Synthetic Behavior Sequence Generation using Generative Adversarial Networks

FATEME AKBARI, DeGroote School of Business, McMaster University, Canada
KAMRAN SARTIPI, Department of Computer Science, East Carolina University, U.S.A
NORM ARCHER, DeGroote School of Business, McMaster University, Canada

Due to the increase in life expectancy in advanced societies leading to an increase in population age, data-driven systems are receiving more attention to support the older people by monitoring their health. Intelligent sensor networks provide the ability to monitor their activities without interfering with routine life. Data collected from smart homes can be used in a variety of data-driven analyses, including behavior prediction. Due to privacy concerns and the cost and time required to collect data, synthetic data generation methods have been considered seriously by the research community. In this paper, we introduce a new Generative Adversarial Network (GAN) algorithm, namely *BehavGAN*, that applies GAN to the problem of behavior sequence generation. This is achieved by learning the features of a target dataset and utilizing a new application for GANs in the simulation of older people's behaviors. We also propose an effective reward function for GAN backpropagation by incorporating n-gram based similarity measures in the reinforcement mechanism. We evaluate our proposed algorithm by generating a dataset of human behavior sequences. Our results show that BehavGAN is more effective in generating behavior sequences compared to MLE, LeakGAN, and the original SeqGAN algorithms in terms of both similarity and diversity of generated data. Our proposed algorithm outperforms current state-of-the-art methods when it comes to generating behavior sequences consisting of limited-space sequence tokens.

ACM Reference Format:
Fateme Akbari, Kamran Sartipi, and Norm Archer. 2022. Synthetic Behavior Sequence Generation using Generative Adversarial Networks. 1, 1 (September 2022), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Health monitoring of older people with the aim of providing on-time care and health condition prediction has received a considerable amount of study. The availability of datasets on older people's daily behavior can benefit a large body of studies including: applications of machine learning methods on predicting and detecting anomalous behaviors [6, 28, 31, 36, 41, 50, 55], predicting health conditions [18] or predicting clinical health scores [11]; and development of reminder and recommender systems in healthcare support and the supervision of long-term behavior [8, 25, 65]. Furthermore, the efficiency and effectiveness of deep learning methods depend on the quality and quantity of training data. Due to the following reasons, existing datasets of real data do not meet the requirements of research in this area: i) *scale of data*: training of machine learning models tends to require large amounts of data; ii) *privacy of data*: health monitoring poses privacy concerns to the people whose activities are being recorded [27]; and iii) *labeled data*: supervised models need labeled data for training, and labeling data is a tedious and time-consuming task.

Authors' addresses: Fateme Akbari, DeGroote School of Business, McMaster University, Hamilton, ON, Canada, akbarif@mcmaster.ca; Kamran Sartipi, Department of Computer Science, East Carolina University, Greenville, NC, U.S.A; Norm Archer, DeGroote School of Business, McMaster University, Hamilton, ON, Canada.

Manuscript submitted to ACM

That being said, synthetic data generation methods have been considered extensively for simulation studies. In particular, the current safety concerns imposed by the COVID-19 pandemic has made it rather impossible to access older people homes to collect data for an extended period of time. Generating synthetic datasets is widely used in different domains of study such as computer vision and natural language processing to address the issue of data scarcity. Apart from model-based data generators and simulators [38, 57], Generative Adversarial Networks (GANs) have recently attracted the attention of many researchers for generating realistic images, texts, EHRs or even music based on small amounts of real data [7, 29, 46, 62, 63]. However, GANs have rarely been used for generating realistic data related to human behavior. Such a dataset could be very beneficial for health monitoring, given the sensitivity of this type of data. In this research, we apply GANs to learn the features of a real dataset [10] that consists of the daily activities of a person and mimics human behavior to generate a realistic synthetic dataset. The results of this research can be used to train various machine learning and deep learning models with the aim of predicting anomalous behavior in older people.

Our method can be used to generate behavior sequences for persons living in multi-resident houses. To accomplish this, each resident's original data must be segregated. After then, data production for each data set will be carried out. This allows for generating behavior sequences for each resident. If it is necessary to consider the behavior of others in analyzing the behavior pattern of one person, which is not the usual form of analyzing ADL patterns, the behavioral sequences produced for different people can be concatenated. In this case, we need to consider some limitations on parallel activities of two or more people at home, such as using washroom which may only be used by one person at the time, if the resident has one washroom. If the house only has one washroom, for example, residents cannot use it at the same time. To account for this constraint while creating synthetic data, post-processing of created sequences must be conducted to check for constraints and eliminate sequences that violate resource limits. Figure 1 illustrates an overall view of how such a dataset can complement the activity monitoring and analysis of older adults at home. In the "sensor data collection" layer, the sensor data are collected from a target individual at home identified from other residents at home. In case of more than one person at home, the use of RFIDs (or similar method of identification) is required to separate the activities of each person at home. Collected ground truth data will then serve as a basis for data augmentation. In "data augmentation" layer, our proposed solution comes into practice to increase the amount and diversity of data. Lastly, in "data analytics" layer, augmented data will be used in Predictive/RL models to provide care givers and old people with timely services so they can live as independently as possible without increasing the burden on the healthcare system.

We show that GANs can be used to build models that generate realistic data for human behavior by mimicking ground-truth data. Although the use of GANs in generating sensor data has been explored widely in the literature [4, 12, 15, 40, 43], generating sequences of human activities without getting into the details of each activity has remained unexplored. Also, existing models have diversity issues with generating sequences of limited-space tokens. In this research, (Figure 1) we focus on the overall behavior patterns of an old person which we believe are dependent on the intensity, order and duration of activities that can be used as the basis for detecting anomalous behavior.

The main contributions of this research are as follows: (1) We have introduced a new application for GANs. To the best of our knowledge, BehavGAN is the first effort to apply GANs in simulating older person's behavior. (2) We propose an effective reward function for adversarial network backpropagation by incorporating n-gram based similarity measures in the reinforcement mechanism. We do this to overcome the issue of generating too many identical records in sequences of limited-space tokens. (3) We improve the speed of the BLEU score calculation by deploying a hash data structure so that it can serve as a part of the reward function in the adversarial training loop. The remainder of this paper is organized as follows. In Section 2 we review related work. In Section 3 we provide some background

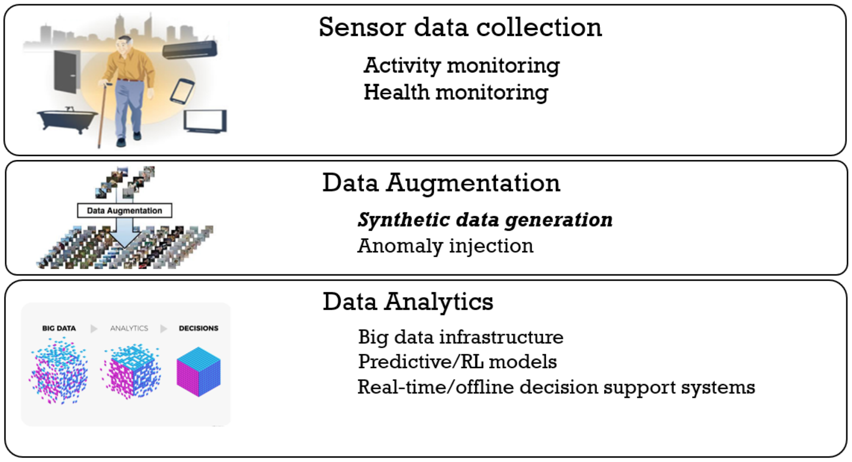

Fig. 1. Three-layer activity monitoring framework for synthetic data generation.

information. We discuss our approach in Section 4, followed by a case study in Section 5. Finally, Sections 6 and Section 7 provide discussions and concluding remarks and introduce potential future studies.

2 RELATED WORK

A broad range of data analytics solutions rely on collecting daily activity data from sensors, including wearable sensors and ambient sensors to provide actual measures of three desired services: comfort, healthcare and security [2, 5, 42]. The potential consequence of sharing sensitive personal data often prevents the adoption of large-scale data collection and hence reduces the effectiveness of these systems. Synthetic data generation has been used as an alternative to real data sharing to overcome these problems and increase data sharing. The generated data only retains the necessary statistics of the real data and is used as a replacement for selected user-sensitive real data segments, thus protecting the privacy of the person(s) being monitored. The generation of synthetic data also provides researchers the opportunity to quickly generate large scale datasets which can lead to improved data analytics. Many of current approaches to synthetic data generation are limited in terms of complexity and realism. Model-based data generation is widely used by scholars in various domains including healthcare [3, 6, 14, 49]. These models require researchers to define the patterns, rules and constraints in advance, making it difficult to build models that can generate datasets with different modes and complex patterns that occur in the real world. For example, SynSys [4] is a data generation approach based on machine learning that generates synthetic time series data using hidden Markov models and regression models that are initially trained on real datasets. The authors tested SynSys on a real annotated smart home dataset and used time series distance measures as a baseline to determine how realistic the generated data was compared to real data. The intent was also to show that SynSys produces more realistic data in terms of distance compared to random data generation, data from another home, and data from another time period. They applied SynSys when only a small amount of ground truth data was available. Using semi-supervised learning the authors demonstrated that SynSys could increase the precision of activity recognition compared to using just the limited amount of real data alone.

