PROJECT FINAL REPORT FOR CPTS 534 SPRING SEMESTER 2022

Anonymous authors

ABSTRACT

This is a final report for Neural Networks class project on the comparison of an LSTM Wasserstein Generative Adversarial Network (stateful GAN) to a basic Generative Adversarial Network on time-series Smart Home data.

1 INTRODUCTION

The goal of this project it to create synthetic data that emulates an authentic dataset using a Wasserstein Generative Adversarial Network. Synthetic datasets are incredibly beneficial in machine and AI learning. Generated datasets can be used to bolster authentic data, to protect privacy and confidentiality of authentic data, to reduce cost of time and resources of authentic data, to simulate desired data, or to resolve biases and imbalances in data. It is a huge advantage to be able to create realistic synthetic data, saving huge amounts of time and effort on data collection, processing, and organization. The authentic dataset for this project is composed of time-series smart-home data which will be used to produce synthetic data. The reasoning for enlarging the authentic dataset is for the purpose of furthering the accuracy and classification for gerontechnology. The application for gerontechnology and this dataset is the classification of sensor data to activities. Classifying activities is beneficial in order to prevent or alert of harmful states of elderly in their homes. Many of the elderly that are not in a retirement, family, or hospice care are living alone and are at risk of not being able to access urgent care after a incident or accident in their homes. And while classifying data is an important part of this issue there is also the challenge of collecting this data. It is time consuming and effort draining to obtain volunteers, permissions, set up sensors, and monitor activities. Some of the effort of this process can be mitigated by using a generative adversarial network to produce authentic synthetic data. By using a LSTM Wasserstein GAN we hope to improve the generative process by using a critic to return losses to the generator instead of a discriminator. The stateful WGAN does not have an oracle applications similar to ours as it is not designed for authentic data, so it will be it's own oracle. We will limit the scope of this project to compare the data similarity of our stateful GAN (oracle) to the traditional GAN (baseline).

2 LITERATURE REVIEW

In the paper "Time-series Generative Adversarial Network" the researchers analyze a timeGAN in comparison with C-RNN-GAN, RCGAN, T-Forcing, P-Forcing, WaveNet, and Wave GAN (1). Researchers propose and prove that the timeGAN, which preserves temporal dynamics through forced step wise supervised loss function, is better able to produce realistic data when it is forced to consider limited time frames. Using the train-on-synthetic, test-on-real technique, models are evaluated on by discriminative score, predictive score, and visualization. Results showed that the timeGAN preformed well across the board for different sets of time series data, and better than many other GANs. This relates to our project because our approach to force time steps and use a Wasserstein critic to improve realism of synthetic data aligns with this paper's research step wise loss function to force the GAN into considering time sequentially in limited windows.

Some approaches to improving time-series data production in GAN's, take a less ambitious path by using GANs to fill in sparse data rather than create entirely new data. This is the case for the researcher in "End-to-End Generative Adversarial Network for Multivariate Time Series Imputation" who utilize a GAN to "first learn to generate new samples that follow the distribution of the training dataset and then impute the missing values, have achieved state-of-the-art performance" which is modeled after Goodfellow and et al.'s work in "Generative adversarial networks" and preformed

outstandingly (2) (3). The goal, which was successfully met, of the End-to-End GAN researchers was to utilize the end-to-end model to simplify and reduce training time for the imputation of the data. In regards to our own project this work is only similar to ours in that it imputes time-series data and utilizes a GAN, in all other aspects there are no similarities, their evaluation methods, GAN structure, and focuses are displaced from our own goal.

For irregularly sampled time series data, conditional GANs have become favored as it allows an additional parameter to be fed to both the generator and the discriminator that can help increase accuracy. A time-Conditional GAN as used in "T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling" to limit time dimensions in line with the real data set (4). The model proposed model is a Conditional Generative Adversarial Network Model composed by two CNNs, one for the generative part and one for the discriminative part. The comparison for the T-CGAN in this paper is against techniques time-slicing and time-warping methods rather than other models, and the T-CGAN out performs both methods. This T-CGAN while dissimilar to our WGAN does aim to refine the time windows as in "Time-series Generative Adversarial Network" and our own efforts, and the trend that having more accurate time windows increase accuracy of synthetic data creation and testing increases our own hope of a successful outcome.

