Advancing Diffusion Models for Human Activity Recognition

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science

by

Sirut Buasai[1]
Jason Dykstra[1]
Dillon McCarthy[1]
Cindy Trac[1,2]

April 27, 2023

Submitted to:
Faculty Advisor: **Professor Elke Rundensteiner**[1,2]
PhD Student Mentors: **Walter Gerych**[2] and **Joshua DeOliveira**[2]

[1]Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA
[2]Department of Data Science, Worcester Polytechnic Institute, Worcester, MA

Abstract

Human activity recognition (HAR) is a challenging problem that involves classifying human actions based on sensor data. Diffusion models are a promising approach for tackling such challenges by learning underlying patterns in the data. However, traditional diffusion models have limitations in handling discrete data, which is common in HAR datasets. To address this, we developed a tabular diffusion model that can handle discrete data and applied it to the UCI HAR dataset. Our tabular diffusion model achieved higher classification accuracy compared to the vanilla diffusion model and was comparable to current state-of-the-art models. The findings have significant implications for the potential of diffusion models in HAR as well as the development of intelligent systems for monitoring human behavior in real-world scenarios.

Acknowledgements

This project would not have been possible without the help of the community around us. The team would like to extend our utmost thanks to:

- Our advisor, **Professor Elke Rundensteiner**, for organizing this project and giving us the opportunity to work on cutting-edge technology. We are better as a team and project because of your advice and encouragement.
- Our PhD mentors, **Walter Gerych** and **Joshua DeOliveira**, for their guidance and support throughout the whole project. Thank you for your mentorship and willingness to answer our many, many questions!
- **Nattakarn Tapasanan**, for her help with our project presentation poster.
- **WPI Academic and Research Computing**, for generously allowing us to use their resources during the course of our project.

Contents

1 Introduction

Human Activity Recognition (HAR) refers to the process of identifying and categorizing physical activities performed by individuals using sensor data. The identified activities can encompass a broad spectrum ranging from basic tasks like sitting, standing, or walking, to more specific activities like hiking or swimming. The sensor data utilized for this classification is typically collected by sensors embedded in mobile devices and other wearable technology. This information often includes data like acceleration, orientation, and position. With the proliferation of smartphones and advancements in high-precision sensors, a unique opportunity has emerged to engage with large volumes of accurate sensor data. In research and literature, public HAR datasets like the UCI, WISDM, Sussex-Huawei Locomotion (SHL), and Extrasensory datasets are among the most commonly used and popular[1].

The practical applications of HAR data span across various fields, including healthcare, security and surveillance, intelligent environments, and entertainment[2]. Notably, HAR data plays a significant role in advancing healthcare by facilitating the monitoring of patients and the identification of illnesses and diseases, such as depression[3], Parkinson's disease[4], autism[5], heart failure[6], and numerous others. These applications are of critical importance as they aim to enhance and preserve human life and rely on the availability of extensive, precise, and heterogeneous datasets.

Challenges with HAR In practical scenarios, human behaviors and patterns in physical activity are diverse and complex. To construct detailed and realistic models that are effective on real-world data, substantial annotated data that represents diverse demographics is essential. This poses two significant challenges for HAR data: the lack of data for less common activities and targeted demographics, such as the elderly.

The process of collecting and annotating datasets is often time-consuming and costly. In practice, the data is usually gathered passively through studies where participants label the activities they perform throughout the day. As such, some activities, such as sleeping, may be performed more often than other activities such as swimming or biking. This results in a considerable class imbalance between different activities, making the data more challenging to train high-performance models.

Several applications for HAR datasets target specific demographics, such as fall detection among the elderly population. Consequently, studies require data from these populations to train accurate models. This poses a challenge due to the difficulty in collecting falling data from the elderly or even individuals with physical disabilities. This is because these populations face challenges that most younger study participants do not encounter, such as reduced mobility, cognitive decline, and comorbidities, which can affect the reliability and performance of HAR models and algorithms. As such, researchers often turn to young, able-bodied individuals to collect these rarer activity data.

To address these challenges, innovative techniques are necessary to provide high-quality, realistic, and clean data without the limitations associated with traditional data collection methods, which can be expensive and time-consuming. Such techniques would enable the development of HAR models that perform optimally across a broad range of activities and demographics, thereby improving their usefulness and applicability in real-world scenarios.

Proposed Solution This study proposed a method for generating realistic human activity recognition (HAR) data using a tabular diffusion model that can handle both continuous and discrete data. Furthermore, we aimed to investigate the performance of the vanilla diffusion model on the HAR dataset and compare it to the current industry standard, the generative adversarial network (GAN). Currently, the state-of-the-art model in the HAR field is the HAR-CTGAN model, which utilizes both continuous and discrete features to generate synthetic data [7]. In a similar fashion, we developed a tabular diffusion model that can leverage both feature types from the UCI HAR dataset. We hypothesized that

the tabular diffusion model will be comparable to the CT-GAN model in terms of performance and that the inclusion of discrete features in HAR problems leads to improved results compared to using only continuous features.

Contributions Our project contributions include:

- Applying a vanilla diffusion model to the HAR domain and comparing its performance to the current industry standard – GAN.
- Implementing a tabular diffusion model that utilizes both continuous and discrete features to the HAR domain.
- Demonstrating supporting evidence that the inclusion of discrete features in addition to the continuous features yields better results than using only continuous features
- Demonstrating that the tabular diffusion model outperforms the current state-of-the-art model – CT-GAN.

2 Related Work

In recent years, the exponential growth in data collection, storage, and solicitation has led to the widespread use of machine learning across diverse domains. However, one of the most significant challenges faced by the machine-learning community is the development of models that can learn and generate realistic data. Generative models, such as the generative adversarial network (GAN) and diffusion models, offer a promising solution by generating synthetic data that closely resembles real datasets. Originally designed for image synthesis, diffusion models have evolved into versatile tools with applications in various machine learning domains beyond just art generation applications.

2.1 Generative Adversarial Networks

A generative adversarial network (GAN) is a type of neural network consisting of two distinct components, namely the generator and the discriminator. The two networks engage in an adversarial learning process in which they compete against each other to minimize and maximize their respective loss functions. The primary objective of the generator is to deceive the discriminator by producing synthetic data that is virtually indistinguishable from real data. On the other hand, the discriminator aims to accurately classify data as either real or synthetic. This adversarial relationship between the generator and the discriminator is essential for the GAN to learn and improve its generative capability over time[8].

2.1.1 Conditional Tabular GAN

One of the challenges in using generative adversarial networks (GANs) for human activity recognition (HAR) is that GANs are typically designed to generate continuous features, while HAR data often contain both continuous and discrete features that are crucial for accurate activity classification. To address this challenge, recent studies have proposed Conditional Tabular Generative Adversarial Networks (CT-GANs) as a solution. CT-GANs are a specialized form of GANs that can generate tabular data including both continuous and discrete features, conditioned on specific inputs.

CT-GANs operate by training a generator network to produce synthetic data that closely resembles real-world HAR data. The generator network is fed with a set of conditioning variables that specify the desired activity to be generated, as well as other relevant information, such as the user's age and gender. The generator then produces synthetic data that is conditioned on these inputs, including both continuous and discrete features.

Compared to traditional GANs, CT-GANs have been demonstrated to be superior. They offer several advantages for generating HAR data, including the ability to simultaneously generate both continuous and discrete features, which enables more precise capturing of the intricate patterns and dependencies in the data. Furthermore, by being conditioned on specific inputs, CT-GANs can generate synthetic data that is tailored to particular users or scenarios, offering more personalized and flexible HAR systems.

However, CT-GANs also have some limitations. GANs, in general, can be limited by unstable training behavior that may produce oscillating convergence or the inability to learn due to mode collapse. CT-GANs also require a substantial amount of high-quality training data and can be computationally expensive to train.

2.2 Evaluating Generative Models

Although the field of generative models has gained popularity in recent years, evaluating them remains a challenging task. In image generation tasks, it is often easy to inspect the generated images and compare them to the real images. However, these comparisons only measure the generative model's performance at a surface level. We still cannot explicitly interpret how the model is sampling the real distribution to generate synthetic images. This is especially harder with tabular data where visual confirmation is not plausible.

One essential evaluation technique is the classification performance between real and synthetic data. If a classifier is trained exclusively on real data, it can be quantitatively determined how the classifier performs when testing the classifier once on real data as a baseline and again on synthetic data as a test. Likewise, the reverse of this can be done, where a classifier is exclusively trained on synthetic data and then tested on both real data as a test and synthetic data as a baseline. Moreover, we can employ statistical analysis such as confusion matrix, confidence, precision, recall, accuracy scores, and F1 scores to evaluate the generative model performance.

3 Background

In this background section, we will provide an overview of diffusion models, with a focus on tabular diffusion and Gramian Angular Fields, and their applications in various fields.

3.1 Diffusion Models

One potential method for generating continuous data is through the application of diffusion models. Diffusion models are a type of generative model that can be trained to simulate the evolution of a probability distribution over time. This technique can be used to produce synthetic continuous data that is comparable to authentic HAR data. The mechanics of diffusion models involve the gradual degradation of data by adding noise, followed by the learning process of restoring the original data through the reversal of the noising process. This concept is akin to denoising autoencoders, a generative architecture composed of an encoder and decoder that is utilized for data reconstruction from noise. While diffusion models are primarily implemented in the image domain, where they have demonstrated utility in enhancing image quality and generating images from text, their flexibility is their greatest asset. In comparison to other generative models, they can effectively capture the underlying structure of the original dataset in a more efficient and stable manner. Forward diffusion and reverse diffusion are two distinct methods for constructing diffusion models. Moreover, unlike GANs, diffusion models exhibit significantly less unstable training behaviors, and longer training generally yields better results.

3.1.1 Forward Diffusion

Forward diffusion is the act of adding noise to data until the data becomes almost pure Gaussian noise. We denote the number of steps of noise added to data as t, which allows us to express the data in X in terms of t in the following equation:

$$X_t = X_{t-1} + N(0, 1) \tag{1}$$

The data at time step t is defined in terms of the data at the previous time step plus some Gaussian noise. Using the Markov assumption, the equation can be written as:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2}$$

Equation 3 defines the forward process and gives the data after t timesteps. However, it requires the knowledge of the previous step. This would require numerous calculations which can get computationally exhaustive for large enough t in order to learn and calculate the parameters for the noise at different steps. Instead, the equation can be rearranged using $\bar{\alpha}$ to represent the cumulative product of $1 - \beta$.

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}}x_{t-1}, (1-\bar{\alpha})I) \tag{3}$$

This new equation allows the data at any timestep to be calculated instantaneously and is required for the reverse diffusion process for calculating loss and comparing estimated values to the true values[9].

3.1.2 Reverse Diffusion

Reverse diffusion is a method for recovering original data from noisy data by gradually learning the quantity of noise that was added at each stage. The model estimates the data distribution at a specific time step t and employs Kullback-Leibler (KL) divergence in the loss function to evaluate the divergence from the known value of X_t obtained from the final forward diffusion equation. The model parameters are then learned through stochastic gradient descent and backpropagation processes. This process is then repeated for various time steps, which forms the training process for diffusion models. Through this training, the model attempts to learn the noise added at each time step to recover the original data. Consequently, the model can be utilized to create novel data samples that are similar to the original data using pure Gaussian noise.

