview,” 2021.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.

[11] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” 2022.

[12] A. Nichol and P. Dhariwal, “Improved denoising diffusion probabilistic models,” 2021.

[13] A. Béres, “Deep diffusion models (ddim) - keras documentation.” `https://keras.io/examples/generative/ddim/`, 2022/06/24.