In addition to classical techniques, there are newer methods that use generative neural networks to create synthetic data. They have proven successful in generating various data types including photo-realistic high-resolution images

[29, 33, 60], realistic text description images [48], and even new text and music composition [22, 63]. The SeqGAN method [63] addresses two challenges in producing discrete token sequences. One key issue is that the generative model's discrete outputs make passing the gradient update from the discriminative model to the generative model challenging. Furthermore, the discriminative model can only evaluate an entire sequence. The SeqGAN method describes the data generator as a stochastic strategy in reinforcement learning (RL). The RL reward signal is generated by a GAN discriminator that evaluates a whole sequence and is then fed back to the intermediate state-action steps via Monte Carlo search.

In [1], a privacy-preserving data generation method using generative neural networks is presented for synthesizing medical texts from a de-identified clinical text corpus. The authors also conduct a thorough utility analysis in different levels to demonstrate the applicability of the proposed data generation method. In [61], a method is introduced to simulate EHRs (Electronic Health Record) composed of many health records of multiple data types by using a GAN model. The authors took feature constraints into account and incorporated key utility measures to assess the generated data. Their study of over 770,000 records from the Vanderbilt University Medical Center's EHR showed that the current model achieves better efficiency in terms of maintaining basic statistics, cross-feature correlations, latent structural properties, feature constraints and associated patterns from real data, without compromising privacy. Feng et al. [17] proposed a GAN-based method for synthesizing human mobility data. The generator, as a sequential modeling network, captures the complicated temporal transitions in human mobility. In order to generate meaningful trajectories, they strengthen the model-free generator with a model-based regional network to incorporate previous knowledge of urban structures. In [32], a method called CWGAN is proposed to generate mobile sensor data including the accelerometer, gyroscope, and magnetometer, to sense phone movements incurred by user operation behaviors. A data augmentation method for EEG emotion detection using GANs is also presented in [37]. This work focuses on generating power spectral density (PSD) and differential entropy (DE) features, which are two commonly used numerical features in emotion recognition tasks. Therefore, the proposed model generates realistic-like PSD and DE features of EEG data. As acknowledged by authors, the temporal dependency is not considered when generating the EEG features.

Generating time series using GANs with the aim of synthesizing human daily activities has attracted much attention recently. SenseGen [4] generates sensor data using an LSTM [26] (Long short-term memory)-based generator that can pass another LSTM-based discriminator model test. Using a dataset of accelerometer traces, obtained from users' everyday activities using smartphones, the authors demonstrate that the deep learning-based discriminator model can only differentiate between actual and synthesized traces in the neighbourhood of 50% accuracy. In [15], GANs were used for generating realistic simulation environments. First, they used an existing simulator that simulated user activities. Then, GANs were used to generate realistic sensor data that accompanies such activities. Results showed that a model trained on real data exhibits comparable output to a model trained on data generated by a GAN. SensoryGANs [59] is a generative adversarial network framework that generates sensor data for human activity recognition. Its authors designed specific GAN models for three human daily activities: stay, walk and jog. They also proposed three visual evaluation methods for assessing the performance of SensoryGANs. Their experimental results showed that SensoryGANs models have the capability to capture the implicit distribution of human activity's sensor data, and the synthetic sensor data generated by SensoryGANs improved human activity recognition. However, this work does not consider the sequence of different activities. Rather, it models each type of activity separately.

The authors in [43] proposed a supervised GAN architecture that learns from feedback of both a discriminator and a classifier agents in order to create synthetic labeled sensor data. This demonstrates the effectiveness of the architecture on a publicly available human activity dataset. Moshiri et al. [40] generated data by using 50% of their raw data in conjunction with a GAN to train an LSTM network for detecting human activities. Their experimental results confirm that using GAN-generated data can improve classification accuracy. Esteban et al. [16] proposed a Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN) to produce real-valued multi-dimensional time series, with an emphasis on their application to medical data. The authors demonstrated that they can successfully generate realistic time-series. The approach they used was to generate a synthetic labelled training dataset and evaluate the performance of a model trained on the synthetic dataset using a real test set, and vice-versa. This illustrated that RCGANs-generated time-series data was useful for supervised training, with only slight performance degradation compared to real test data. This was demonstrated by training an early warning system on a medical dataset of 17,000 patients recorded from an intensive care unit.

In this research, we propose BehavGAN to generate a synthetic behavior dataset based on a real dataset. We build on the original SeqGAN algorithm and incorporate n-gram based metrics in the reward function to address the issue of overtraining and identical record generation in SeqGAN, which turns out to be a challenge when it comes to generating sequences consisting of tokens from a limited token space.

3 BACKGROUND

In this section we discuss the scientific background related to our research.

3.1 Generative Adversarial Networks

GANs [20] are composed of two neural networks: a generator network, and a discriminator network, that are playing a minimax game. The generator network initially generates samples from random noise. The discriminator then continuously learns to distinguish between generated samples and the real data which are both fed into a supervised model. The generator on the contrary receives feedback on the generated sample from the discriminator and tries to generate samples similar to the real data so that the discriminator cannot distinguish generated samples from real data. In other words, discriminator D is trained to maximize the probability of assigning the correct label to both real data samples and generated samples while generator G is simultaneously trained to make it more difficult for the discriminator to distinguish real data from generated data. Thus, D and G play the following two-player minimax game with value function $V(D, G)$:

$$\begin{aligned} \underset{G}{min}G \underset{D}{max}\, V(D,G) =& \mathbf{E}_{Y \approx p_{data}(Y)}[logD(Y)] \\ &+ \mathbf{E}_{z \approx p_z(z)}[log(1 - D(G(z)))]. \end{aligned} \quad (1)$$

where p_{data} is the real data distribution and $p_z(z)$ is input noise used to learn p_g. The value function V is defined so as to maximize the discriminator's error by minimizing the generator's error. According to the above formula, a good generator generates samples similar to real data so that $D(G(z))$ would be close to 1 and $D(Y)$ would be close to 0 leading to maximizing $log(D(Y) + log(1 - D(G(z)))$.

Although several versions of generative adversarial networks such as conditional GANs, DCGAN and InfoGAN [9, 39, 47] have been presented and successfully used for generating verisimilar images, generating sequences of discrete tokens has not received much study. SeqGAN is an effort to close this gap by providing an algorithm that leverages reinforcement learning to calculate a reward based on the discriminator's judgement on complete generated sequences. The authors have also utilized Monte Carlo search to calculate the reward for partial sequences using rollout mechanisms. They have tested the efficiency of their proposed algorithm using text and music datasets [63]. LeakGAN [23] is also an effort to address the issue of long text generation. The authors propose to allow the generator receive leaked information on discriminator's high-level features and incorporate such signals into generation steps. In this research, we propose a model-free behavior sequence generator engine by extending the original SeqGAN method. To the best of our knowledge, this is the first time that GANs have been applied to behavior sequence generation.

In SeqGAN the discriminator reward, which is backpropagated to the generator, is formulated as:

$$R^{G_\theta}_{D_\phi}(a = y_T, s = y_{1:T-1}) = D_\phi(y_{1:T}). \tag{2}$$

In this formula, $D_\phi(y_{1:T})$ is the discriminator's judgement on a complete sequence. This stands for the discriminator's estimate of the probability that the sequence is real. It is then backpropagated to the generator as the reward in reinforcement. We refer you to [63] for further details of SeqGAN.

3.2 Long Short-Term Memory Networks

We employ GANs to generate sequences of daily activities. In terms of the architecture of our model, we need to build a generator model capable of generating sequences of discrete tokens (activity type such as sleeping or eating) and continuous tokens of activity duration. Recurrent Neural Networks (RNNs) tend to suffer from vanishing and exploding gradient problems during training. A long short-term memory (LSTM) recurrent unit is introduced to reliably capture long-term dependencies [26]. Regarding the temporal nature of human behavior, we employ LSTM networks which deal with vanishing and exploding gradient problems by employing forget gates. The application of GANs in temporal sequences has been partially studied [16, 64], but generating discrete sequences is a challenge that needs further research [63].

3.3 Maximum Likelihood Estimation

MLE (Maximum Likelihood Estimation) is a method for determining the values of a model's parameters. The parameter values are chosen to maximize the probability that the model's described process produced the data that were actually observed. MLE aims to maximize the log-likelihood of ground-truth sequences when it comes to sequence generation [52].

3.4 Monte Carlo Tree Search

Monte Carlo Search is a method which is usually used in games to predict the path (moves) that should be taken by the policy to reach the final desired outcome. Brute force searching of an exponentially expanding tree to find the best path is rather infeasible. Monte Carlo tree search explores the best move out of a set of moves by: selecting the best node in the tree → expanding the selected node → simulating the exploration → back-propagating the updated scores. This basic procedure can be applied to any game whose states necessarily have a finite number of moves and finite length. For each state, all feasible moves are determined, N random games are played out to the very end, the scores are recorded and finally the move leading to the best score is chosen [19]. In the original SeqGAN, Monte Carlo search is used for calculating expected reward associated with partial sequences which are generated by the model. To this end, a generated partial sequence is considered as the state and computed reward is the score which is required to be maximized. At each state, the generator needs to decide what action to take (generating next token). In fact, Monte Carlo search allows the generator to have a confident estimation of the long-term reward associated with taking each action considering the current state.