3.2 Tabular Diffusion

The work referenced so far has been prior research in the field of the vanilla diffusion model, which exclusively accommodates continuous data. It is advantageous for the diffusion model to be able to generate both continuous and discrete data given that large tabular datasets are sparsely available. Furthermore, the generation of synthetic data of this nature would not be subject to common General Data Protection Regulation (GDPR) or similar statutes allowing synthetic data to be used for research purposes while protecting individual data privacy.

3.2.1 Traditional Diffusion

Diffusion models employ a noise addition process that iteratively transforms a starting distribution to a desired target distribution. In the context of HAR, we can utilize the diffusion model to learn the real data distribution through training. By modeling the evolution

of the distribution over time, diffusion models can capture the continuous dynamics of the activity.

Although diffusion models present a promising approach for generating continuous data in HAR, the current research in this field remains limited. While diffusion models can proficiently generate synthetic data with continuous features, they lack the inherent capability to produce discrete features. To address this challenge, modifications to the diffusion model architecture are required to enable both continuous and discrete feature generation. One promising approach is to leverage a tabular diffusion model that integrates both continuous and discrete features through a multinomial model.

3.2.2 Multinomial Diffusion

Multinomial Diffusion[10] facilitates the noising process of categorical variables by employing a closed-form and reversible solution. In the case of continuous data, traditional diffusion involves the addition of small Gaussian noise. However, since categorical variables can assume a finite number of classes, the addition of Gaussian noise is not applicable. Instead, the probability distribution is treated as the data to be noised, whereby a fully noisy categorical feature would possess an equal chance of selecting any of its classes. Consequently, each time categorical noise is added, the probability for each class changes, and the distribution can be resampled to acquire the noised data.

Hoogeboom defined the diffusion process using a one-hot encoding of the discrete data $x_t \in \{0,1\}^K$. They expressed the forward process with x_t where t is the number of noise steps added to the data as a categorical distribution with β_t chance of resampling the category uniformly:

$$q(x_t|x_0) = C(x_t|\bar{\alpha}_t x_0 + (1-\bar{\alpha}_t)/K \tag{4}$$

$$\text{where } \beta_t = 1 - \alpha_t$$
$$\text{and } \bar{\alpha_t} = \prod_{\tau=1}^{t} \alpha_\tau.$$

They then computed the categorical posterior $q(x_{t-1}|x_t, x_0)$ for reverse diffusion in closed-form:

$$q(x_{t-1}|x_t, x_0) = C(x_{t-1}|\theta_{post}(x_t, x_0)) \tag{5}$$

$$\text{where } \theta_{post}(x_t, x_0) = \tilde{\theta} / \sum_{k=1}^{K} \tilde{\theta}_k$$
$$\text{and } \tilde{\theta} = [\alpha_t x_t + (1-\alpha)/K] \odot [\bar{\alpha}_{t-1} x_0 + (1 - \bar{\alpha}_{t-1}/K]$$

3.3 Gramian Angular Fields

Gramian Angular Field (GAF) is a powerful method for representing time series data as a two-dimensional image that can be effectively processed by diffusion models or other image-based neural networks. By transforming data collected from sensors – such as accelerometers and gyroscopes – into images, machine learning models can be leveraged to upsample or classify time series data. In addition, GAFs offer a means to display multiple channels of data in a single image, thereby eliminating the need for multiple images and reducing model training complexity. Given the triaxial data collected by accelerometers, different axes of motion (axes x, y, and z) can be captured in a GAF by utilizing different color channels.

The process of creating GAFs begins by first representing time series data in polar coordinates rather than Cartesian coordinates. A Gramian matrix is then constructed, where each entry is the cosine of the summation of two angles formed by lines in the polar coordinate system. The conversion of time series data into polar coordinates entails encoding

each data point in the series as the angular cosine value with the timestamp as the radius[11]. The result is a graph akin to that shown in Figure 1, where each point represents a step in the time series. The plot's points that are closer to the origin of the coordinate system correspond to the earlier time series data points, whereas those towards the outer edges are the last data points in the series.

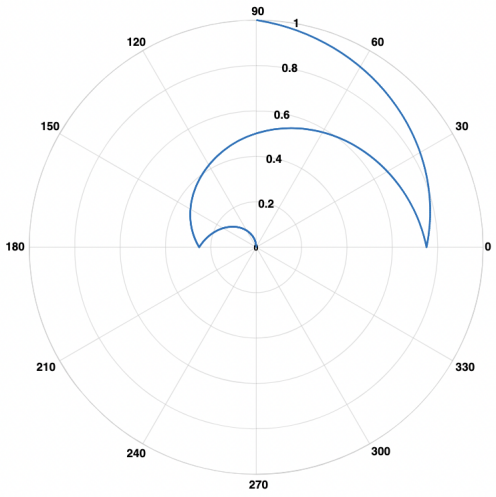


Figure 1: Polar plot of a sinusoidal time series

To create the Gramian Angular Different Field (GADF), the matrix can be defined as:

$$GADF_{i,j} = \sin(\Psi_i + \Psi_j) = tr(\sqrt{I - TS^2}) \cdot TS - tr(TS) \cdot (\sqrt{I - TS^2}) \tag{6}$$

Where $GADF_{i,j;|i-j|=k}$ represents the correlation by the difference of directions given time k. I represents the unit vector, and $tr(X)$ represents the transpose of X. This formula will result in a Gramian matrix of the size TxT.

Our team aimed to evaluate the performance of the tabular diffusion model we developed in comparison to existing alternatives. To this end, we aimed to compare classification results between GAF representation as input and the raw time series data. Despite the difference in data representation, the information encoded in the original dataset theoretically should be preserved in the transformed format.

3.3.1 Applications of GAFs

GAFs can be used as a pre-processing step to transform complex, multidimensional time-series data into images, which can then be used as input to diffusion models. In particular, GAFs are useful for applications that have cyclic patterns because it preserves the cyclic nature of the time series data. It is able to do so because of the conversion of every time point to a point on a unit circle. GAFs have many applications, including:

Time Series Classification. GAFs can serve as input to machine learning algorithms (e.g. convolutional neural networks (CNNs) or support vector machines (SVMs)) to classify time series data.[12] This is especially relevant to human activity recognition, as sensor data must be classified in order to better analyze the data.

Time Series Forecasting. In addition, GAFs can be leveraged as inputs to machine learning models to enable the prediction of future time series data. Numerous time series data exhibit periodic patterns, especially in HAR. By integrating GAFs as inputs, machine

learning algorithms can effectively learn to identify these recurring patterns and provide accurate predictions. These patterns are often pivotal in forecasting future values with a high degree of precision[13].

Feature Extraction. GAFs can effectively extract features from data. During the conversion of time series data into GAFs, features like periodicity and symmetry can be extracted. These features can subsequently be utilized as inputs for machine learning algorithms to execute regression, clustering, or classification tasks. Since GAFs are constructed through polar coordinate transformations, they are impervious to rotations and translations. This distinctive characteristic renders them less vulnerable to variations within the data.

4 Methodology

In the progression of diffusion models, two distinct approaches were pursued. The first approach centered around the adaptation of diffusion with HAR to enable the support of tabular data. This required an in-depth exploration of categorical diffusion, which necessitates an independent noising function, or a loss function that is separated from the continuous loss function. The second approach was geared towards advancing diffusion models for HAR through the transformation of data into GAFs. Subsequently, classifier models were employed to train and classify the GAF images.

4.1 Process for Tabular Diffusion

Extensive research and development efforts have been undertaken to advance the field of diffusion models for images. Nonetheless, limited progress has been made towards developing models that can be integrated with HAR or tabular databases. To initiate work on diffusion models, the "Denoising Diffusion Models" GitHub repository by Siddiqui was employed as a foundational resource[14]. The repository provides a rudimentary diffusion model for images, along with associated mathematical tools for conducting forward and reverse computations.

4.1.1 Vanilla Diffusion with Images

The initial aim of this investigation was to gain a comprehensive understanding of the denoising diffusion model code. The model was developed using PyTorch framework to realize the diffusion model principles that were initially introduced by Sohl-Dickstein et al. in 2015[15]. Accordingly, a series of experiments were conducted to simulate the diffusion process using representative image datasets, with a view to visualizing the resultant output in latent space.

The repository underwent a preprocessing and data cleaning step to enhance its comprehensibility and functionality for users. Additional visualization features were also implemented as part of a bug-fixing process for both forward and reverse diffusion. The trial tests with corrected plots are shown in Figure 2 and 3

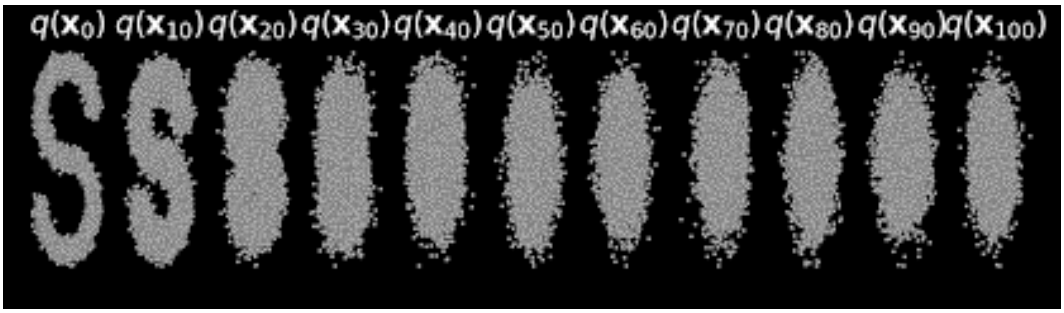


Figure 2: Forward Diffusion *adding* 100 steps of noise

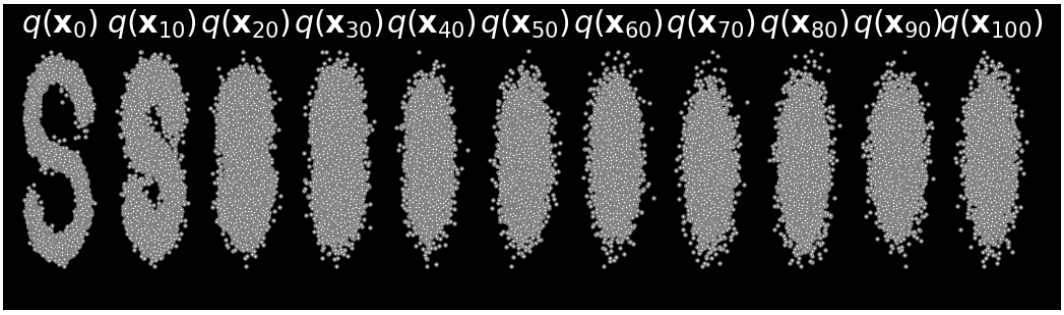


Figure 3: Reverse Diffusion *removing* 100 steps of noise

The original image data formed an S-shaped manifold in latent space. Gaussian noise was added in 100 increments with the fully noised image being the 100th time step computed from the forward diffusion process described in Equation 2. Thereafter, the reverse diffusion process was trained with deep unsupervised learning by the model over 10,000 reverse training steps. The training steps refer to a convention maintained by Siddiqui and are equivalent to epochs, which are the number of iterations over the entire data set. The result following the training is a recovered structure of the original data from pure Gaussian noise in Figure 3.