Other techniques for augmenting time series data have been proposed in literature, such as jittering, scaling, warping and permutation. And is a varied selection of studies utilizing GAN models for generating synthetic sequences in domains such as text (5), finance (6), bio signals (7), sensor (8), and smart grid data (9). The authors in (10) implemented a data augmentation method for wearable sensor time series data to classify the motor state of patients with Parkinson's disease. This work in gerontechnology emulates the goals for this project to be able to better able to classify movements of older individuals in their homes, as to recognize death, or injury. As with all techniques and implementations of any neural network there is no best way and these related works provide a solid foundation for this project.

3 Data Set

The preliminary dataset for this project is home sensor data composed of a time stamp, sensor, signal, and label of activity. It was retrieved from Washington State University's CASAS at this web address: http://casas.wsu.edu/datasets/. The raw dataset is distributed among 6 csv files of data, about 80 megabytes. Raw data is as shown in Figure 2, processing leaves it in the form shown in Figure 3. Since the goal of a GAN is to create data our synthetic data will be based of the processed data form. Our processed data is composed of time stamp, the 34 original sensors and 14 labels of activity which are vectorized labels.

4 Baseline

Our baseline is a simple generative adversarial network composed of a 1D-convolutional layers (3). This is one of the most simplistic versions of GAN models, that generated a simple monochromatic, low-resolution images. Each image/window is sized as (384, 48, 1), 48 is the number of features included with the labels, and 384 is the time frame used to maintain a sequential time series. The sequentially is key due to the importance of time in the data. It would be incorrect to have sensors catalog time frames a day apart, or even years apart with sensor data that is occurring in a hour time frame. Even a simple GAN has several components the generator and the discriminator or critic, Section 10 shows the basic GAN model implemented. The baseline's discriminator is composed of 4 1D convolutional layers in tandem with leaky ReLU, dropout and batch normalization. The generator only utilizes a single 1D convolutional layer just before the generator model is wrapped in a hyperbolic tangent activation function, others layers are 2D convolutional transpose layers with batch normalization and leaky ReLU. Metrics for both the discriminator and generator used losses for binary cross-entropy and sparse categorical cross-entropy, Adam for optimization and for metrics generic keras accuracy.

5 Main Approach

The approach for this project is to build a generative adversarial network with a stateful generator where the state is a bi-directional LSTM layer. Our goal is to generate synthetic data and GANs align with that goal. Our challenges will be to preserve previous windows' context and allow arbitrary amounts of data with context and enforce monotonically increasing step times. Both of these choices are made to allow for generated data to capture and generalize time series data. Pros and cons of our discriminator model choices are that LSTMs are slower than Convolutional layers, so although the bi-directional LSTM should allow for better feedback in our critic it will take longer than just a CNN. A pro of LSTMs is the ability to recognize past to future dependencies and future to past dependencies. The only implementation choice we made was to restrict the sensors to ones with binary signals.

To measure the effectiveness of GAN models, we decide to test on the generated data with a CNN model. The goal is to compare generated data between base GAN & stateful GAN and real data. Since it's a time series data and we want to preserve the temporal relationship, a window size of 16 is used, which means 16 rows of data are treated as one image, therefore the image size would be 16×34 where 34 is the number of features. The output would be a one hot encoding result for 14 features.

First step, was to generated data from base GAN and stateful GAN, both of which are trained for 10 epochs. After GAN models training, we obtain 160,000 data records from the GAN models, respectively. Each data record is an image size of 16 x 34 as mentioned earlier. The data then go through the CNN model for training. The architecture of CNN model is shown in Figure 8. It consists of four convolutional blocks, each block is a Conv1D layer followed by activation function. The model outputs a 1 x 14 one hot encoded array that represents the predicted label. Finally, we input the real data with the same format into the trained models to get accuracy of both models.

6 Evaluation Metric

To compare our GANs, we compare them to the real data distribution. This is done by joint probability distributions, proportion checking, TSTR vs TRTR (train on synthetic; test on real vs train on real; test on real), and Granger causality.

To check our GAN's progress during training, we look to the loss functions. The base GAN uses binary cross-entropy and the stateful GAN uses Wasserstein distance plus a gradient penalty (to entice an L2-unit norm of weights).

Our qualitative metrics are shown at the end of this report. They include line plots of losses and accuracies, bar plots, and heat-maps. Our quantitative metrics are the final accuracy on the testing data in TSTR versus TRTR, the joint probabilities, and Granger causality.