3.5 N-gram based policy gradient

The concept of N-gram has been widely used in the NLP literature, especially when it comes to evaluating language models. The main idea behind this concept is that by counting the number of common N-grams between a sequence and a ground-truth sequence we can evaluate the similarity of the two sequences. Therefore, N-gram based metrics such as BLEU (Bilingual Evaluation Understudy) score [45] can also be used to train the neural network by guiding its policy. That being said, RL policy-gradient algorithms [56] can optimize BLEU. Then, the minimization objective can be formulated as:

$$J(\theta) = \mathbb{E}[R_t | s_0, G_\theta] \tag{3}$$

where R_t is the expected value of the BLEU score given the prefix s_0 and the generation policy G_θ to follow. One major issue with this algorithm is that BLEU is not a computationally cheap metric. In this paper, we present a solution for implementing BLEU score calculations using a hash table data structure to make it feasible to calculate the BLEU score in the adversarial training loops of the algorithm. In Section 4 we explain how our algorithm reinforces the generator policy in a direction that does not sacrifice its diversity for similarity. We backpropagate the discriminator's rewards combined with the BLEU score to maintain the similarity of generated sequences while preventing the generator network from repeatedly generating real data sequences.

4 APPROACH

In this section, we introduce our approach for synthesizing human behavior sequences in a limited environment such as a home. First, we present a thorough explanation on how we represent behavior. Then, we introduce our solution with an emphasis on how it addresses some issues associated with synthetic behavior generation.

4.1 Behavior representation

We need to model human indoor behavior for relatively unconstrained environments. In our model, we consider the start time of ADLs as the baseline for the order of tokens in sequences. In the case of interleaved ADLs, ADLs will be put in the sequence according to their start time. To handle the simulation of concurrent activities, we divide the duration of interleaved activities into the shared execution time and the time that they are performed separately. We define specific tokens for the concurrent activities (for example, eatrelax for the shared execution time that the resident has been eating food while relaxing on the couch). There are two different situations that interleaved ADLs can occur: 1) ADL1 starts, ADL2 starts, ADL1 ends, ADL2 ends 2) ADL1 starts, ADL2 starts, ADL2 ends, ADL1 ends. So, we define a special token (ADL12) for the concurrent activities. Using the combined token, the sequence of activities can be represented as: 1) ADL1, ADL12, ADL2, and 2) ADL1, ADL12, ADL1, subsequently. This approach can be expanded to handle concurrency of more than two activities. However, given that we are dealing with humans, many concurrent tasks are unlikely.

Considering behavior as a sequence of discrete tokens (sleeping, eating, watching TV and preparing meals to name a few), two important quantities emerge: i) *Content*: activities that constitute a behavior; and ii) *Order*: the temporal arrangement of the constituent activities. The idea of tokenizing behavior in our work is similar to the way researchers in Natural Language Processing (NLP) have looked at documents as vectors of their constituent words (see Vector Space Model, VSM [51]). Approaches such as VSM capture the content of a sequence in an efficient way. However, they completely ignore its order. Behavior is not fully defined by its activity-content alone; rather, by its natural activity-orderings. Therefore, a model to capture activity order in an explicit manner is needed. To this end we consider a sliding window of size W over a behavior sequence to take into account all possible sequences of length T.

Due to the fact that the behavior sequence will be fed into an LSTM generator, we have to consider a fixed length for the input behavior sequence. However, behavior sequences can be of any length as people perform different number of ADLs each day. We propose two approaches for tackling this issue: a) Sliding window (with a shift delta 1) allows for sliding over the dynamic-length sequences and capturing ADL dependencies. In this approach, although the length of sequences is fixed to a predefined value (sliding window length), truncating the sequences does not harm capturing ADL dependencies as the dependency between the token at the truncating point and its post- or pre- tokens will be observed in the previous and next sequences, respectively, when the window slides over the original sequence. The sliding window size needs to be determined depending on the type of analysis that the generated data will be used. If data is to be used for learning short patterns, it makes sense to have a small sliding window. b) as a solution, several approaches employ sequence padding and truncation [21, 24, 35, 44, 54]. This entails determining a single length for all sequences, then truncating longer sequences to that length or filling shorter sequences with a "fake" character until they reach that length. Padding is the process of filling in gaps in a sequence with a character that isn't present in the sequence. Padding tokens can be inserted at any point in the sequence. However, in practice they are often added to the end of sequence [34].

Both approaches have pros and cons. While the first approach is efficient in generating partial sequences, the sliding window length parameter can affect the effectiveness of the model. Setting a small fixed value for the sliding window length can result in losing long-term ADL dependencies. For example, setting the window size to X would prevent the model from learning the dependencies that can exist between each token and the tokens that are more than X tokens apart. Also, setting it too high will restrict the model from creating short sequences and make it difficult for the model to learn and generate sequences that are similar to the original data.

On the other hand, padding approach allows the model to process variable-length sequences as input and output. However, padding can lead to a decrease in the efficiency (accuracy) of the model as a result of adding padding tokens to the original sequence. We propose to consider each sequence as the ADL sequence of a day. For identifying the maximum sequence length, we investigate the original data to find the longest daily sequence. With that information, a large enough value for Maximum Sequence Length (MSL) will be defined. Then, sequences with shorter length than MSL will be padded.

To determine an appropriate value for T, we need to find a small-enough number that, while it limits model complexity, is a good length to cover a representative sequence of the individual's patterns of behavior. In this paper, we model human behavior B as an ordered sequence of events:

$$B = e_1, e_2, ..., e_i, ..., e_W \tag{4}$$

where e_i refers to an event. We define event e_i as a pair of activity a_i and duration d_i:

$$\begin{aligned} e_i = (a_i, d_i); where\ \ a_i \in \{activity\ \ types\}\ and \\ d_i \in \{activity\ \ duration\ \ range\} \end{aligned} \tag{5}$$

(Sleeping,400),(Toilet,6),(Meal_Prep,11),(Eating,14),(Relax,16),(Leave_Home,21)

Sliding window of size 4

1. Map each activity type to a distinct digit (10,11,12,…,20)
   (10,400), (11,6), (12,11), (13,14), (14,16), (15,21)
2. Normalize activity duration to be in range 1 to 5
   (10, 3.9), (11, 2.1), (12, 2.6), (13, 2.3), (14, 1.8), (15, 1.3)
   *Note that we normalize each type of activity, separately
3. Discretize normalized duration
   (10,3),(11,2),(12,2),(13,2),(14,1),(15,1)
4. Differentiate between activity type durations of each activity type
   activity type concatenated with duration >>> duration : '10'+'3' >>> 103
   The input tensor: (10, 103), (11, 112), (12, 122), (13, 132), (14, 141), (15, 151)

Fig. 2. An example of behavior encoding to show the process of preprocessing raw data to turn it into input tensors

Then, we reshape B to a flat tensor B' in order to feed it into the algorithm:

$$\begin{aligned} B' &= y_1, y_2, ..., y_k, ..., y_T; \\ &where \;\; y_k = a_i \; if \; k \; is \; odd \; and \\ &y_k = d_i \; if \; k \; is \; even \\ &s.t. \quad i = [\frac{k+1}{2}] \end{aligned} \tag{6}$$

where T is the window size and equals $2 \times W$. It is worth mentioning that activity type is categorical data which needs to be encoded in integers so it can be fed into the LSTM model. For activity duration, we also discretize the values so the model deals with categorical values. We believe that, while it does not hurt the accuracy of the model, it simplifies the model by decreasing the state space. As illustrated in Figure 2, first, we normalize the duration of each activity type separately as the range of duration in different activity types varies. Then, an equal width discretization method is applied to turn the duration values into categorized values.

4.2 Identical sample generation issue

Generative adversarial networks (GANs) can be difficult to train when it comes to generating sequences consisting of tokens from a limited token space. The issue lies in the fact that the GAN model is trained to generate samples similar to the real data. Thus it is probable that the model will repeat itself and generate records that are identical to the real data. An intuition behind why identical sample generation issue occurs is that the discriminator's output is the only information that is provided to the generator. Therefore, if the discriminator identifies that a generated sample is very similar to the real data, it passes high rewards to the generator and the generator continues to generate from that pattern repeatedly. This issue becomes more severe when it comes to generating data from a limited token space, including behavior sequence generation. In behavior sequences, token space is limited to the activities that an individual can realistically do, which is likely to have less variety than would be seen, for example, in language space or image space.

To address this issue, we introduce a combined reward method that incorporates the BLEU score in the reinforcement mechanism. According to SeqGAN, $R^{G_\theta}_{D_\phi}$ is an action-value function of a sequence, that calculates the expected accumulative reward starting from state s, taking action a, and following policy G_θ. The discriminator's reward is calculated both for complete and partial sequences as follows:

$$R^{G_\theta}_{D_\phi}(a = y_t, s = B'_{1:t-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi({B'_{1:t}}^n), {B'_{1:t}}^n \in MC^{G_\beta}(B'_{1:t}; N) & for\ t < T \\ D_\phi(B'_{1:t}) & for\ t = T \end{cases} \tag{7}$$

where $D_\phi(B'_{1:T})$ is the discriminator's output for a complete sequence (when $t = T$), $B'_{1:T}$, indicates the probability that the sequence is from real sequence data or not. For partial sequences (when $t < T$), N samples of complete sequences (${B'_{1:t}}^n$) that are sequels to the partial sequence will be selected from Monte Carlo tree to be used for estimating the ultimate reward associated with a partial sequence $D_\phi({B'_{1:t}}^n)$. As shown in the above formula, the discriminator reward is calculated after the generation of each token (activity $a = y_t$) with a current state of s.