4.1.2 Noising Categorical Data

The initial step in advancing the diffusion model for tabular diffusion was understanding and implementing diffusion for categorical data. Hoogeboom et al. introduced a closed-form noising process in Equation 4; however, their implementation was based on logarithms. The preexisting mathematical models were adapted to support log-based computations, but the resulting structural incompatibilities and challenges encountered with logarithmic functions for discrete data differed significantly from the existing vanilla diffusion architecture. As such, alternate approaches to noising categorical data were pursued[16] and Jonemeth[17] before revisiting the multinomial diffusion concept and devising functional non-logarithmic functions. This entailed the following steps:

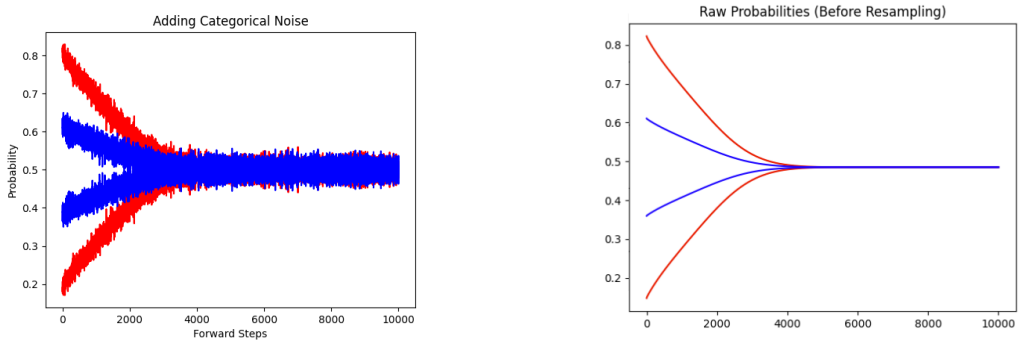
1. Indexing the classes within each discrete feature into a one-hot encoding.
2. Calculating the probability distribution from the given dataset for each class in each discrete feature.

3. Resampling a new probability distribution from a uniform categorical noise as predictions.

4. Computing the new probability distribution after t time steps of noise using Equation 4.

5. Normalizing and resampling from the distribution to obtain the noised categorical data.

Once this functionality was integrated into the code, the loss function was then tested toy data to validate its correctness. The process of noising the distributions for the probability and categorical data itself over time is presented in Figure 4, depicting two features with two classes each.



(a) Distribution After Resampling

(b) Distribution Before Resampling

Figure 4: Noising Process with Categorical Data

Each feature is depicted in a distinct color, wherein the red feature exhibits a class distribution of [0.95, 0.05] and the blue feature displays [0.6, 0.4]. Upon introducing every step of categorical noise, the probability of selecting a specific class for each feature gradually converges toward 0.5. Once approximately 4,000 steps of noise are incorporated, the categorical data is fully distorted, enabling both classes for each feature to have an equal likelihood of being chosen during resampling. Post-resampling, the distribution remains relatively uniform, albeit with some noise introduced in the thousand samples taken. Parallel experiments were conducted to authenticate the procedure for any number of features and classes.

4.1.3 Denoising Categorical Data

The denoising process must be trained and learned over time to predict the distribution of categorical variable x_{t-1} given x_0 and x_t where x_0 is the distribution with zero noise added and x_t is the distribution with t noise steps added. To accomplish this task, the training function for reverse diffusion from the vanilla diffusion model was employed, but with a distinct loss function designed specifically for the discrete features.

The greatest difficulty in developing tabular diffusion was engineering the discrete loss function. Hoogeboom et al. defined the target for reverse diffusion in Equation 5, but the obscurity in some definitions, complications with multiple features, and dimensionality elsewhere provided the greatest challenges. Particularly, the equation defines $\theta_{post}(x_t, x_0) = \tilde{\theta} / \sum_{k=1}^{K} \tilde{\theta}_k$. This equation can be interpreted in two ways:

1. A way of normalizing the target θ across all time steps.

2. A way of normalizing the target θ across each feature at every time step.

The lack of clarity in this equation posed a challenge to the implementation of the experiment. To clarify this in our paper, through experimental analysis, $\theta_{post}(x_t, x_0) = \tilde{\theta} / \sum_{k=1}^{K} \tilde{\theta}_k$ represents the normalization of target θ across each feature at every time step.

The loss function then consisted of the following algorithmic steps with all data being encoded in one-hot format $x_t \in \{0, 1\}^K$:

1. Select a random time step t for each data entry in the batch (i.e. with a batch size of 128, an equivalent of 128 forward time steps are randomly chosen).
2. Get $t - 1$ for each random time step and ensure non-negative values.
3. Calculate x_t for each time step using the noising process validated earlier.
4. Extract variance, probability of sampling uniformly, and other constants for diffusion.
5. Compute the expected value θ_t with Equation 5 at each time step.
6. Normalize each feature at every time step to get a true probability distribution for each class.
7. Input random noise into the diffusion model to get the output.
8. Compare the output with the target and measure MSE.

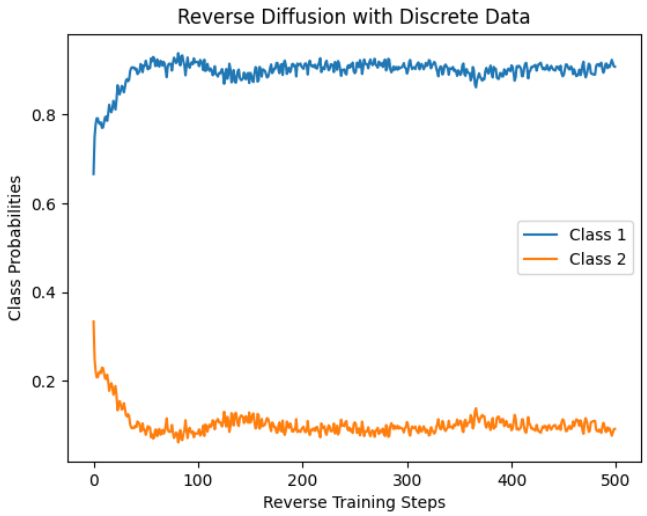


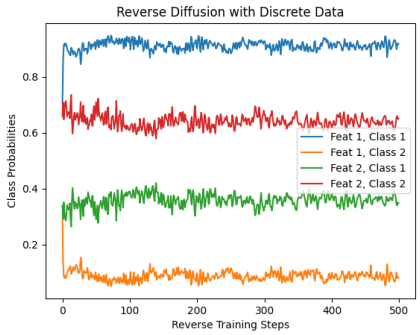
Figure 5: Reverse Diffusion with One Discrete Feature and Two Classes. Target values [0.9, 0.1]

The initial verification process evaluated the reverse diffusion approach by sampling two classes from distributions [0.9, 0.1] for a single feature. The outcome of the training is presented in Figure 5, which displays the probability of each class at each training step. Initially, the model generated probabilities close to 0.5 for selecting either class. However, it quickly learned the joint distribution's shape and accurately recovered the original distribution.

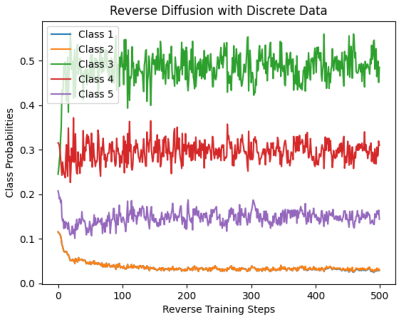
For the model to handle multiple features, the one-hot encodings of each feature were concatenated to produce a single vector of one-hot encodings. Subsequently, the model was

modified to index over this vector and normalize the distributions across each feature before generating a final vector of length $\sum_{n=1}^{N} k_n$, where N denotes the number of features and k_n indicates the number of classes in a given feature. Similarly, the loss function had to iterate through the vector to introduce noise to one feature at a time and combine the noised data into a single vector.

Finally, full validation was conducted using experimental data, comprising multiple features with diverse classes and probabilities. The reverse process during training is depicted in Figure 6, which illustrates the joint probability distributions that the model learned for each class.



(a) Reverse Diffusion with Two Discrete Features and Four Classes. Target values [0.9, 0.1, 0.35, 0.65]



(b) Reverse Diffusion with One Discrete Feature and Five Classes. Target values [0.01, 0.04, 0.5, 0.3, 0.15]

Figure 6: Denoising Process with Categorical Data

As anticipated, the convergence of the model takes longer when dealing with distributions across multiple classes. Figure 6a portrays the nearly perfect recovery of the original target of [0.9, 0.1, 0.35, 0.65], as the model learned the distribution of two features with two classes each, resulting in the final distribution of [0.9168, 0.0832, 0.3484, 0.6516]. However, with each additional class present, as depicted in Figure 6b, the model's ability to recover the original distribution diminishes slightly due to the additional variance and noise from other features. In this case, after 500 training iterations, the target of [0.01, 0.04, 0.5, 0.3, 0.15] was learned to be [0.0292, 0.0304, 0.4870, 0.3100, 0.1434]. In order to accurately recover the original distribution, it is crucial that each class has sufficient data points in each batch during resampling. Moreover, the model requires significantly more samples to model the discrete features' probability distribution when dealing with multiple features. Figure 6a employed 1,000 samples in the dataset, whereas Figure 6b employed 10,000 samples. With additional discrete features with multiple classes, there are fewer instances of a class in each training batch. This hinders the model's ability to accurately observe the total class distribution over the training period. Thus, given the sample batch size, Figure 6b displays more variance in the distributions due to the imbalances between each class within each discrete feature. With multiple discrete features, it is recommended to train the model using larger batch sizes and longer training periods to accurately model each class probability distribution.

4.1.4 Tabular Diffusion Architecture

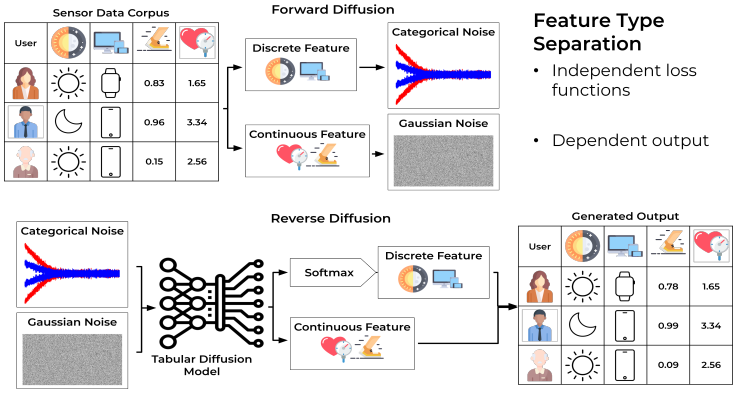
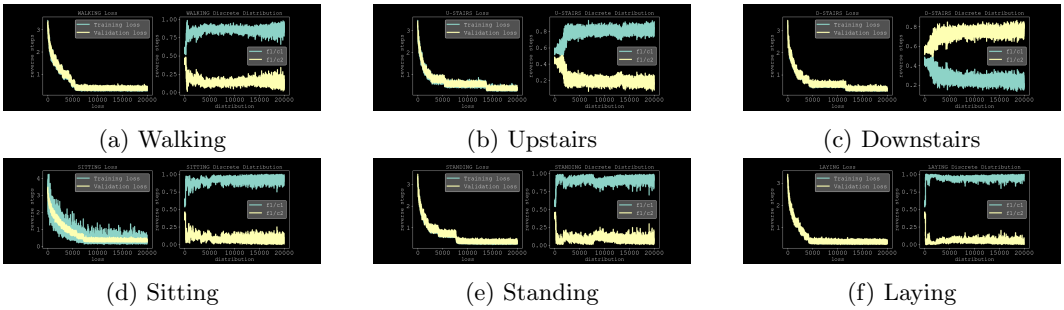


Figure 7: Tabular Diffusion Model Architecture

In our tabular diffusion model architecture, we have incorporated the aforementioned KL divergence loss function for our continuous features and a multinomial loss function for our discrete features. In the forward diffusion process, we utilized closed-form equations (Equations 2 and 4) to determine the data sampled at any given time step. In the reverse diffusion process, we separated the input based on feature type – continuous and discrete features. Subsequently, we have subjected the discrete feature to a preprocessing step for ensuring that the features are converted to one-hot encoding. Thereafter, the two feature types are concatenated and passed through three conditional linear layers to learn any relationships between the continuous and discrete features in the process. Following this, the output is duplicated into two tensors – one for continuous features and the other for discrete features. The continuous tensors are then passed through 5 linear layers with the ReLU activation functions. The resulting output is backpropagated with the KL divergence loss function using Equation 2. On the other hand, the discrete sensors are passed through 3 linear layers with the ReLU activation functions. To obtain the discrete value, the discrete tensors are finally passed through the softmax activation function. The output is subsequently backpropagated with the previously described multinomial loss function using Equation 5.