7 Results & Analysis

7.1 Baseline

Results for our baseline were poor, as expected. For the training of the GAN there was a trend of the accuracy be high and error rate decreasing, but only to a certain point. Meddling with epoch length starting at 1000 epochs, then 500, to 200, and then to 100 it was determined that there was a sweet spot of minimal error and high accuracy at about 100 either slightly above or below. After the sweet spot error continually jumped up and accuracy for the real data completely dropped and began only fitting to the synthetic data, which implies over-fitting for fake data once a certain number of iterations, mostly likely due to mode collapse. The mode collapse is expected here, and we hope to avoid it with our implementation of the stateful GAN. In the longer epochs for the basic GAN the over-fitting of the fate data is changed at about 400 epochs for accuracy and error, instead the real data returns to fitting as the fake data increases till 500 epochs and then plateaus at a higher error rate than the real data, as can be seen in image Figure 5.

$$W_1([.9, .1, .4, .2], [1, 0, 1, 0]) < W_1([.9, .1, .4, .2], [0, 1, 0, 1]) \tag{1}$$

So,

$$[.9, .1, .4, .2] \implies [1, 0, 1, 0] \tag{2}$$

Figure 1: Mapping the output data to alternating signals. We use earth-movers distance (Wasserstein distance) to calculate the distance between the two distributions.

7.2 Baseline 2

We also implemented another baseline GAN for assurance. This second GAN suffered from mode collapse and alternated between sensors M013 and M015 with the activity "Meal Preparation". This GAN uses CNN architecture for the discriminator and the generator with binary cross entropy as the loss function and the ADAM optimizer.

7.3 Main Approach: Stateful GAN

The stateful GAN reliably considers the context of previous windows and does not have mode collapse. However, the generator omits some of the less represented sensors and activities. Likely, the climb to reach that better minima was too great for the generator—it was cheaper to omit a couple activities. The generator loss's trend increases over epochs (Figure 12). However, as of 25 epochs, it does not diverge. The critic's loss is more consistent. For the stateful GAN's train on synthetic; test on real (TSTR) (Figure 11), the testing on real was as accurate as it was to train on real (Figure 9).

Besides the omission of some underrepresented sensors, the joint probability distributions of sensor|activity and activity|sensor closely match the real data (see Figure 15 and Figure 14). The distributions of sensor and activity frequency are also close (Figure 16). However, we found that the signals were not strictly alternating given the same sensor. To fix this, we implemented Wasserstein distance in post-processing to find the alternating sequence closest to the predicted sequence (see Figure 1). All in all, we have a GAN that can generate an arbitrary amount of time series data in context.

7.4 Comparing Approaches

With a base GAN2 and a WGAN, there are inevitable differences in success to the approach. To compare the authenticity of both GAN models data production, we use the CNN model to implement the train on synthetic and test on real method. The control for accuracy is set by training and testing our CNN model with real data. Our original hypothesis is that our base GAN1 and GAN2 would do worse than the WGAN. The evidence validates our hypothesis, and the performance gap between the WGAN and base GANs performances was surprising. The WGAN's accuracy was on par with the train on real and test on real performance. Base GAN2, when trained, failed to accurately classify even a decent percentage of the real data. This means that the base generative data is not accurately replicating our real data, to even a small degree. This is mostly likely due to a high mode collapse that the model falls into, regardless of loss changes and penalties attempting to improve accuracy. In Figure 9 We can see that the CNN model trained and tested on real data is able to achieve roughly 70% accuracy. The stateful GAN is able to achieve 65% accuracy, and base GAN2 accuracy is not even 1%. We can correctly assume that the Wasserstein approach for our critic in the GAN was beneficial and booster authenticity of data production.

8 Error Analysis

An analysis initially implemented was Granger-Causality, a statistical concept of causality that is based on prediction. "According to Granger causality, if a signal X1 "Granger-causes" (or "G-causes") a signal X2, then past values of X1 should contain information that helps predict X2 above and beyond the information contained in past values of X2 alone"(11). There are several troubles with implementation of Granger-Causality in addition to arguments over implementation and correct

use. Results are obtained from single instances of real home data, multiple home data, synthetic data from the stateful GAN in single time windows, and synthetic base GAN1 over multiple time windows. These results were difficult to understand and explain, so they have been left out of our final analysis. They are available to view in the git repository under the synthetic-data folder.