Since we want to guide the generator in a direction that avoids generating completely identical sequences as the real data and moreover generates a diverse variety of sequences, we need to evaluate it in terms of diversity. The BLEU score gives us a sense of how similar the generated sequence is to the reference set (the real data). Then, we can conclude that samples with a very high BLEU score are likely to trap the model into the issue of identical sample generation. Therefore, we define a new action-value function based on the BLEU score as:

$$R^{G_\theta}_{b}(a = y_t, s = B'_{1:t-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} R_b({B'_{1:t}}^n), {B'_{1:t}}^n \in MC^{G_\beta}(B'_{1:t}; N) & for\ t < T \\ R_b(B'_{1:t}) & for\ t = T \end{cases} \tag{8}$$

where $R_b(B'_{1:T})$ is the BLEU score associated with a complete sequence $B'_{1:T}$ which indicates the similarity of $B'_{1:T}$ to the reference data. Now that the model has a sense of the diversity of the generated sample, we define a combined reward that is a function of both R and R_b:

$$R_{comb} = f(R, R_b) = \begin{cases} max(R) - R & if\ R_b > Threshold \\ R & otherwise \end{cases} \tag{9}$$

where max(R) equals $max(R^{G_\theta}_{D_\phi}(a = y_t, s = B'_{1:t-1}) : y_t \in \gamma)$ that is the maximum discriminator reward calculated for N generated sequences in every rollout. γ is the vocabulary of candidate tokens. R and R_b also refer to the discriminator's reward and the BLEU reward defined respectively in equations 7 and 8. An overall picture describing the adversarial learning mechanism in our proposed solution is presented in Figure 3.

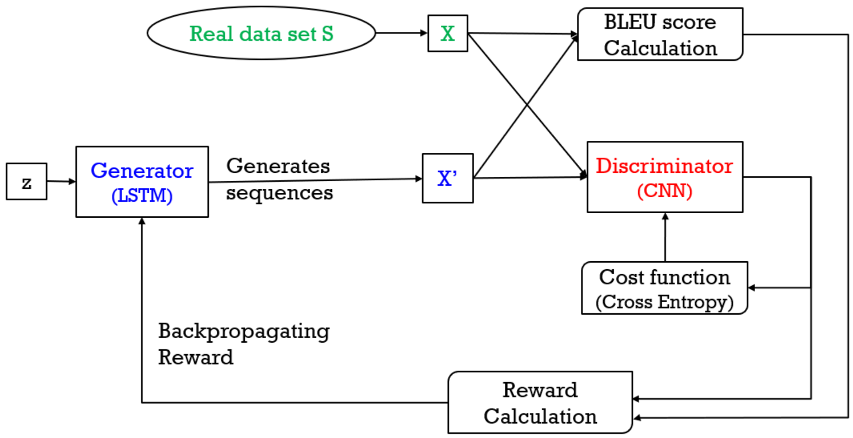

Fig. 3. Overall architecture of BehavGAN describing the proposed adversarial learning mechanism. The generator network starts generating batches of sequences from a normal distribution. Then, generated sequences with the label "fake" along with real sequences with the label "real" are fed into the discriminator network to distinguish real data from fake data. The Discriminator network keeps training based on Cross-Entropy loss. A reward for the generated sequences will be calculated using the BLEU score based on Equation 9. This reward is backpropagated to the generator to guide its learning by reinforcing quality yet diverse sequences.

4.3 Computational cost issues with BLEU score calculation

As specified by Papineni Papineni et al. [45], the BLEU score is a modified n-gram measure of precision of a hypothesis, given a set of references R. “Modified precision” is the maximum number of occurrences for each n-gram of a hypothesis in the reference set, with an upper bound of the number of occurrences for that n-gram in the hypothesis. The geometric mean is calculated over the precisions for all values of n, and multiplied by a brevity penalty which is 1.0 if the hypothesis sentence is of the same or smaller length than the reference sequence, and less than 1.0 otherwise. Thus, a BLEU score of 1.0 means that for all n-grams in the hypothesis, there is at least one sequence in the reference set in which the number of n-gram occurrences is equal to or greater than that of the hypothesis sequence. Its length is also less than or equal to the length of the hypothesis sequence. Assuming that we fix the length of generated sequences to be equal to the length of the reference sequences, brevity penalty would be 1. Usually, n is set to 4 and we denote this metric as BLEU-4 which can measure the similarity between sequences by counting unigrams, bigrams, trigrams and 4-grams. The larger the value of n, the smaller is the BLEU score. In this paper, we apply BLEU-4 as a similarity metric to be calculated in every training loops of the generator.

As demonstrated in Algorithm 1, in every training epoch our algorithm, BehavGAN, calculates the BLEU score for a batch of generated samples. In other words, in each epoch k, the generator generates a sequence which will then be used to calculate the BLEU scores associated with the complete sequence and all the possible partial sequences. The method estimates the BLEU score for partial sequences by applying the rollout mechanism. As explained above, in the rollout mechanism a sample of size N is picked to estimate the BLEU score. Considering the fact that the BLEU score calculation is time-consuming, we need to resolve this issue as this calculation will be performed $N \times T \times K$ times; where N is the rollout number, T is the sequence length and K is the number of epochs.

Therefore, each time we compute the BLEU score for a hypothesis sequence, we compare its n-grams with those of the reference sequences. Since the reference sequences are not changing, we can count the n-grams in the reference

Algorithm 1: Behavior GAN (BehavGAN).)

Input: Generator Policy: G_θ; Roll-out Policy G_β; Discriminator Policy D_ϕ;
Real Sequence Dataset (Positive Samples) $S = X_{1:T}$
Output: Synthetic Sequence Data (Negative Samples)

```
Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on S
β ← θ
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ using negative and positive(S) samples via minimizing cross entropy
repeat

  for g-steps do
    Generate a sequence B'_{1:T} = (y_1, ..., y_T) ≈ G_θ
    for t in 1 : T do
      Compute R^{G_θ}_{D_ϕ}(a = y_t, s = B'_{1:t-1}) by Eq.7
      Compute R^{G_θ}_b(a = y_t, s = B'_{1:t-1}) by Eq.8
      Compute R_comb by Eq.9
    end for
    Update generator parameters via policy gradient
  end for
  for d-steps do
    Use Current G_θ to generate negative samples and combine with given positive samples S
    Train discriminator D_ϕ for k epochs
  end for
  β ← θ
until BehavGAN converges
```

sequences only once and utilize that number each time we need to calculate the BLEU score for a new hypothesis, instead of counting the n-grams for every candidate calculation. To optimize the BLEU score calculation we use a hash table data structure. In our implementation, we use the python dictionary data structure where the *key* is the "n-gram" and its associated *value* is the maximum number of occurrences of the corresponding n-gram in a reference sequence. This way, a huge number of calculations are pre-calculated once and the resulting constant values are easily accessible. This implementation makes our algorithm capable of providing feedbacks based on the BLEU metric in a timely manner.

5 CASE STUDY

In this section, we discuss our experimentation to generate synthetic dataset based on a real dataset. We introduce the real dataset we used, the proposed combined reward evaluation, and discuss the evaluation process which illustrates the improvement to the quality of generated data compared to those in baseline methods such as MLE, LeakGAN and SeqGAN.

Table 1 presents the specification of the high performance computing platform, i.e., the Nvidia's DGX-1 HPC server, that we used for computations in this research project.

Table 1. NVIDIA high performance computing platform used in this research.

HPC Server	
GPU Architecture	NVIDIA Volta
GPU Product	NVIDIA Tesla V100
Driver Version	418.126.02
CUDA Version	10.1
GPU Memory	16 GB HBM2
Memory Bandwidth	900 GB/sec
System Memory	251 GiB
Operating System	
OS Version	Ubuntu 18.04.4
Software	
Programming Language	Python 3.6.9
Libraries	NVIDIA Release 20.01-tf2

Table 2. Example data from CASAS-Aruba dataset.

Date	Time	SensorID	SensorStat	Activity
2010-11-04	00:03:50.20	M003	ON	Sleeping begin
2010-11-04	00:03:57.39	M003	OFF	
2010-11-04	00:15:08.98	T002	21.5	
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
2010-11-04	05:40:43.64	M003	OFF	Sleeping end
2010-11-04	05:40:51.30	M004	ON	
2010-11-04	05:40:52.34	M005	OFF	BedToToilet begin
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
2010-11-04	05:43:30.279	M004	OFF	BedToToilet end

5.1 Real dataset

To develop a GAN for generating synthetic yet realistic dataset, we chose two real daily activity datasets as ground-truth data to test the effectiveness of BehavGAN. These were: (i) the CASAS-Aruba dataset [10] which consists of activities that a woman performed in a home during a period of seven months; and (ii) Kastaren dataset [58] consisting of 28 days of sensor data with annotation of activities.