Loss Function and Discrete Feature Distribution



(a) Walking (b) Upstairs (c) Downstairs

(d) Sitting (e) Standing (f) Laying

Figure 8: Loss function and discrete feature distribution over the training period on 15 features.

Figure 8 illustrates the training and validation loss of the tabular diffusion model along-

side the discrete distribution over time. In this case, the discrete feature is a binary feature with a probability distribution of [0.95, 0.05]. As depicted in Figure 8, the tabular diffusion model works as expected given that the training and validation loss decreases over time and the discrete distribution separates from a random distribution of 0.5 into the target distribution of [0.95, 0.05].

4.2 Process for Gramian Angular Fields

To advance the study of GAFs in the context of HAR, we transformed the time series sensor data into GAF images. This approach facilitates the use of diffusion models to generate images for classifiers to train and classify data more effectively. By leveraging GAFs, we can capture the underlying dynamics of sensor data more accurately, opening up opportunities to use image-based techniques.

4.2.1 Generating GAFs with HAR

In our experiment, we utilized the UCI Human Activity Recognition Using Smartphones Dataset[18], which contains a large dataset of time series data. By converting each timestep of the accelerometer and gyroscope data into a GAF, we were able to visualize the entire dataset in a compact visual format.

4.2.2 Generating GAFs by Axis and Class

To gain a deeper understanding of potential patterns in the HAR data, we created GAFs by axis and class. We defined the input dataset to be a singular axis of total training data for each GAF and differentiated the data by the six classes present in the UCI HAR dataset (Walking, Upstairs, Downstairs, Sitting, Standing, Laying). By generating GAFs by class and axis, we were able to observe key features that differentiate the GAFs by class, such as the coloration and pattern of the GAFs.

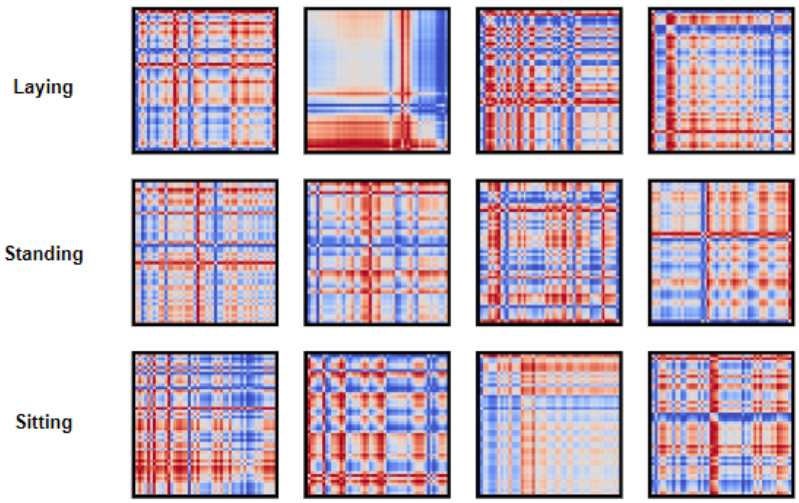


Figure 9: Randomized sample of GAFs generated on Standing, Laying, Sitting HAR X-Axis data.

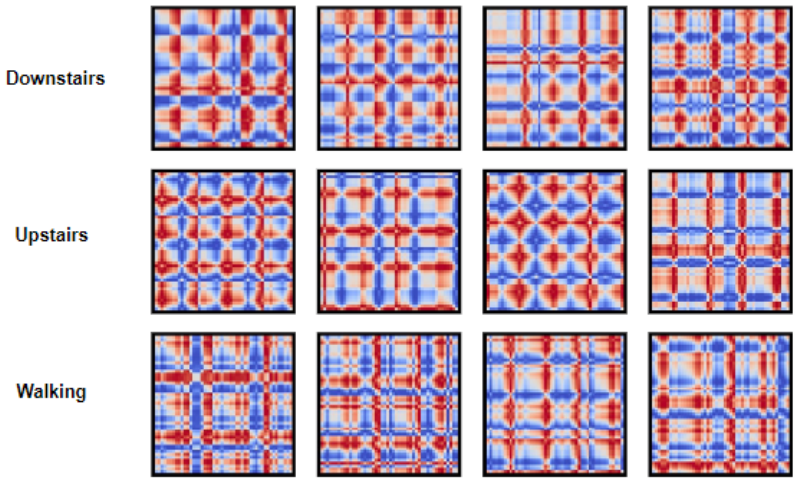
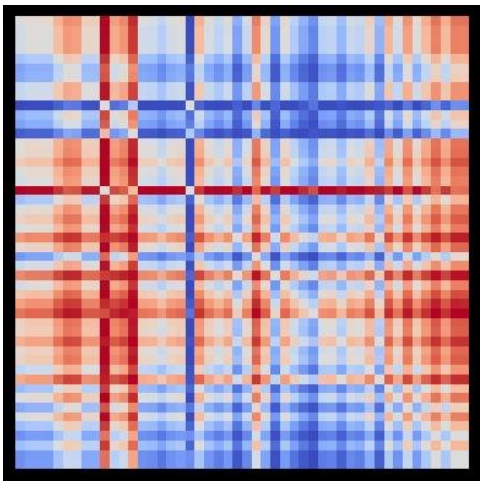
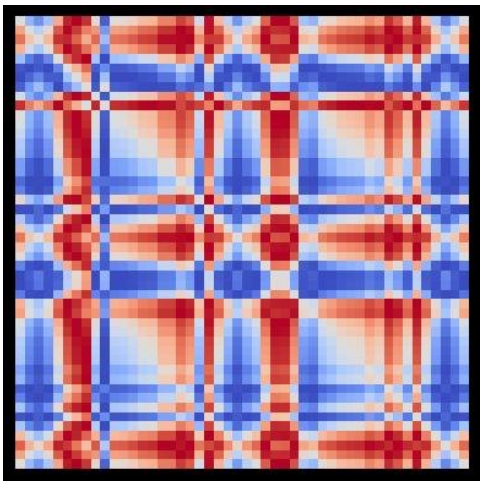


Figure 10: Randomized sample of GAFs generated on Walking, Downstairs, Upstairs HAR X-Axis data.



(a) Example of GAF image describing passive action (Sitting, X-Axis, Timestep: 4017).



(b) Example of GAF image describing active action (Upstairs, X-Axis, Timestep: 2146).

Figure 11: Passive vs. Active Action

Figure 9 and 10 illustrate the GAF representation of each HAR class. This enabled us to differentiate between passive actions (Standing, Laying, Sitting) and active actions (Walking, Upstairs, Downstairs), where similar actions produced similar GAFs. We found that active actions tend to have bolder coloring throughout the GAFs and a starburst pattern, while passive actions tend to present with more muted colors.

4.2.3 Combining Individual GAFs to Create RGB GAFs

While generating GAFs by axis allowed us to uncover some patterns in the data, it did not provide a complete picture. Therefore, we generated all X, Y, and Z-axis data into

individual GAFs by axis. We then assign a color channel to each axis (X - red, Y - green, Z - blue) and combine them into one singular RGB-colored GAF to better visualize all three axes in one representation.

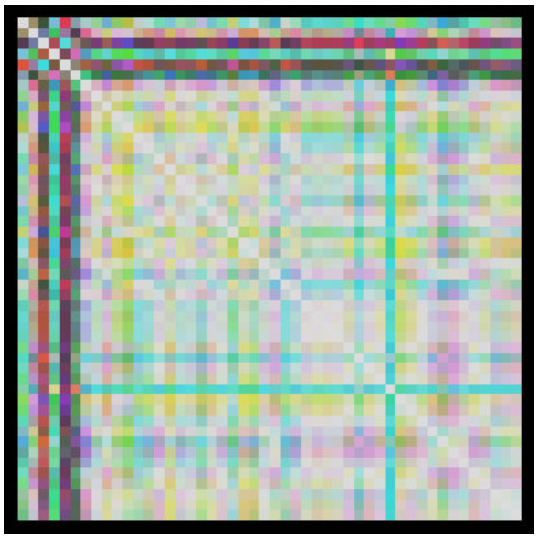


Figure 12: Example of combined axis GAF by RGB color.

4.2.4 Efficacy of GAFs vs. Time Series Data as Classifier Input

Before determining which input type to use for our diffusion model, we wanted to compare the efficacy of GAFs as input to that of the original time series data. To do so, we ran both time series data and GAFs through classifiers. We used a random forest model for analyzing time series data and a convolution neural network (CNN) for analyzing GAFs. The decision to use different models was made based on the computational requirements of the respective models and the characteristics of the input data types. For time series data, we used a random forest model because of its ability to handle large datasets efficiently and identify intricate decision boundaries. RGB GAFs were used as input to a Convolution Neural Network (CNN), which is typically used to extract higher representations for images. By using different models for different data types, the study was able to compare the efficacy of GAFs as input to that of the original time series data and draw conclusions about the effectiveness of diffusion models on the UCI HAR dataset.

5 Results

5.1 HAR Data Input Classification Evaluation

The random forest model yielded an F1 score of 0.91 when applied to the raw time series data. This is an indicator that was effective in classifying human activity based on the original time series data. Although there were other models to experiment on raw times series data with, random forest already yielded an F1 score of 0.91, even when we gave the GAFs an advantage. The data was vectorized for each row of the time series, causing it to lose its temporal information. Had we run the times series data through more complex models, it would have most likely produced better results. GAFs did not prove to be as effective for this dataset, resulting in an F1 score of 0.239 ($\pm$ 0.023). This indicates that

GAFs did not capture the underlying patterns in the data as effectively as the original time series data. Because of these results, we elected to use statistical features as input for our tabular diffusion model.

F1 Scores by Input and Model		
Input Type	Raw Time Series Data	GAFs
Model Type	Random Forest	CNN
F1 Score	0.919 (± 0.007)	0.239 (± 0.023)

Table 1: Results of using raw time series data and GAFs as input to the classifier.