Our stateful GAN's (LSTM + WGAN) losses are seen in Figure 12. As is evident, the generator had higher and more frequent changes in loss, while the critic of the WGAN maintains its losses. The generator fluctuates between -10 and 40 for losses, five times that of the critic which stays with a margin of about 10.

9 Future Work

A slight issue in out data was that under represented features are not replicated in the WGAN generator which means that certain sensors are not represented in our synthetic data. To improve this we can send the critic more windows of less common sensors. We call this "window replay" and its analogous to action replay in reinforcement learning. This forces the generator to look at examples of window data it is lacking. Another way to approach this issue is to greatly penalize the loss functions for missing entire sensors. This might look like an arcing loss function (similar to momentum). Again, our main challenge is the issue of omitted data in our generator. Another problem is the increased losses the generator experiences over the critic, which implies the critic is learning at a faster rate. To negate this trend it might be beneficial in the future to force the generator through more iterations than the critic to allow it more time to catch up in learning. We hope for future projects that we will be able to further improve our work.

10 Code

Link: https://github.com/jbroot/nn534Project

References

[1] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[2] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E2gan: End-to-end generative adversarial network for multivariate time series imputation," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, p. 3094–3100, 2019.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[4] G. Ramponi, P. Protopapas, M. Brambilla, and R. Janssen, "T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling," 2019.

[5] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, vol. 21, pp. 21–32, academia. edu, 2016.

[6] L. Simonetto, *Generating spiking time series with Generative Adversarial Networks: an application on banking transactions*. PhD thesis, Master's thesis, University of Amsterdam, 2018.

[7] S. Haradal, H. Hayashi, and S. Uchida, "Biosignal data augmentation based on generative adversarial networks," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 368–371, IEEE, 2018.

[8] M. Alzantot, S. Chakraborty, and M. B. Srivastava, "Sensegen: A deep learning architecture for synthetic sensor data generation," 2017.

[9] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "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)*, pp. 1–6, 2018.

[10] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, Nov 2017.

[11] A. Seth, "Granger causality," Jul 2007.

List of Figures

	Time	Sensor	Signal	Activity
0	2009-07-18 00:11:32.703001	M012	ON	Sleeping_in_Bed
1	2009-07-18 00:11:36.984001	M012	OFF	Sleeping_in_Bed
2	2009-07-18 00:14:15.015001	M012	ON	Sleeping_in_Bed
3	2009-07-18 00:14:19.296001	M012	OFF	Sleeping_in_Bed
4	2009-07-18 01:10:37.640001	M012	ON	Sleeping_in_Bed
5	2009-07-18 01:10:41.937001	M012	OFF	Sleeping_in_Bed
6	2009-07-18 01:43:16.343001	M012	ON	Sleeping_in_Bed
7	2009-07-18 01:43:20.640001	M012	OFF	Sleeping_in_Bed
8	2009-07-18 01:57:25.765001	M012	ON	Sleeping_in_Bed
9	2009-07-18 01:57:30.375001	M012	OFF	Sleeping_in_Bed
10	2009-07-18 03:07:25.015001	M012	ON	Sleeping_in_Bed
11	2009-07-18 03:07:28.968001	M012	OFF	Sleeping_in_Bed
12	2009-07-18 03:34:18.000001	M012	ON	Sleeping_in_Bed
13	2009-07-18 03:34:22.296001	M012	OFF	Sleeping_in_Bed
14	2009-07-18 03:35:02.468001	M012	ON	Sleeping_in_Bed
15	2009-07-18 03:35:06.500001	M012	OFF	Sleeping_in_Bed
16	2009-07-18 04:51:57.281001	M012	ON	Sleeping_in_Bed
17	2009-07-18 04:52:01.593001	M012	OFF	Sleeping_in_Bed
18	2009-07-18 04:54:12.281001	M012	ON	Sleeping_in_Bed
19	2009-07-18 04:54:16.156001	M012	OFF	Sleeping_in_Bed
20	2009-07-18 05:16:50.781001	M012	ON	Sleeping_in_Bed
21	2009-07-18 05:16:55.078001	M012	OFF	Sleeping_in_Bed
22	2009-07-18 05:39:40.187001	M012	ON	Sleeping_in_Bed
23	2009-07-18 05:39:44.109001	M012	OFF	Sleeping_in_Bed
24	2009-07-18 05:59:12.234001	M012	ON	Sleeping_in_Bed