The CASAS-Aruba dataset is limited to activities performed by a single person. A few examples from this dataset are shown in Table 2. In this dataset, eleven types of indoor activities were included. MealPreparation, Relax, Eating, Work, Sleeping, WashDishes, BedtoToilet, EnterHome, LeaveHome, Housekeeping and Resperate were recorded using motion sensors, door sensors and temperature sensors. As shown in Table 2, start and end times for each activity were recorded, making it possible to calculate the duration of the activity. Also, the time ordering of activities was captured. In Kastaren dataset seven different activities are annotated, namely: Leave house, Toileting, Showering, Sleeping, Preparing breakfast, Preparing dinner and Preparing a beverage. Table 3 presents some overall statistics on these datasets.

Table 3. Statistics from different datasets.

Dataset	Number of Records	Number of Activities
CASAS-Aruba	1,719,558	6,468
Kastaren	2,120	245

5.2 Results

In this section, we present the results of applying BehavGAN on the CASAS-Aruba and Kastaren datasets, separately. As discussed in Section 4, we first encoded the dataset records and defined a sliding window of size 10 (BehavGAN), or padded sequences until length 20 is reached (BehavGAN_padded) from which behavior tensors were calculated for the model. We chose 10 and 20 for the sequence lengths by analyzing the real data sequences. It turned out that most days have less than 10 ADLs, which suggests defining the sliding window size of 10. Also, the maximum length of daily sequences in the real data is 16, which is why we set the padded sequence length to 20 to allow for generating marginally longer sequences. These tensors are represented in Algorithm 1 as members of the S set. The Algorithm then generates negative samples via the generator network and eventually outputs a final generated dataset. Table 4 illustrates the parameters we set for running the BehavGAN algorithm on CASAS-Aruba and Kastaren datasets. We ran the model for the two datasets, separately.

We employed the same architecture for both the generator and the discriminator networks as in the original SeqGAN study. The Tanh activation function is used in the generator's LSTM network. The hidden states are then mapped into the output token distribution via a Softmax output layer. For pre-training, the generator implements Negative Log-Likelihood Loss (MLE pre-training steps). The generator seeks to maximize the reward as well as the discriminator's loss. The discriminator network outputs the likelihood that a given sequence is real using a fully connected Sigmoid layer. Before the final fully connected layer, it adds a highway layer and a dropout layer (0.75). The discriminator uses Cross Entropy loss. All parameters are randomly initialized. Both networks use Adam optimizer. For calculating the BLEU reward ($R_b^{G_\theta}$) we used BLEU-4 as it is usually used for evaluating the similarity of an hypothesis sequence to a reference set.

We also ran SeqGAN and LeakGAN algorithms [23, 63] with real data as input. For the sake of comparison we implemented a MLE model to generate synthetic data, which aims to maximize the log-likelihood of ground-truth sequences. Simply put, it is trained to predict the next token based on the ground-truth tokens that have come before it. This method was also used in the pre-training of SeqGAN and our proposed algorithm, but in this case we did not use it for pre-training but for the training process. Figure 4 shows the distribution of BLEU-4 scores for CASAS-Aruba data as well as generated data using MLE, SeqGAN, LeakGAN, BehavGAN, and BehavGAN_padded (BehavGAN with padding). The purpose of this comparison is to examine if the similarity of the generated data to real data is comparable to what happens in real data. We want it to resemble what happens with real-world data. To compare the distribution of synthetic data with that of real data we calculate BLEU-4 score for each dataset. To calculate BLEU scores for synthetic datasets, we consider the real data as the reference set while the generated data with each model is considered as the candidate set. To calculate BLEU scores for the real data (CASAS), we partitioned the real data into two separate ordered subsets. The first half goes to the candidate set and the second half goes to the reference set. By comparing all candidate sequences with the reference set we calculate the similarity of the first half to the second half.

As shown in this figure, the generated sequences using SeqGAN, LeakGAN and MLE are very similar to the reference set (CASAS data). The issue with data distribution of three baseline methods is that Using baseline models, a major

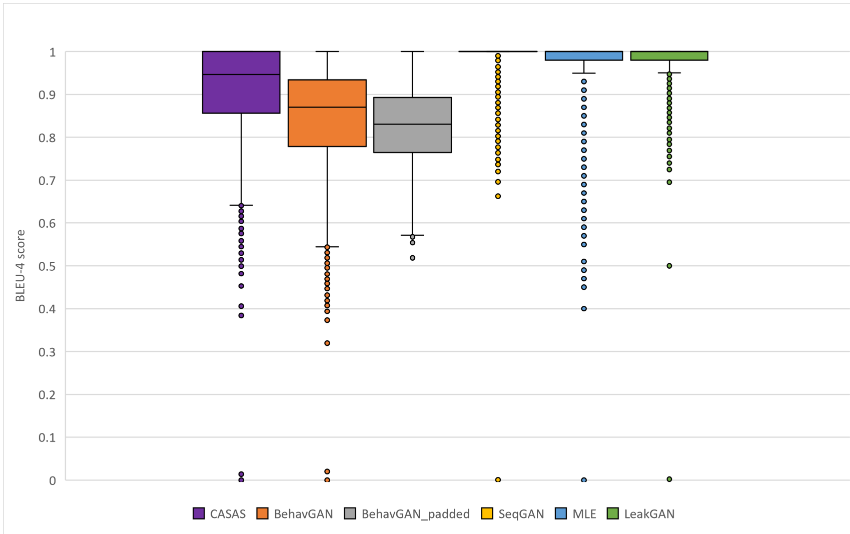

Fig. 4. Comparison of BLEU-4 Score Distribution for CASAS-Aruba Data with Synthetic Data Generated with MLE, LeakGAN, SeqGAN, BehavGAN, and BehavGAN_padded (BehavGAN with padding) using box plots. A large portion of the generated data using baseline models is too similar to the original data (i.e., Q1, Q2(Median), Q3, and Maximum are all too close), seriously affecting the diversity of the generated data. In comparison to MLE, LeakGAN, and SeqGAN, our technique reduces the proportion of generated sequences that are excessively similar to the real data (BLEU-4 score near 1). An important point to note here is that our proposed algorithm (with and without padding) can generate sequences that are comparable to the real dataset (median is still around 0.85) while avoiding the generation of a significant number of identical records (Q3 is slightly higher than 0.9).
* The circles on the box plots represent outliers.

Table 4. Run parameters

Parameter	**CASAS-Aruba**	**Kastaren**
No of Generated Records	10,000	1,000
BL_Threshold	0.85	0.8
Sequence Length[1]	10, 20	10, 20
Pre-training Epochs	50	250
Training Epochs	150	200
Generator's Learning Rate	0.08	0.03

amount of the generated data is too similar to the original data (i.e., Q1, Q2(Median), Q3, and Maximum are all too similar), reducing the diversity of the generated data. Our method, as compared to MLE, LeakGAN, and SeqGAN, decreases the proportion of generated sequences that are overly close to the real data (BLEU-4 score near 1). It is important to note that our proposed algorithm can produce sequences that are comparable to the real dataset (median remains around 0.85) while avoiding the generation of a large number of identical records (Q3 is in the neighborhood of 0.9). This feature of our BehavGAN makes it a better solution for generating synthetic data. Furthermore, the results suggest that employing padded sequences in the model has no considerable impact on the similarity and diversity of generated sequences. Setting a longer maximum length (20) for generated sequences in BehavGAN_padded may explain why BLEU-4 scores are marginally lower.

Table 5. Comparison of similarity and diversity metrics on CASAS-Aruba and Kataren datasets.

Data set	**Algorithm** AVG - VAR	**BLEU-4**	**Identical Record Ratio**
CASAS-Aruba	MLE	97.2% - 0.003	14.2%
	LeakGAN	94.0% - 0.04	49.8%
	SeqGAN	99.4% - 0.00006	47.6%
	BehavGAN	**89.3% - 0.020**	**8%**
	BehavGAN_padded	83.6% - 0.011	8.1%
	Real data	90.8% - 0.013	6%
Kastaren	MLE	91.2% - 0.0025	17.3%
	LeakGAN	95.1% - 0.06	61.2%
	SeqGAN	92.9% - 0.033	56.3%
	BehavGAN	**87.4% - 0.024**	**12%**
	BehavGAN_padded	82.4% - 0.041	7.6%
	Real data	88.5% - 0.027	8.4%

Table 6. BLEU score calculation speed

	With hash table	**Without hash table**
Run Time (150 Epochs)	145 mins	4,100 mins

Table 5 presents comparison metrics in terms of similarity (BLEU-4 score average and variance) and diversity (identical records proportion) for experiments on CASAS and Kastaren datasets. In this table, real data is used as the baseline for comparison. We provide the average BLEU-4 score for the real data, MLE-, LeakGAN-, SeqGAN-, BehavGAN, and BehavGAN_padded-generated data to illustrate that our proposed algorithm is capable of generating a synthetic dataset with a high similarity to the real data. For calculating the BLEU score we consider the real data and the generated data as the reference set and the candidate set, respectively. Moreover, according to this table our proposed reward method improves the output dataset by decreasing identical sequences while maintaining an acceptable similarity rate. We ran each model for five times and report the average value for each reported item. In order to further analyze the ability of BehavGAN in generating interleaved activities, we compute the BLEU-4 score to measure the similarity of generated sequences that include concurrent activities with the reference set. The results (an Average BLEU-4 score of 0.86 with a Variance of 0.08) indicate that generated sequences that include concurrent activities still have high similarity to the reference data. We also investigate the diversity of these sequences by calculating the Identical Record Ratio. Only 14 percent of these sequences are identical to the reference sequences.