5.2 Generative Model Machine Evaluation

Recent research[7] indicates that GANs are the prevalent generative model for HAR. The CT-GAN is presently considered the leading generative model for the HAR domain. Therefore, we are assessing the performance of vanilla diffusion and tabular diffusion relative to their GAN counterpart, which serves as the current benchmark in the industry. Four models, namely Vanilla GAN, Vanilla Diffusion, CT-GAN, and Tabular Diffusion, are evaluated through machine performance analysis, as shown in Table 2 and Table 3. Our machine evaluation process consists of two steps:

1. One random forest classifier is trained on real data and then evaluated on synthetic data generated by the aforementioned models.

2. Four random forest classifiers are trained on four synthetic data generated by different models and then evaluated on real data

The random forest classifier was chosen due to its efficient processing speed on a large dataset as well as its ability to find complex decision boundaries, which can be critical in classifying HAR data.

5.2.1 Machine Evaluation Performances on Different Generative Models

Table 2 and Table 3 present the classification performance of real and synthetic data using the aforementioned machine evaluation methods. The models were trained and inferred on 20 features – 19 continuous features and 1 discrete feature. The discrete feature represents the individual from whom the sensor data is collected. The feature consisted of 30 classes equating to 30 different individuals.

Model Type	F1 Scores by HAR Classes (Classifier Trained on Real Data)					
	Walking	Upstairs	Downstairs	Sitting	Standing	Laying
Baseline	0.904 (± 0.042)	0.905 (± 0.041)	0.931 (± 0.031)	0.819 (± 0.101)	0.739 (± 0.091)	0.999 (± 0.001)
Vanilla GAN	0.401 (± 0.209)	0.631 (± 0.090)	0.514 (± 0.230)	0.647 (± 0.102)	0.720 (± 0.109)	0.995 (± 0.003)
Vanilla Diffusion	0.741 (± 0.048)	0.711 (± 0.113)	0.785 (± 0.103)	0.538 (± 0.198)	0.434 (± 0.241)	0.996 (± 0.003)
CT-GAN	**0.876 (± 0.055)**	0.754 (± 0.010)	0.856 (± 0.011)	**0.715 (± 0.105)**	**0.736 (± 0.093)**	0.995 (± 0.002)
Tabular Diffusion	0.804 (± 0.048)	**0.821 (± 0.098)**	**0.883 (± 0.081)**	0.623 (± 0.215)	0.578 (± 0.222)	**0.997 (± 0.002)**

Table 2: Classifier trained on real data performance when testing on synthetic data.

Table 2 reveals that the performances of CT-GAN and tabular diffusion are similar. CT-GAN exhibits superior performance in the walking, sitting, and standing classes, whereas the tabular diffusion model outperforms in the upstairs, downstairs, and laying classes. However, it is important to note that none of the models surpass the baseline, which is the classifier's

performance on real data. In addition, the vanilla diffusion model performs better than the vanilla GAN model in four categories except for sitting and standing. Overall, the tabular versions of both GAN and diffusion models perform better than their vanilla counterparts.

Model Type	F1 Scores by HAR Classes (Classifier Trained on Synthetic Data)					
	Walking	Upstairs	Downstairs	Sitting	Standing	Laying
Vanilla GAN	0.481 (± 0.251)	0.494 (± 0.182)	0.392 (± 0.311)	0.310 (± 0.189)	0.518 (± 0.094)	0.980 (± 0.002)
Vanilla Diffusion	**0.894 (± 0.102)**	0.738 (± 0.194)	**0.877 (± 0.130)**	0.313 (± 0.233)	0.600 (± 0.253)	0.997 (± 0.003)
CT-GAN	0.785 (± 0.021)	**0.791 (± 0.112)**	0.805 (± 0.092)	0.479 (± 0.332)	0.543 (± 0.213)	0.989 (± 0.002)
Tabular Diffusion	0.725 (± 0.092)	0.771 (± 0.092)	0.750 (± 0.109)	**0.545 (± 0.312)**	**0.689 (± 0.273)**	**0.998 (± 0.002)**

Table 3: Classifier trained on synthetic data performance when testing on real data.

Based on the data in Table 3, diffusion models exhibit better performance than GAN models in five out of six classes. The vanilla diffusion model outperforms others in the walking and downstairs classes, whereas the tabular diffusion model outperforms others in the sitting, standing, and laying classes. In addition, CT-GAN performs better than other models in the upstairs class.

Overall, taking into account the results from both Table 2 and Table 3, the tabular diffusion model outperforms others in six out of twelve metrics. The CT-GAN model outperforms others in four out of twelve metrics, while the vanilla diffusion model outperforms others in two out of twelve metrics.

5.3 Tabular Diffusion Model Feature Space

Multiple feature spaces were also tested on the tabular diffusion model in this project. The following sections present the machine evaluation performances of the tabular diffusion models with various feature sets. Additionally, we conducted principal component analysis (PCA) on each feature set to visualize the dissimilarities between the distribution of real data and synthetic data.

5.3.1 Machine Evaluation Performances on Tabular Diffusion Model with Different Feature Space

Table 4 and Table 5 present the F1 score performances in UCI HAR dataset classification using the established machine evaluation methods. The random forest classifiers are trained on either real or synthetic data using 4, 10, 15, or 20 features. Subsequently, the classifiers are tested on both real and synthetic data with each of the features.

Number of Features	Testing Data	F1 Scores by HAR Classes (Classifier Trained on Real Data)					
		Walking	Upstairs	Downstairs	Sitting	Standing	Laying
4	Real	0.771 (± 0.012)	0.834 (± 0.031)	0.656 (± 0.023)	0.709 (± 0.022)	0.758 (± 0.013)	0.999 (± 0.001)
	Synthetic	0.461 (± 0.312)	0.641 (± 0.213)	0.596 (± 0.221)	0.562 (± 0.254)	0.636 (± 0.144)	0.979 (± 0.002)
10	Real	0.867 (± 0.177)	0.864 (± 0.102)	0.838 (± 0.108)	0.766 (± 0.209)	0.813 (± 0.108)	0.999 (± 0.001)
	Synthetic	0.641 (± 0.273)	0.769 (± 0.219)	0.736 (± 0.172)	**0.663 (± 0.222)**	**0.644 (± 0.146)**	0.998 (± 0.002)
15	Real	0.933 (± 0.007)	**0.930 (± 0.021)**	**0.923 (± 0.012)**	0.748 (± 0.055)	0.810 (± 0.019)	0.999 (± 0.001)
	Synthetic	**0.804 (± 0.106)**	**0.821 (± 0.107)**	**0.883 (± 0.114)**	0.623 (± 0.311)	0.578 (± 0.351)	0.997 (± 0.002)
20	Real	**0.940 (± 0.013)**	0.912 (± 0.044)	0.873 (± 0.072)	**0.821 (± 0.078)**	**0.832 (± 0.130)**	**0.999 (± 0.001)**
	Synthetic	0.802 (± 0.099)	0.785 (± 0.048)	0.854 (± 0.101)	0.623 (± 0.213)	0.598 (± 0.310)	**0.999 (± 0.001)**

Table 4: Classifier trained on real data performance when testing with 4, 10, 15, and 20 feature spaces.

Table 4 presents the classification performances on 4, 10, 15, and 20 feature spaces from the random forest classifier trained on real data. Based on Table 4, when the classifiers are tested on real data, the use of 20 features resulted in the highest F1 scores in four out of six classes, including Walking, Sitting, Standing, and Laying, while 15 features produced the highest F1 scores in the remaining two classes, which are Upstairs and Downstairs. When

the classifiers are tested on synthetic data, however, 15 features generated the highest F1 scores in three out of six classes, specifically Walking, Upstairs, and Downstairs. Moreover, 10 features produced the highest F1 scores in two out of six classes, which are Sitting and Standing, and 20 features yielded the highest F1 scores in only one out of six classes, namely Laying.

Number of Features	Testing Data	F1 Scores by HAR Classes (Classifier Trained on Synthetic Data)					
		Walking	Upstairs	Downstairs	Sitting	Standing	Laying
4	Real	0.526 (± 0.314)	0.756 (± 0.142)	0.525 (± 0.215)	0.451 (± 0.312)	0.702 (± 0.209)	0.999 (± 0.001)
	Synthetic	0.619 (± 0.122)	0.706 (± 0.106)	0.678 (± 0.165)	0.570 (± 0.308)	0.682 (± 0.151)	0.999 (± 0.001)
10	Real	0.670 (± 0.182)	0.693 (± 0.088)	0.714 (± 0.099)	**0.557 (± 0.227)**	0.687 (± 0.172)	0.999 (± 0.001)
	Synthetic	0.812 (± 0.056)	0.831 (± 0.017)	0.845 (± 0.088)	0.735 (± 0.100)	0.762 (± 0.056)	0.999 (± 0.001)
15	Real	0.725 (± 0.099)	0.771 (± 0.096)	0.750 (± 0.045)	0.545 (± 0.266)	0.689 (± 0.176)	0.999 (± 0.001)
	Synthetic	0.911 (± 0.005)	0.877 (± 0.062)	0.947 (± 0.009)	0.845 (± 0.014)	0.850 (± 0.018)	0.999 (± 0.001)
20	Real	**0.853 (± 0.077)**	**0.820 (± 0.052)**	**0.756 (± 0.142)**	0.450 (± 0.315)	**0.723 (± 0.088)**	**0.999 (± 0.001)**
	Synthetic	**0.935 (± 0.014)**	**0.930 (± 0.015)**	**0.952 (± 0.022)**	**0.874 (± 0.087)**	**0.859 (± 0.045)**	**0.999 (± 0.001)**

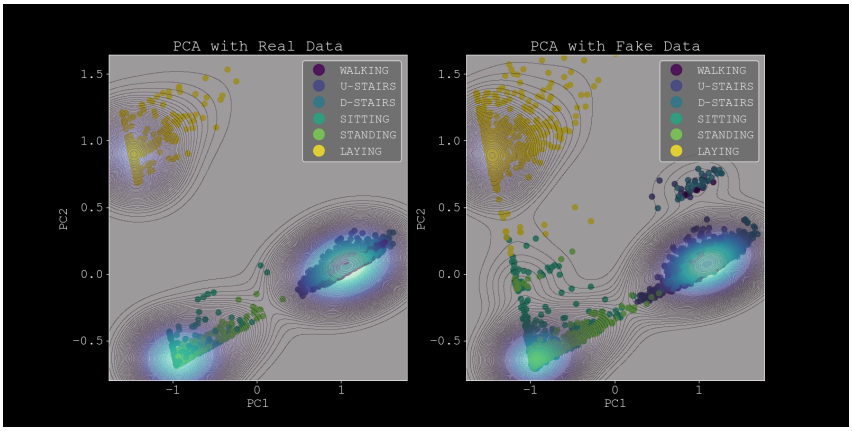
Table 5: Classifier trained on synthetic data performance when testing with 4, 10, 15, and 20 feature spaces.

Table 5 presents the classification performances on 4, 10, 15, and 20 feature spaces from the random forest classifier trained on synthetic data. Based on Table 5, the classifier performs consistently better on 20 features with one exception, namely the Sitting class when tested on real data. In this case, the classifier trained on 10 features outperformed the one trained on 20 features.

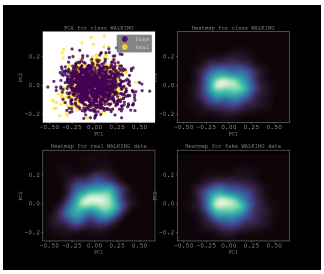
5.3.2 Tabular Diffusion Model Synthetic Data Visualization

The next section showcases principal component analysis (PCA) results on real and synthetic data sets from 4, 10, 15, and 20 feature spaces. Figures 13, 14, 15, and 16 comprise seven sub-figures each. The first sub-figure (Figure 13a, 14a, 15a, and 16a) shows the multi-class PCA between real and synthetic data. The subsequent six sub-figures (Figure 13b, 13c, 13d, 13e, 13f, 13g, 14b, 14c, 14d, 14e, 14f, 14g, 15b, 15c, 15d, 15e, 15f, 15g, 16b, 16c, 16d, 16e, 16f, and 16g) display the PCA results of each class, as well as the heat map of combined real and synthetic data, the heat map of only real data, and the heat map of only synthetic data. In each class's sub-figure, the top-left plot represents the PCA plot between real and synthetic data, the top-right plot is the heat map of combined real and synthetic data, the bottom-left plot is the heat map for only the real data, and the bottom-right plot is the heat map for only the synthetic data.

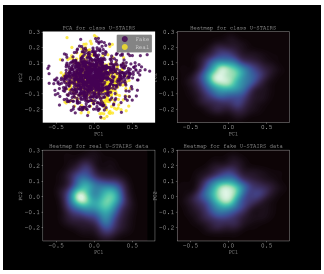
4 Features Visualization



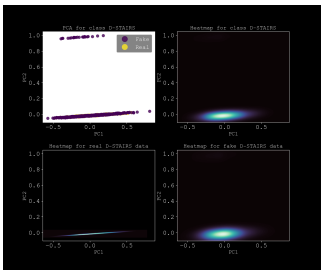
(a) 4 Features PCA



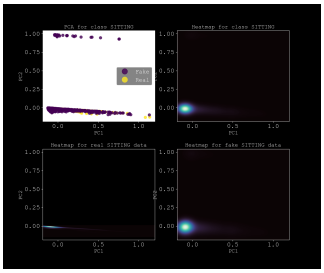
(b) Walking



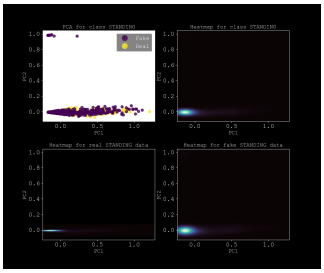
(c) Upstairs



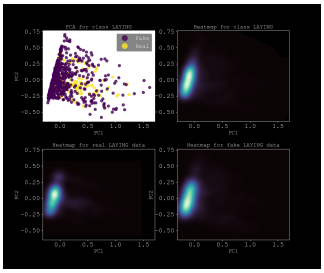
(d) Downstairs



(e) Sitting



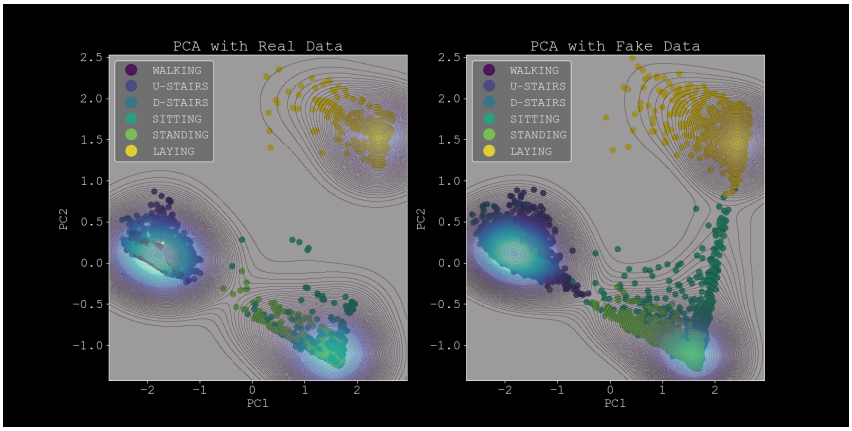
(f) Standing



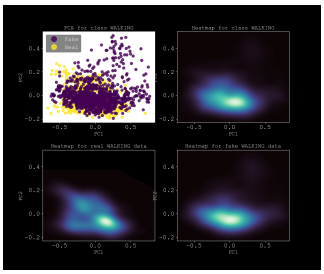
(g) Laying

Figure 13: PCA and heatmap of the six HAR classes with 4 features – Walking, Upstairs, Downstairs, Sitting, Standing, Laying – generated by the tabular diffusion model.

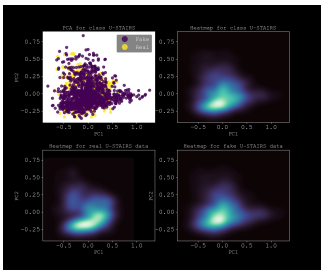
10 Features Visualization



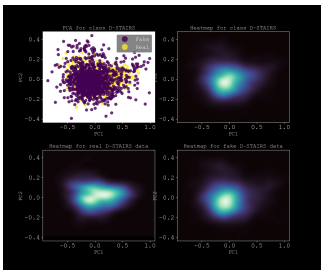
(a) 10 Features PCA



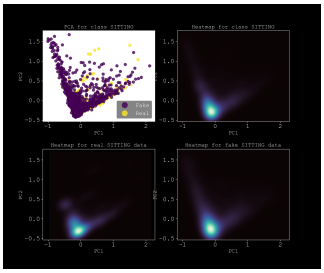
(b) Walking



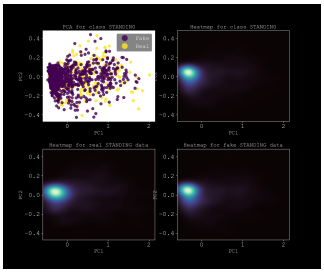
(c) Upstairs



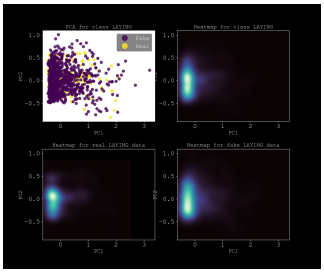
(d) Downstairs



(e) Sitting



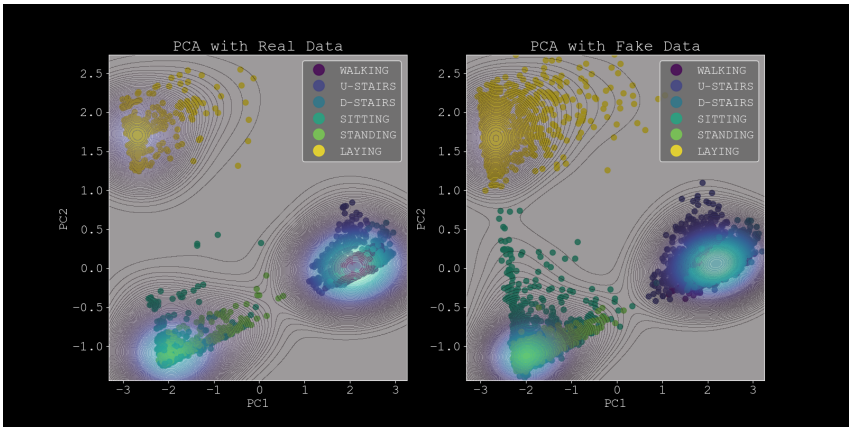
(f) Standing



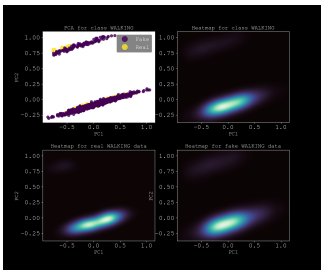
(g) Laying

Figure 14: PCA and heatmap of the six HAR classes with 10 features – Walking, Upstairs, Downstairs, Sitting, Standing, Laying – generated by the tabular diffusion model.

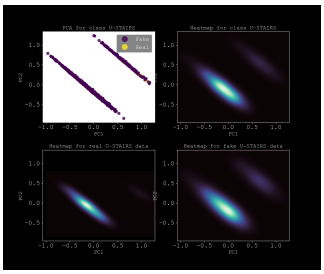
15 Features Visualization



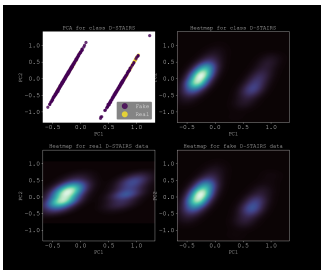
(a) 15 Features PCA



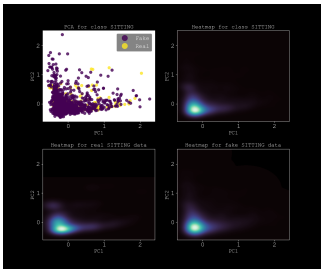
(b) Walking



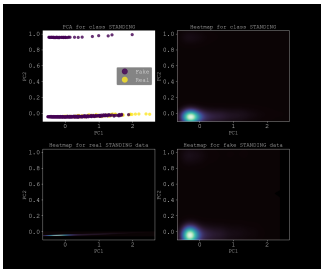
(c) Upstairs



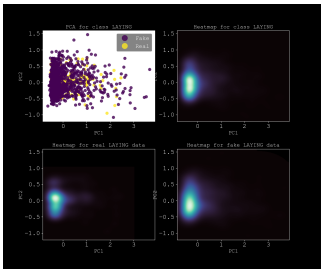
(d) Downstairs



(e) Sitting



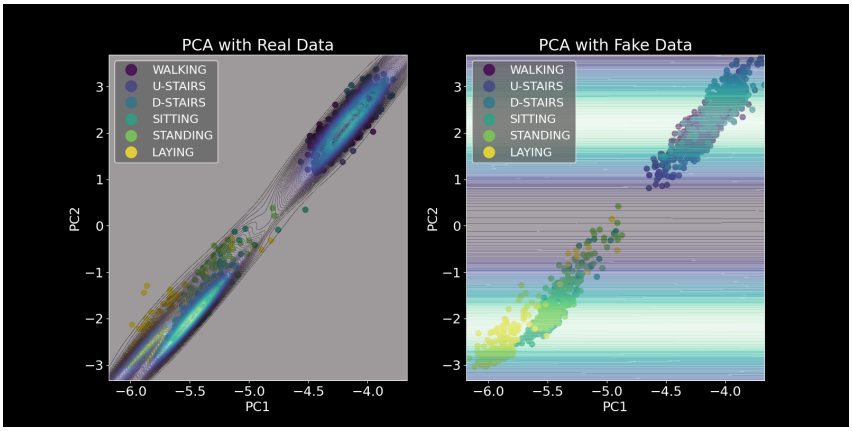
(f) Standing



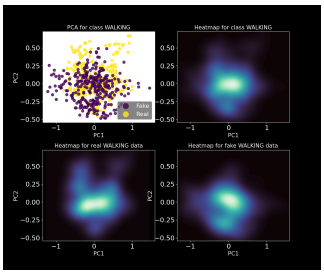
(g) Laying

Figure 15: PCA and heatmap of the six HAR classes with 15 features – Walking, Upstairs, Downstairs, Sitting, Standing, Laying – generated by the tabular diffusion model.