Table 6 shows how the speed of BLEU score calculation is enhanced by implementing a hash table structure. In this table, the run time of our algorithm (for 150 epochs) with and without the enhancement solution is compared. The runtime of the original SeqGAN algorithm for the same number of epochs and parameters is slightly lower, i.e. 130 mins, which is not a significant difference considering the fact that our proposed algorithm outputs higher quality data.

[1]Sequence Length for all models is set to 10, except for BehavGAN_padded, which has Sequence Length of 20

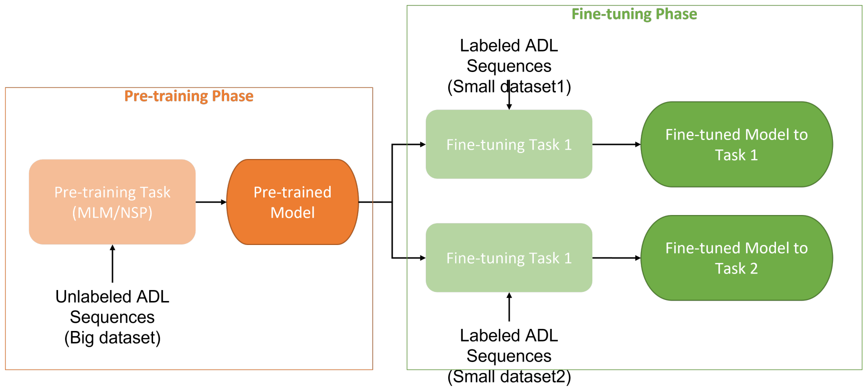

Fig. 5. Two-Phase training in BERT models using unlabeled and labeled data.

5.3 Effectiveness of BehavGAN

In this section, we perform an experiment to evaluate the effectiveness of BehavGAN in synthesizing behaviour sequences. This experiment is designed to demonstrate the effectiveness of generated data in machine learning tasks. We describe the task and the results of training the model using data augmented by synthesized data vs training the model with only real data.

Bidirectional Encoder Representations for Transformers (BERT) are standard building blocks for training task-specific Natural Language Processing (NLP) models [13]. When fine-tuned utilizing domain-specific labeled data, pre-trained BERT models have been shown to be effective, cost-effective, and time-efficient in addressing downstream tasks [21]. This is greatly beneficial since models are pre-trained using general unlabeled data, where labeling is a costly and time-consuming task and little labeled data is available. Subsequently, they can be fine-tuned to a particular supervised task, such as sentiment classification, with a rather small, labeled dataset as illustrated in Figure 5.

The input to a BERT model is text/sequence spans, such as sentences divided by special tokens [SEP]. Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) are two tasks used to pre-train the BERT model with unlabeled data to capture the inter-dependencies between words and sentences. BERT can extract several contextual and structural features during pre-training if adequate training data is provided.

Masked Language Modeling (MLM) is the process of masking tokens in a sequence with an arbitrary probability of 15% - 20% with a masking token, [MASK], and instructing the model to fill (predict) that mask with an appropriate token. The goal of the training is to reduce the cross-entropy loss between the original masked tokens and the predicted ones as much as possible. This allows the model to focus on the right (tokens on the right side of the mask) and left (tokens on the left side of the mask) contexts at the same time. Models may learn textual patterns from unlabeled data via MLM, which is employed in pre-training tasks. NSP is used to pre-train the model by having it anticipate the sentence that comes after each one in the training corpus.

In this section, we demonstrate the effectiveness of BehavGAN, our proposed synthetic behavior sequence generation approach, by presenting the results of a MLM task that is trained on both original and synthetic data generated by BehavGAN, as well as synthetic data generated by three baseline methods. For each method, we train separate masked models with training data, which includes 90% of real data augmented by the generated data with the corresponding method. The trained model is then evaluated on test data, which is 10% of the real data. Each sequence is treated as an

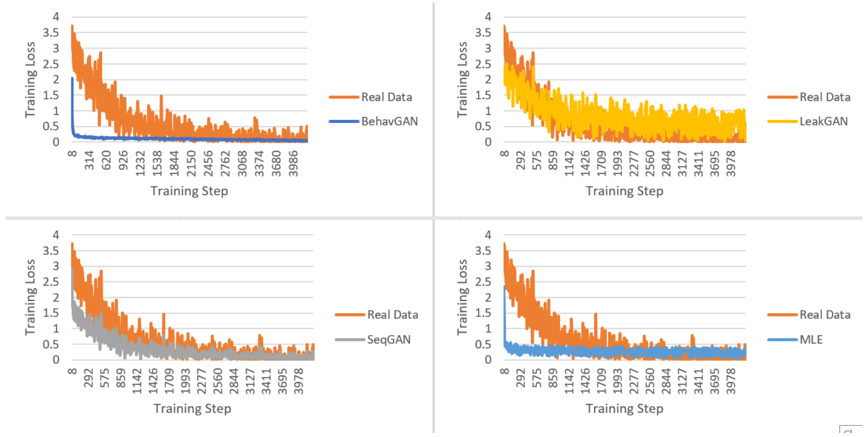

Fig. 6. A Comparison of Training Loss for the Masked Language Modeling Task using Real Data, MLE-, LeakGAN-, SeqGAN-, and BehavGAN-generated Data.

Table 7. The Evaluation Loss of the MLM task - CASAS dataset

Algorithm	**Cross-Entropy Loss**
MLE	0.59
LeakGAN	0.63
SeqGAN	0.62
BehavGAN	**0.53**
Real data	0.64

input to the MLM task. During the training process, 15% of the tokens will be chosen at random and masked. The model is trained to predict masked tokens throughout the training process as shown in Figure 7. As a result, the model's low evaluation loss suggests that it has acquired contextual and structural features of the data, making it suitable for use as a pre-trained model for tasks including abnormality detection, next activity prediction, etc. MLM has also been directly used to solve problems like System Log Anomaly Detection [30] and Text Denoising [53].

For this experiment, we used BERT-base-uncased from Hugging Face library with six attention heads. We run the baseline methods as well as the BehavGAN to generate 10,000 records. Then, we combine synthetic data with 90% of real data. Now, we run the MLM task with each training set. In the evaluation step, the aforementioned trained models will be evaluated using the evaluation set (10 percent of the real data from CASAS). In Figure 6, we present the evolution of the masked model during the training steps. As illustrated in this figure, training loss decreases throughout the training process. However, training the model with BehavGAN results in the most consistent and rapid reduction in training loss.

Table 7 presents the evaluation loss of the MLM experiment, where the number in front of each method shows the evaluation loss of the MLM on the data generated by that method. Also, the number in front of Real data shows the evaluation loss of the MLM on the original CASAS data. We can deduce from these findings that BehavGan has effectively increased the model's accuracy in predicting masked tokens (0.11 decrease in the Cross-Entropy Loss of the model). Other methods do not show noticeable improvement in training the model.

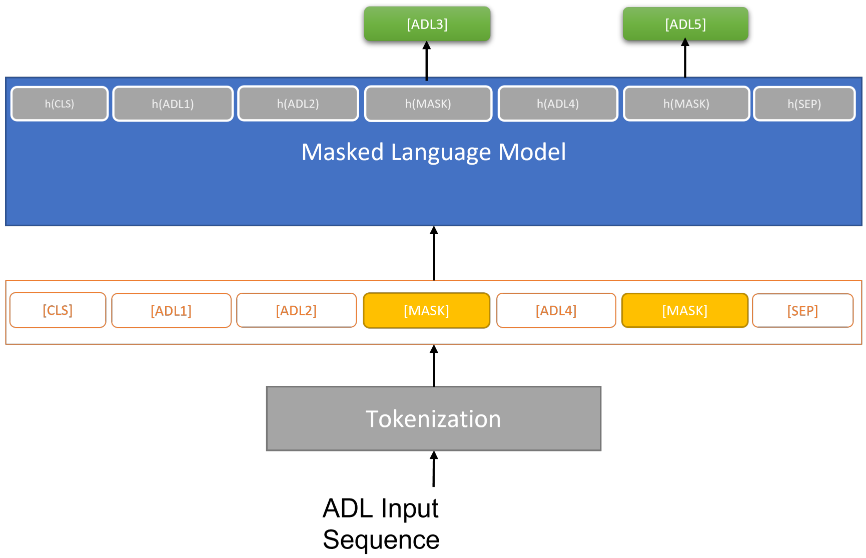

Fig. 7. An illustration of the Masked Language Modeling task for predicting MASK tokens in ADL sequences.