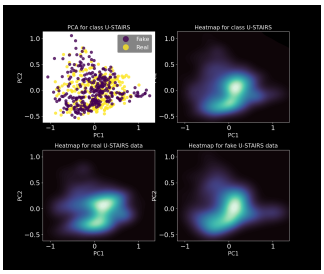
20 Features Visualization



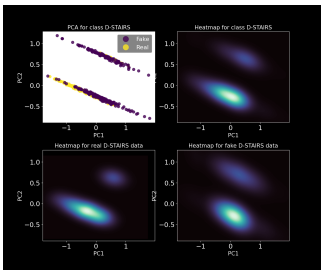
(a) 20 Features PCA



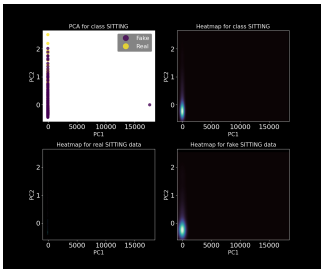
(b) Walking



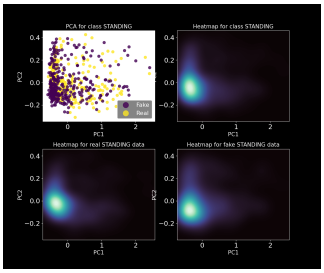
(c) Upstairs



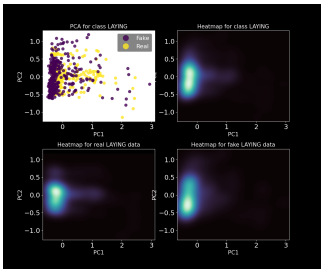
(d) Downstairs



(e) Sitting



(f) Standing



(g) Laying

Figure 16: PCA and heatmap of the six HAR classes with 20 features – Walking, Upstairs, Downstairs, Sitting, Standing, Laying – generated by the tabular diffusion model.

Based on the PCA plots for the 4, 10, 15, and 20 feature spaces, it appears that higher feature spaces result in better synthetic data quality, as evidenced by the distinct groupings and class separations in the multi-class PCA plot. Additionally, the heat map comparisons between the real and synthetic data (bottom left and bottom right heat map plot) for each class demonstrate that the synthetic data distribution closely mirrors that of the real data, providing evidence that the synthetic data quality replicates the real data distribution. While the real and synthetic data distributions are not identical, they are comparable, indicating that the model is actively learning and not generating random noise. However, it is worth noting that some classes exhibit sharper decision boundaries, as the real data distribution is often concentrated in one area while the synthetic data distribution contains some noise. This is particularly evident in Figures 13d, 13e, 13f, 15f, and 16e.

6 Discussion

During the development of our tabular diffusion model, we considered GAFs as potential input. Even though we ultimately decided to use statistical features as input to our tabular model, exploring GAFs allowed us to experiment with new technology and gain insights into its applications within the field of HAR. In this section, we discuss our work with GAFs and tabular diffusion to highlight their strengths, limitations, and potential for future research.

6.1 Gramian Angular Fields Representation

The team originally hypothesized that converting time series data to GAFs would provide a more compact (and thus easier to process) input for machine learning models to learn on. Several complications arose during and after the process of converting the data. We combined each of the three individual axis GAFs into an RGB GAF in hopes that we can better capture all of the accelerometer data of each time step in one image. However, this made the data more complex and convoluted.

One possible explanation for why GAFs did not perform well on a CNN classifier is the high dimensionality of each image. Despite using a CNN for its ability to reduce high dimensionality in images, our GAFs might have been too highly dimensioned. Like with any model, there is more work that could be done with adjusting hyperparameters and using different models. For the purposes of testing the efficacy of GAFs and time series data as viable input sources, we came to a conclusion earlier into the testing phase to be able to focus more on the tabular diffusion model.

Given more time, there are a number of expansions we could make with GAFs, including:

1. More exploratory analysis to determine which axis is a better indicator of action. This may be difficult because the orientation of the sensor is user-dependent. For instance, some users may have their phones oriented upside down in their pockets instead of upright. This impacts the data gathered for each X, Y, and Z axis of the accelerometer. This also impacts the axis (or axes) that may contain the most meaningful data on what action is being carried out. However, if we were to determine which axis provides the most meaningful information, we could focus on the GAFs generated from that specific axis data rather than trying to combine all three into one GAF.

2. Flattening the GAFs in order to minimize the complexity of using GAF images as input. We could then run classification models based on the flattened images. GAFs can be complex to work with due in part to their high dimensionality. The size of a GAF depends on the length of the given time series data. For instance, a time series of length 50 produces a 50x50 GAF containing 2500 pixels. By flattening the images into a one-dimensional vector, we can reduce their dimensionality. However, in doing so, we could lose spatial information. For certain applications, this is less of a concern if we are less focused on the spatial relationships between points.

3. Flattening the images also opens researchers up to using machine learning models that work well with the input of flat vectors. CNN was the primary model used to test the efficacy of the GAFs because it is known for reducing high dimensionality in images. Some alternative models to consider include other neural networks and support vector machines.

6.2 Tabular Diffusion Model

This study aimed to investigate the effectiveness of the tabular diffusion model on the UCI HAR dataset and compare it to vanilla diffusion and the state-of-the-art generative model for HAR – GAN. The findings indicated that the diffusion model generally performs

better than GAN on this dataset. Additionally, the tabular versions of both GAN and diffusion models showed superior performance compared to their vanilla counterparts, consistent with previous studies indicating that incorporating discrete features with HAR data can improve classification accuracy.

Based on the results, the performance of the tabular diffusion model is comparable to that of the current state-of-the-art model, CT-GAN. As indicated in Table 3, three of the four models trained on synthetic data – the vanilla diffusion model, CT-GAN model, and tabular diffusion model – were able to produce good results when tested on real data, demonstrating that the generative models effectively capture the essential characteristics of the real data and can generate synthetic data that resembles those characteristics. However, the performance of the sitting and standing classes was only slightly above the chance level across all four generative models. These two classes share very similar features, which the generative models may not be able to distinguish, as evident from the PCA comparisons between real and synthetic data.

Additionally, the results suggest that increasing the number of features for the tabular diffusion model enhances synthetic data quality, with classifiers trained on synthetic data and more features yielding better F1 scores when tested on both real and synthetic data. Nonetheless, further optimization and analysis are required to determine the optimal number of features for the tabular diffusion model due to computational limitations in the hardware.

It should be noted that the promising results achieved with the tabular diffusion model were based on a reduced feature space of the UCI HAR dataset, which was limited to only 20 features. This is due to the constraints of our implementation of the model, and we were unable to train the model for a sufficient amount of time to achieve good results with larger feature spaces. Our attempt to run the model with 40 features resulted in an inability to determine if the model was still learning after 72 hours, as well as a significant amount of NaNs in the synthetic data generated from the model.

Moreover, due to the lack of available research on the optimal hyperparameters for the tabular diffusion model in the context of HAR, we were unable to fully explore the impact of hyperparameter tuning on the model's performance. These hyperparameters, which include the number of forward and reverse training steps, the training batch size, and the learning rates for both the continuous and discrete losses, play a crucial role in determining the efficacy of the model. Further research is necessary to explore the impact of hyperparameter tuning on the performance of the tabular diffusion model in HAR applications.

7 Limitations

7.1 Gramian Angular Fields Limitations

Gramian Angular Fields provide a compact method of visualizing time series data, which is often presented in extensive JSON files. Although GAFs allow for an alternative representation of time series data, it also has a number of limitations.

First, the transformation from time series data to GAFs can result in a loss of information. The resulting GAF may not capture all details of the original data after undergoing the conversion process. The higher level of detail in the time series data, the bigger the potential information loss.

Second, generating GAFs requires significantly more computational resources. If an individual or organization does not have easy access to powerful computing infrastructure like WPI's Turing Research Cluster, it would be very difficult to run analysis on GAFs in a timely and efficient manner.

Third, the conversion process for GAFs can add to the convolution of data. More than one point in the original time series data can be mapped to the same pixel in a GAF. This

can cause some information to be lost in the GAF, making it more difficult for both humans and machines to extract meaningful patterns or insights from the data. Convolution of data can also introduce noise into the image, further complicating data analysis.

7.2 Tabular Diffusion Limitations

While the performance of the tabular diffusion model was promising, there were several limitations associated with our implementation of the models and the results we obtained.

One limitation of our implementation of the tabular diffusion model architecture is that it has a structural restriction on the input. Specifically, the discrete portion of the input must be in the label encoding format. While there are workarounds available to bypass this restriction, we have not yet found a solution to accept both one-hot and label encoding formats.

Another limitation of the tabular diffusion model is the need for retraining it when there are changes in the number of features, classes, and other built-in hyperparameters. With nine hyperparameters, this can be both an advantage and disadvantage as it enables the model to be more tailored to specific problems. However, the large number of hyperparameters makes algorithmic hyperparameter optimization challenging and time-consuming, thus affecting the training and validation process.

Furthermore, the training time required for the tabular diffusion model is significantly longer than its GAN counterpart, CT-GAN. In this study, it took approximately 72 hours to train and validate the tabular diffusion model, while the CT-GAN was able to achieve comparable results within 3 hours. Further research may be required to examine the relationship between the training time of the tabular diffusion model and its performance to determine the advantages of this model. However, we suggest that other researchers assess their computing resources to determine whether the tabular diffusion model's performance justifies its long training time.

In addition, our implementation of the tabular diffusion model has limitations on the number of features it can handle. The model is not suitable for data with more than 20 features, which is one of its major constraints. The UCI HAR dataset used in this study has more than 561 features per data point, but we achieved favorable outcomes for up to 20 features. For each additional feature, we anticipate a significant increase in training time, which prevented us from training the model sufficiently to achieve favorable results with more than 20 features.

Moreover, it is important to note that the UCI HAR dataset used in this study is highly regarded for its cleanliness within the HAR field. The dataset is accurately labeled, and the features included in the dataset are generally distinct between the six available classes. Therefore, it is reasonable to assume that the performance of the tabular diffusion model may deteriorate when applied to other real-world HAR data that may have greater variability and noise in the data.

Finally, it is worth noting that our research findings only compared the diffusion models with the GAN models. Therefore, we recommend conducting further comparisons of the tabular diffusion models with other generative models, including Variational Autoencoders (VAEs), to gain a more comprehensive understanding of their performance and potential applications.

8 Conclusion

In this study, we investigated the effectiveness of the vanilla diffusion model and tabular diffusion model on the UCI HAR dataset. We applied classifiers to both raw time series

data and GAFs and found that converting HAR data to GAFs did not significantly improve classification performance for this particular dataset. However, GAFs offer a more compact representation of time series data.

Our findings revealed that the vanilla diffusion model generally outperformed the GAN model, and including discrete features enhanced classification performance in HAR classification problems, which is consistent with previous research. In both GAN and diffusion models, their tabular counterparts – CT-GAN and tabular diffusion – achieved better classification performance than the vanilla models. Finally, we observed that the tabular diffusion model achieved results similar to the current state-of-the-art model for HAR – CT-GAN.

Our research also sheds light on the limitations of the diffusion models, such as the longer training time and limited ability to handle large feature sets. Future studies could explore ways to optimize and scale up the training process of the diffusion models for handling larger datasets

Overall, the findings of this study demonstrate the potential of diffusion models as a promising approach for tackling challenging problems in the field of HAR. Future work in this area could help unlock new opportunities for the development of intelligent systems that can analyze and understand human behavior in real-world scenarios, with applications ranging from healthcare to sports performance monitoring.

Individual Contributions

This section denotes the individual contributions of each member of the team. Note there may be some overlap when multiple team members worked in collaboration.

Sirut Buasai

- GANs and HAR
  - Established codebase documentation conventions
  - Implemented preprocessing utility methods for data loading, data cleaning, and data visualization
  - Implemented vanilla GAN models for the UCI HAR dataset
- Vanilla and Tabular Diffusion Models
  - Researched CT-GAN, HAR-GAN, Tabular Diffusion, and Multinomial Diffusion implementation and evaluation methods
  - Implemented utility classification methods for model evaluation and visualization between real and synthetic data
  - Wrote testing and visualization scripts for the multinomial loss function
  - Wrote documentation for all diffusion models
  - Bug-fixed multinomial loss function regarding multiple discrete features
  - Adapted the tabular diffusion model to work with UCI HAR dataset
  - Collected evaluation metrics for vanilla GAN, CT-GAN, vanilla diffusion, and tabular diffusion models
  - Collected evaluation metrics for tabular diffusion models across 4, 10, 15, and 20 features
  - Collected visualizations for tabular diffusion models performance on various hyperparameters settings
- Final Project Paper
  - Researched into GANs, CT-GANs, HAR, Diffusion Model, and Multinomial Diffusion Model
  - Wrote Challenges with HAR, Proposed Solution, and Contributions in the Introduction section
  - Wrote and edited Tabular Diffusion in the Background section
  - Wrote and edited Process for Tabular Diffusion in the Methodology Section
  - Wrote Generative Model Machine Evaluation and Tabular Diffusion Model Feature Space in the Results section
  - Wrote Tabular Diffusion Model in the Discussion section
  - Wrote Tabular Diffusion Limitations in the Limitations section
  - Wrote the Conclusion section
  - Edited all sections for the final draft

Jason Dykstra

- Researched into GANs via tutorials, videos, and research papers

- Helped create the infrastructure for the GAN that was used to test our HAR dataset
- Read the overview article for understanding Gramian Angular Fields (GAFs)
- Researched further to understand the math behind converting time series data to images
  - Did some more research on how GAFs work in general, and how to implement them in Python
- Overcame complications with creating RGB layered GAFs by using different axes of triaxial time series data for different color channels.
- Wrote code to remove unnecessary visual components from generated GAFs such as axes on plots
- Wrote code to generate RGB GAFs for both the triaxial training and test datasets in the provided HAR data folder. Separated generated GAFs by train/test and by class to be easily parsable by classification model.
- Included code to generate parent and child folders if they did not already exist, so that the code could be run on a new environment without issues.
- Researched into diffusion models
- Helped clean up existing code for vanilla diffusion model on image generation
- Created visual representations of initial diffusion model results, including graphs, PCA, and heat maps
- Creation of test script for running GAF classifier on Turing
- Experimented with numerous methods of implementing classifiers to test GAF vs time series data performance
- Researched other time-series-to-image implementations to minimize information loss

Dillon McCarthy

- Developed evaluation metrics for generative models, including machine evaluation and separability tests
- Tested and learned vanilla diffusion models on images
- Adapted and tested vanilla diffusion models on HAR
- Created an introductory presentation and explanation for diffusion models and developed a basic quiz to test knowledge for learners
- Modified vanilla diffusion to work for categorical data
  - Created a new noise function inspired by Hoogeboom et al. Multinomial Diffusion
  - Designed new model architecture
  - Engineered new loss function used in reverse diffusion
  - Improved data pipeline to support the concatenation of multiple discrete features through the model

  - Developed a new method of generating discrete data from the model (as opposed to continuous)
  - Tested and evaluated the model's effectiveness across multiple classes and a variety of discrete data
- Combined vanilla diffusion and categorical diffusion in a single tabular diffusion model
  - Modified model architecture
  - Constructed a joint loss function separating data
  - Invented a new generation function combining two independent methods for generating data
  - Evaluated model on toy data, including machine evaluation and PCA
- Wrote the first section of 1. Introduction, sections 3.1, 3.2-3.2.2, 4-4.1.3

Cindy Trac

- Built upon GAF generation code to regenerate GAFs on as square images
- Bug-fixed Jason's C-Term classifier for GAFs on HAR data
- Experimented with a number of classifiers to test efficacy of GAFs vs time series classification
- Implemented code to manipulate GAFs format to improve (vectorize time series, flatten GAFs, etc)
- Finalized and cleaned up Random Forest, CNN, and 1-D CNN classifiers
- Overcame many complications with Turing to be able to run all models, facilitated other team members through the process
- Collected evaluation metrics for Random Forest, CNN
- Researched and wrote:
  - Abstract
  - 1 - Introduction (HAR)
  - 2, 4, 5, 6, 7 - Introductions to Related Work, Methodology, Results, Discussion, Limitations
  - 3.3.1 Applications of GAFs
  - 4.2 - Process of Gramian Angular Fields
  - 4.2.1 - 4.2.4 Generating GAFs with HAR, By Axis and Class, Create RGB GAFs, Efficacy of GAFs vs. Time Series
  - 5.1 - HAR Data Input Classification Evaluation
  - 6.1 - Gramian Angular Fields Representation
  - 8 - Conclusion
- Edited all sections for the final draft and significantly added to:
  - 3.1 Diffusion Models

References

[1] Marcin Straczkiewicz, Phillip James, and Jukka-Pekka Onnela. "A systematic review of smartphone-based human activity recognition methods for health research". In: *npj Digital Medicine* 4 (1 2021), p. 148. DOI: 10.1038/s41746-021-00514-4.

[2] Zaheer Hussain, Quan Z. Sheng, and Weina E. Zhang. "A review and categorization of techniques on device-free human activity recognition". In: *Journal of Network and Computer Applications* 167 (2020), p. 102738. DOI: 10.48550/arXiv.1906.05074.

[3] A Barua et al. "Human activity recognition in prognosis of depression using long short-term memory approach". In: *International Journal of Advanced Science and Technology* 29 (2020), pp. 4998–5017.

[4] N Shawen et al. "Role of data measurement characteristics in the accurate detection of Parkinson's disease symptoms using wearable sensors". In: *Journal of Neuro Engineering and Rehabilitation* 17.1 (2020), p. 52.

[5] Suman Raj and Sarfaraz Masood. "Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques". In: *Procedia Computer Science* 167 (2020), pp. 994–1004.

[6] Karim Bayoumy, Mohamed Gaber, and Ahmed et al. Elshafeey. "Smart wearable devices in cardiovascular care: where we are and how to move forward". In: *Nature Reviews Cardiology* 18 (2021), pp. 581–599.

[7] Joshua DeOliveira et al. "HAR-CTGAN: A Mobile Sensor Data Generation Tool for Human Activity Recognition". In: *IEEE International Conference on Big Data* (2022). DOI: 10.1109/BigData55660.2022.10020848.

[8] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *ArXiv* (2014). DOI: 10.48550/arXiv.1406.2661.

[9] J.R. Siddiqui. *Diffusion Models Made Easy*. https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da. 2022.

[10] Emiel Hoogeboom et al. "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions". In: *arXiv preprint arXiv:2102.05379* (2021).

[11] Bilal Sadiq Ahmed et al. "A Deep Learning Spatiotemporal Prediction Framework for Mobile Crowdsourced Services". In: *Mobile Networks and Applications* 24.3 (2019), pp. 1120–1133.

[12] Chunyang Yang et al. *Multivariate Time Series Data Transformation for Convolutional Neural Network*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8700425. 2021.

[13] Cristiano Mazzoni. *How to Encode Time-Series into Images for Financial Forecasting using Convolutional Neural Networks*. https://towardsdatascience.com/how-to-encode-time-series-into-images-for-financial-forecasting-using-convolutional-neural-networks-5683eb5c53d9. 2021.

[14] J. R. Siddiqui. *Denoising diffusion model*. https://github.com/azad-academy/denoising-diffusion-model. 2022.

[15] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. 2015, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *arXiv preprint arXiv:2006.11239* (2020). URL: https://arxiv.org/abs/2006.11239.

[17] Jonemeth. *CategoricalDiffusion*. https://github.com/jonemeth/CategoricalDiffusion. 2021.

[18] Davide Anguita et al. *Human Activity Recognition Using Smartphones Dataset*. https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones. 2012.

Appendix

A Project Presentation Poster

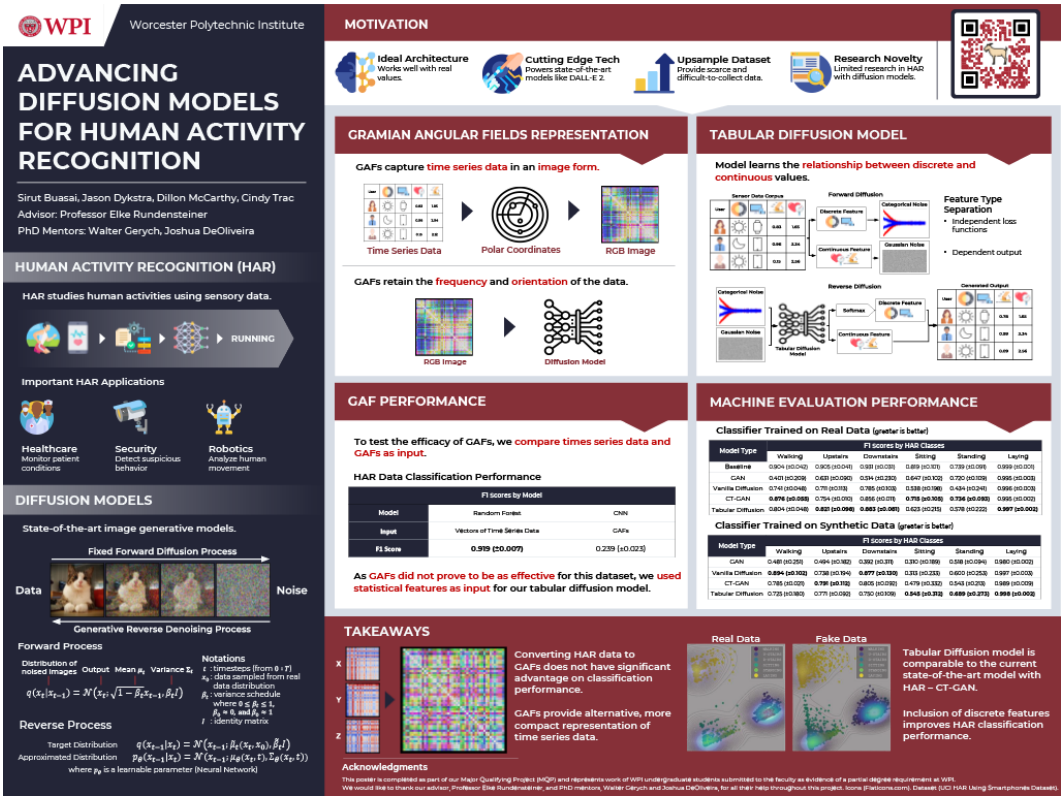


Figure 17: Our team's project presentation poster for the WPI MQP Project Day.