5.4 Human Evaluation

In addition to evaluating the similarity and diversity of generated data using the BLEU score metric, human evaluation was conducted in order to make sure the generated behavior sequences made sense in terms of the order and the duration of activities. To that purpose, we enlisted the help of three raters, who were given 50 sample sequences to rate on a scale of 1 to 5, with the higher the number, the more likely the sequence is perceived to be real. Two raters are Ph.D. students in the field of health management, and one is a Ph.D. student in the field of management sciences. Prior to collecting their judges, we double-checked that they fully comprehended the task. To ensure that they only use their common sense to determine if the presented sequence is real or fake, we anonymized the data and released no detail about the typical ADL patterns in the data. The sample data includes 10 random samples from real data, random samples generated by the baseline methods, 10 samples each, and 10 random samples generated by our proposed method. We excluded generated sequences that are identical to the real data since the BLEU score evaluation revealed that a large proportion of generated sequences by baseline methods was identical to the real dataset. This is not desired when it comes to synthesizing data. Instead, we want the method to be able to generate valid yet diverse sequences. Table 8 presents the result of our human evaluation phase. For each model, we report the average score from the three raters. The results indicate that the generated sequences of the BehavGAN make more sense when compared to the MLE-, the SeqGAN- and the LeakGAN-generated sequences. Table 9 provides a few samples of sequences that are generated by BehavGAN and the SeqGAN method, the method with best results among the baseline methods, to illustrate the superiority of the BehavGAN output. The SeqGAN-generated sequences simply repeat some frequent sub-sequences, for example, (LeaveHome, EnterHome) or (MealPreparation, Relax), but the method performs weakly in generating various activity sequences and does not follow some common-sense rules such as eating after meal preparation. The reason is the SeqGAN tries to maximize the similarity to ground-truth sequences, thereby sacrificing its diversity. BehavGAN does show better performance in generating various patterns, and its sequences mostly follow common-sense rules

Table 8. Human evaluation score

Method	Average Human Score
MLE	2.1
SeqGAN	3.2
LeakGAN	2.6
BehavGAN	**3.7**
Real data	**4.8**

of daily activities. The behaviour sequence in the model we propose can be simulated in the same way as GANs can simulate the meaningful order of words in linguistic models. The reason the generator doesn't generate sequences like "LeaveHome -> WashDishes -> EnterHome," for example, is that this pattern doesn't appear in the real sequences we supplied the model. Actually, the "LeaveHome" token always follows the "EnterHome" token. However, in comparison to the ground truth sequences, BehavGAN has issues in dealing with sleep duration.

6 DISCUSSION

Nowadays, the aging global population is an issue that puts financial burdens on governments. Health monitoring for older people can be a partial solution to this issue by providing timely care as well as predicting personal health conditions. While the availability of datasets on older people's behavior can benefit research, a limited number of relevant datasets are publicly available. We believe that introducing a method capable of generating quality sequence behavior data would contribute significantly to research in this area . In this regard, we have developed a method to employ Generative Adversarial Networks (GANs), which is proven to be effective in generating realistic images, texts or even music pieces based on rather small number of real data. Although GANs are considered as a state-of-the-art and successful method for synthetic image generation, they have rarely been used for generating human behavior data.

In our experiments, we found that parameters of the algorithm need to be set carefully in order to achieve optimal results. In this regard, to specify the threshold parameter we need to consider the sequence token space. In other words, for generating sequences that constitute a wide range of tokens, the threshold parameter should be set to a relatively lower number since the generated sequences are then less likely to be highly similar to the real data (reference set). Also, the number of generated sequences needs to be reasonable so that the model does not repeat itself in generating sequences. Moreover, we used BLEU-4 as the similarity metric because we found it appropriate based on our sequence lengths. For sequences of longer or shorter length, BLEU-4 might not be a valid choice.

For future research, behavior representation needs further improvement to capture more features of an individual's life. Due to the unavailability of a real dataset containing more features such as vital signs or health status we will await future developments of this nature.

7 CONCLUSION

In this paper, we introduced a new version of GANs, which applies GANs to the problem of behavior sequence generation by learning the features of a target dataset. Our proposed method contributes to the data generation literature by generating a diverse yet similar dataset that consists of sequences of a person's activities. In our proposed algorithm, BehavGAN, we introduce a combined reward method that incorporates the BLEU score in the reinforcement mechanism of the original SeqGAN algorithm. Our algorithm guides the generator in a direction that avoids generating identical sequences. To do so, we use the BLEU score as a similarity metric that evaluates the similarity of generated data to the

Table 9. Sample real sequences and sample sequences from the baseline methods and the proposed method

Method	**Sample Sequence**
SeqGAN	Relax133;MealPreparation20;Relax31;LeaveHome2;EnterHome137;MealPreparation36;Relax138; MealPreparation69;Relax199;Sleeping62
SeqGAN	LeaveHome2;EnterHome101;Relax221;MealPreparation26;Relax125;MealPreparation85;Relax282; Sleeping367;BedtoToilet1;Sleeping43
SeqGAN	Relax145;LeaveHome2;EnterHome144;LeaveHome2;EnterHome101;LeaveHome2;EnterHome150; MealPreparation10;Relax140;LeaveHome2
SeqGAN	MealPreparation41;Relax129;MealPreparation62;Relax47;MealPreparation63;Relax84;LeaveHome1; EnterHome159;LeaveHome1;EnterHome138
BehavGAN	Eating9;Relax74;Work27;MealPreparation22;Relax51;Sleeping183;BedtoToilet1;Sleeping180; BedtoToilet3;Sleeping117
BehavGAN	LeaveHome1;EnterHome128;WashDishes6;Relax26;MealPreparation61;Eating32;Relax94; Sleeping452;BedtoToilet2;Sleeping148
BehavGAN	EnterHome153;Eating17;Relax87;Work58;MealPreparation86;Relax26;Eating36;Sleeping171; BedtoToilet3;Sleeping224
BehavGAN	Eating6;Relax141;WashDishes4;Relax70;Sleeping64;BedtoToilet2;Sleeping311;MealPreparation72; Relax103;Eating45
BehavGAN	MealPreparation11;Realx7;MealPreparation45;Relax3;MealPreparation18;Relax19;MealPreparation6; Relax1;Eating&Relax15;Relax9
CASAS-Aruba	LeaveHome2;EnterHome145;Relax234;Housekeeping9;Relax89;Work20;Relax340;Sleeping343; BedtoToilet4;Sleeping377
CASAS-Aruba	Relax141;MealPreparation17;Eating19;WashDishes4;Relax51;LeaveHome1;EnterHome125; MealPreparation84;Relax287;LeaveHome1
CASAS-Aruba	MealPreparation15;Eating9;MealPreparation57;Eating13;Relax72;Eating16;Relax138; Housekeeping19;Work66;MealPreparation33
CASAS-Aruba	Relax313;Sleeping359;BedtoToilet1;Sleeping350;MealPreparation36;Relax87;Eating62;WashDishes7; Relax22;LeaveHome2
CASAS-Aruba	Relax61;Sleeping58;MealPreparation18;Relax1;Eat&Relax7;Relax59; MealPreparation6; Relax3;MealPreparation25;Relax4; Eating12

Note: Each event is represented by the type of activity performed followed by its duration in minutes. Events are separated by ";".

input data. We also enhanced the speed of the BLEU score calculation via a hash table structure to make it possible to incorporate the BLEU score into our new reward method.

We have implemented the BehavGAN algorithm and tested it by generating datasets of human behavior sequences based on two different ground truth datasets. To accomplish this, we encoded the behavior as sequences of activities. Synthetic data which was generated by GAN was evaluated in terms of its similarity to the real data as well as diversity of the generated samples. We also design a machine learning task (MLM) to showcase the effectiveness of generated data in improving the learning of the task. Our results show that BehavGAN is more effective in generating behavior sequences compared to MLE, LeakGAN, and the original SeqGAN algorithm. Finally, a human evaluation is carried out to determine the quality of the generated data. Our proposed algorithm outperforms state-of-the-art methods when it comes to generating behavior sequences consisting of limited-space sequence tokens.

REFERENCES

[1] Md Momin Al Aziz, Tanbir Ahmed, Tasnia Faequa, Xiaoqian Jiang, Yiyu Yao, and Noman Mohammed. 2021. Differentially Private Medical Texts Generation Using Generative Neural Networks. *ACM Transactions on Computing for Healthcare (HEALTH)* 3, 1 (2021), 1–27.

[2] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. 2012. A review of smart homes—Past, present, and future. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)* 42, 6 (2012), 1190–1203. ISBN: 1094-6977 Publisher: IEEE.

[3] Talal Alshammari, Nasser Alshammari, Mohamed Sedky, and Chris Howard. 2018. SIMADL: simulated activities of daily living dataset. *Data* 3, 2 (2018), 11. Publisher: Multidisciplinary Digital Publishing Institute.

[4] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. 2017. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 188–193.

[5] Farzad Amirjavid, Abdenour Bouzouane, and Bruno Bouchard. 2014. Data driven modeling of the simultaneous activities in ambient environments. *Journal of Ambient Intelligence and Humanized Computing* 5, 5 (2014), 717–740. Publisher: Springer.

[6] Damla Arifoglu and Abdelhamid Bouchachia. 2019. Abnormal Behaviour Detection for Dementia Sufferers via Transfer Learning and Recursive Auto-Encoders. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 529–534.

[7] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. 2019. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association* 26, 3 (2019), 228–241. Publisher: Oxford University Press.

[8] Hapugahage Thilak Chaminda, Vitaly Klyuev, and Keitaro Naruse. 2012. A smart reminder system for complex human activities. In *2012 14th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 235–240.

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*. 2172–2180.

[10] Diane J. Cook. 2010. Learning setting-generalized activity models for smart spaces. *IEEE intelligent systems* 2010, 99 (2010), 1. Publisher: NIH Public Access.

[11] Diane J Cook and Maureen Schmitter-Edgecombe. 2021. Fusing ambient and mobile sensor features into a behaviorome for predicting clinical health scores. *IEEE Access* 9 (2021), 65033–65043.

[12] Samundra Deep, Xi Zheng, Chandan Karmakar, Dongjin Yu, Leonard GC Hamey, and Jiong Jin. 2019. A survey on anomalous behavior detection for elderly care using dense-sensing networks. *IEEE Communications Surveys & Tutorials* 22, 1 (2019), 352–370. Publisher: IEEE.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Giovanni Diraco, Alessandro Leone, and Pietro Siciliano. 2019. AI-Based Early Change Detection in Smart Living Environments. *Sensors* 19, 16 (2019), 3549. Publisher: Multidisciplinary Digital Publishing Institute.

[15] Ali el Hassouni, Mark Hoogendoorn, and Vesa Muhonen. 2018. Using generative adversarial networks to develop a realistic human behavior simulator. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 476–483.

[16] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633* (2017).

[17] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3426–3433.

[18] Roschelle Fritz, Katherine Wuestney, Gordana Dermody, and Diane J Cook. 2022. Nurse-in-the-loop smart home detection of health events associated with diagnosed chronic conditions: A case-event series. *International Journal of Nursing Studies Advances* (2022), 100081.

[19] Sylvain Gelly and David Silver. 2011. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* 175, 11 (2011), 1856–1875. ISBN: 0004-3702 Publisher: Elsevier.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[21] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)* 3, 1 (2021), 1–23.

[22] Bin Guo, Hao Wang, Yasan Ding, Wei Wu, Shaoyang Hao, Yueqi Sun, and Zhiwen Yu. 2021. Conditional Text Generation for Harmonious Human-Machine Interaction. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 2 (2021), 1–50.

[23] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. Issue: 1.

[24] Mehak Gupta, Thao-Ly T Phan, H Timothy Bunnell, and Rahmatollah Beheshti. 2022. Obesity Prediction with EHR Data: A deep learning approach with interpretable elements. *ACM Transactions on Computing for Healthcare (HEALTH)* 3, 3 (2022), 1–19.

[25] Yongkoo Han, Manhyung Han, Sungyoung Lee, A. M. Sarkar, and Young-Koo Lee. 2012. A framework for supervising lifestyle diseases using long-term activity monitoring. *Sensors* 12, 5 (2012), 5363–5379. Publisher: Molecular Diversity Preservation International.

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780. Publisher: MIT Press.

[27] Théo Jourdan, Antoine Boutet, Amine Bahi, and Carole Frindel. 2020. Privacy-Preserving IoT framework for activity recognition in personal healthcare monitoring. *ACM Transactions on Computing for Healthcare* 2, 1 (2020), 1–22.

[28] Stein Kristiansen, Konstantinos Nikolaidis, Thomas Plagemann, Vera Goebel, Gunn Marit Traaen, Britt Øverland, Lars Aakerøy, Tove-Elizabeth Hunt, Jan Pål Loennechen, Sigurd Loe Steinshamn, et al. 2021. Machine Learning for Sleep Apnea Detection with Unattended Sleep Monitoring at Home. *ACM Transactions on Computing for Healthcare* 2, 2 (2021), 1–25.

[29] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, and Zehan Wang. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

[30] Yukyung Lee, Jina Kim, and Pilsung Kang. 2021. LAnoBERT: System Log Anomaly Detection based on BERT Masked Language Model. *arXiv preprint arXiv:2111.09564* (2021).

[31] Dingwen Li, Jay Vaidya, Michael Wang, Ben Bush, Chenyang Lu, Marin Kollef, and Thomas Bailey. 2020. Feasibility Study of Monitoring Deterioration of Outpatients Using Multimodal Data Collected by Wearables. *ACM Transactions on Computing for Healthcare* 1, 1 (2020), 1–22.

[32] Yantao Li, Jiaxing Luo, Shaojiang Deng, and Gang Zhou. 2021. CNN-based Continuous Authentication on Smartphones with Conditional Wasserstein Generative Adversarial Network. *IEEE Internet of Things Journal* (2021).

[33] Shuo Liu, Mingliang Gao, Vijay John, Zheng Liu, and Erik Blasch. 2020. Deep Learning Thermal Image Translation for Night Vision Perception. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 1 (2020), 1–18.

[34] Angela Lopez-del Rio, Maria Martin, Alexandre Perera-Lluna, and Rabie Saidi. 2020. Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. *Scientific reports* 10, 1 (2020), 1–14.

[35] Angela Lopez-del Rio, Alfons Nonell-Canals, David Vidal, and Alexandre Perera-Lluna. 2019. Evaluation of cross-validation strategies in sequence-based binding prediction using deep learning. *Journal of chemical information and modeling* 59, 4 (2019), 1645–1657.

[36] Ahmad Lotfi, Caroline Langensiepen, Sawsan M. Mahmoud, and Mohammad Javad Akhlaghinia. 2012. Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour. *Journal of ambient intelligence and humanized computing* 3, 3 (2012), 205–218. ISBN: 1868-5137 Publisher: Springer.

[37] Yun Luo, Li-Zhen Zhu, Zi-Yu Wan, and Bao-Liang Lu. 2020. Data augmentation for enhancing EEG-based emotion recognition with deep generative models. *Journal of Neural Engineering* 17, 5 (2020), 056021.

[38] Weina Ma and Kamran Sartipi. 2015. Synthesizing scenario-based dataset for user behavior pattern mining. *International Journal of Computer and Information Technology* 4, 6 (2015), 855–866.

[39] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[40] Parisa Fard Moshiri, Hojjat Navidan, Reza Shahbazian, Seyed Ali Ghorashi, and David Windridge. 2020. Using GAN to Enhance the Accuracy of Indoor Human Activity Recognition. *arXiv preprint arXiv:2004.11228* (2020).

[41] Mohamed Tarik Moutacalli, Abdenour Bouzouane, and Bruno Bouchard. 2015. The behavioral profiling based on times series forecasting for smart homes assistance. *Journal of Ambient Intelligence and Humanized Computing* 6, 5 (2015), 647–659. Publisher: Springer.

[42] Ehsan Nazerfard. 2018. Temporal features and relations discovery of activities from sensor data. *Journal of Ambient Intelligence and Humanized Computing* (2018), 1–16. Publisher: Springer.

[43] Skyler Norgaard, Ramyar Saeedi, Keyvan Sasani, and Assefaw H. Gebremedhin. 2018. Synthetic sensor data generation for health applications: A supervised deep learning approach. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 1164–1167.

[44] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. 2018. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics* 34, 17 (2018), i821–i829.

[45] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[46] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[47] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[48] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396* (2016).

[49] Jennifer Renoux and Franziska Klugl. 2018. Simulating daily activities in a smart home for data generation. In *2018 Winter Simulation Conference (WSC)*. IEEE, 798–809.

[50] Daniele Riboni, Claudio Bettini, Gabriele Civitarese, Zaffar Haider Janjua, and Rim Helaoui. 2015. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 149–154.

[51] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620. ISBN: 0001-0782 Publisher: ACM New York, NY, USA.

[52] Yuxuan Song, Ning Miao, Hao Zhou, Lantao Yu, Mingxuan Wang, and Lei Li. 2020. Improving maximum likelihood training for text generation with density ratio estimation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 122–132.

[53] Yifu Sun and Haoming Jiang. 2019. Contextual text denoising with masked language models. *arXiv preprint arXiv:1910.14080* (2019).

[54] Ahmet Sureyya Rifaioglu, Tunca Doğan, Maria Jesus Martin, Rengul Cetin-Atalay, and Volkan Atalay. 2019. DEEPred: automated protein function prediction with multi-task feed-forward deep neural networks. *Scientific reports* 9, 1 (2019), 1–16.

[55] Nagender Kumar Suryadevara, Subhas C. Mukhopadhyay, Ruili Wang, and R. K. Rayudu. 2013. Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Engineering Applications of Artificial Intelligence* 26, 10 (2013), 2641–2652. ISBN: 0952-1976 Publisher: Elsevier.

[56] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[57] Jonathan Synnott, Chris Nugent, and Paul Jeffers. 2015. Simulation of smart home activity datasets. *Sensors* 15, 6 (2015), 14162–14179. Publisher: Multidisciplinary Digital Publishing Institute.

[58] Tim Van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. 2008. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*. 1–9.

[59] Jiwei Wang, Yiqiang Chen, Yang Gu, Yunlong Xiao, and Haonan Pan. 2018. SensoryGANs: an effective generative adversarial framework for sensor-based human activity recognition. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[60] Min Wang, Congyan Lang, Liqian Liang, Songhe Feng, Tao Wang, and Yutong Gao. 2020. End-to-End Text-to-Image Synthesis with Spatial Constrains. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 4 (2020), 1–19.

[61] Chao Yan, Ziqi Zhang, Steve Nyemba, and Bradley A. Malin. 2020. Generating Electronic Health Records with Multiple Data Types and Constraints. *arXiv preprint arXiv:2003.07904* (2020).

[62] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. 2017. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847* (2017).

[63] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[64] Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. 2018. Generative adversarial network for synthetic time series data generation in smart grids. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 1–6.

[65] Yan Zhao, Baoqiang Ma, Pengbo Jiang, Debin Zeng, Xuetong Wang, and Shuyu Li. 2020. Prediction of Alzheimer's Disease Progression with Multi-Information Generative Adversarial Network. *IEEE Journal of Biomedical and Health Informatics* (2020). ISBN: 2168-2194 Publisher: IEEE.