Figure 2: Raw Authentic Data In Data Frame View

	Time	Signal	D021	D022	D023	D024	D025	D026	D027	D028	...	Housekeeping	Leave_Home	Meal_Preparation	Other_Activity	Personal_Hygiene	Relax	Sleeping_Not_in_Bed	Sleeping_in_Bed	Take_Medicine	Work
0	1247875892703001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
1	1247875896984001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
2	1247876055015001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3	1247876059296001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
4	1247879437640001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
5	1247879441937001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
6	1247881396343001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
7	1247881400640001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
8	1247882245765001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
9	1247882250375001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
10	1247886445015001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
11	1247886448968001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
12	1247888058000001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
13	1247888062296001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
14	1247888102468001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
15	1247888106500001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
16	1247892717281001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
17	1247892721593001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
18	1247892852281001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
19	1247892856156001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
20	1247894210781001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
21	1247894215078001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
22	1247895580187001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
23	1247895584109001000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
24	1247896752234001000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0

Figure 3: Processed Authentic Data In Data Frame View

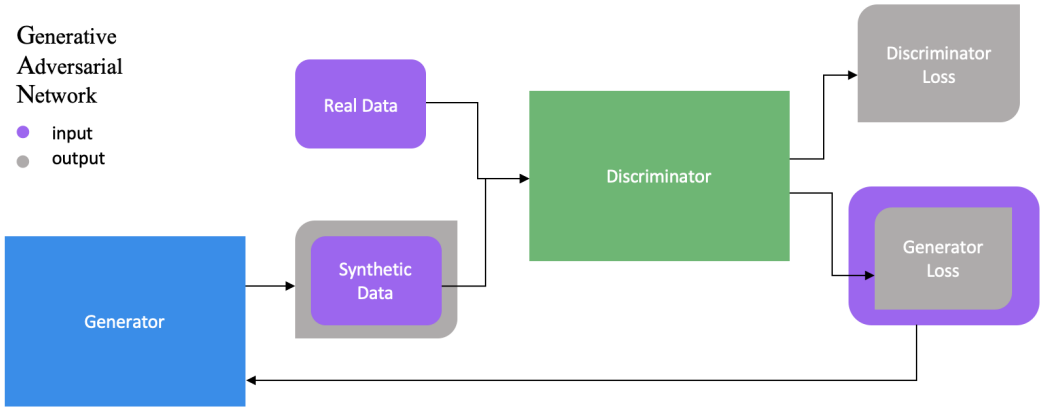


Figure 4: Basic GAN Model

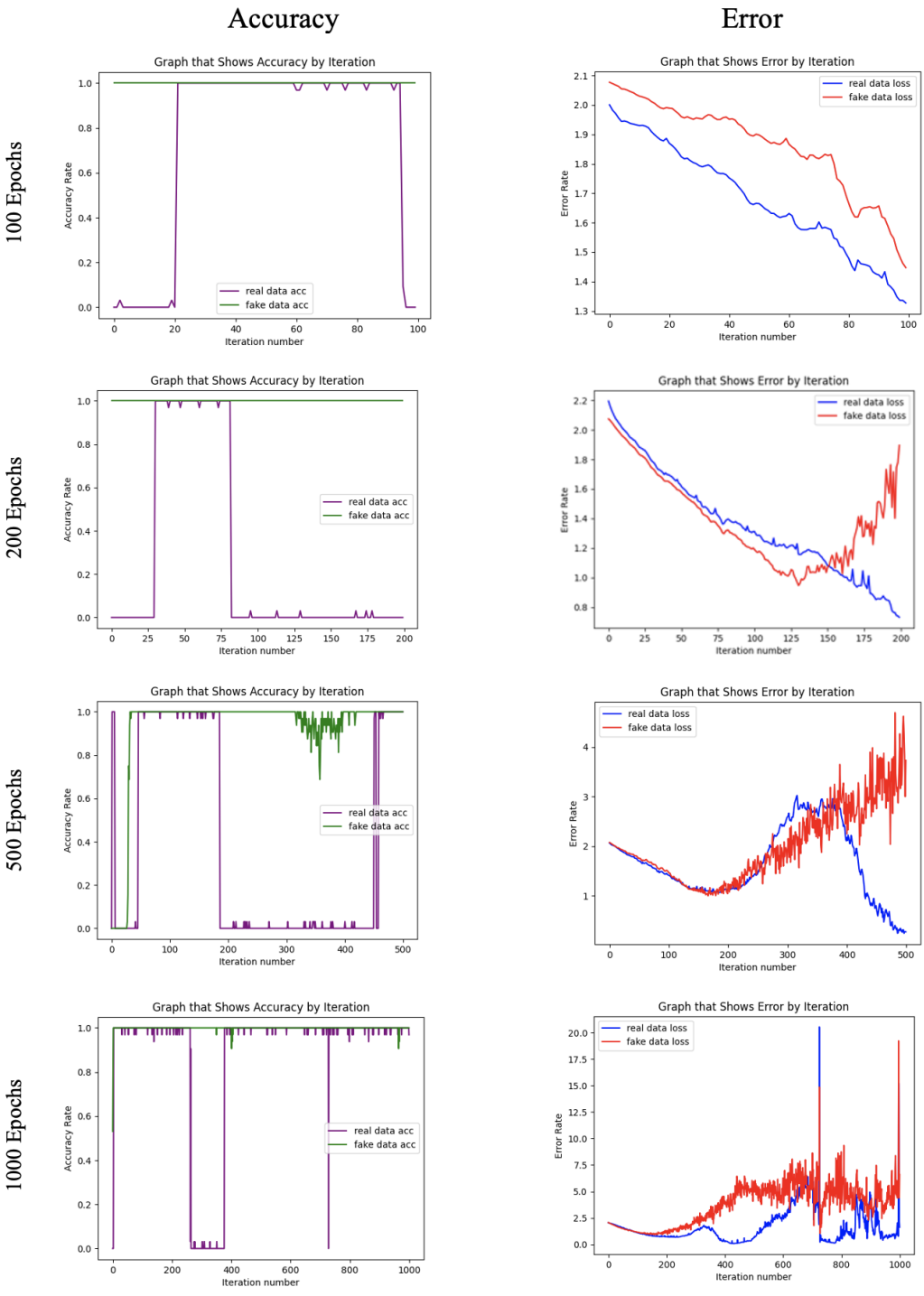


Figure 5: Basic GAN Accuracy & Error by Epoch

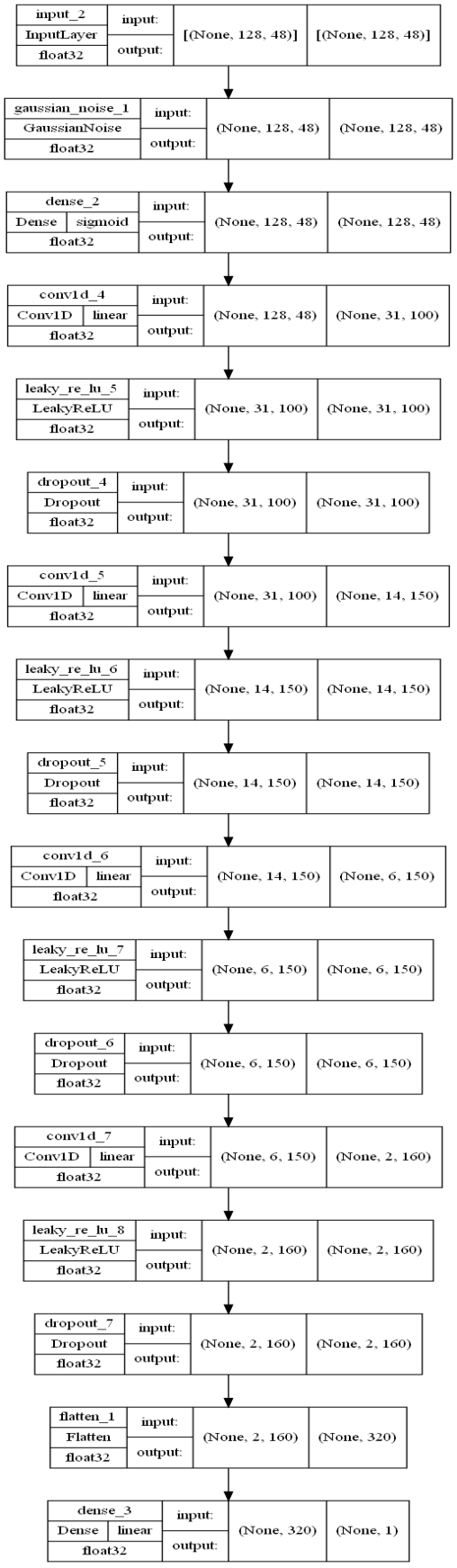


Figure 6: Stateful GAN's critic

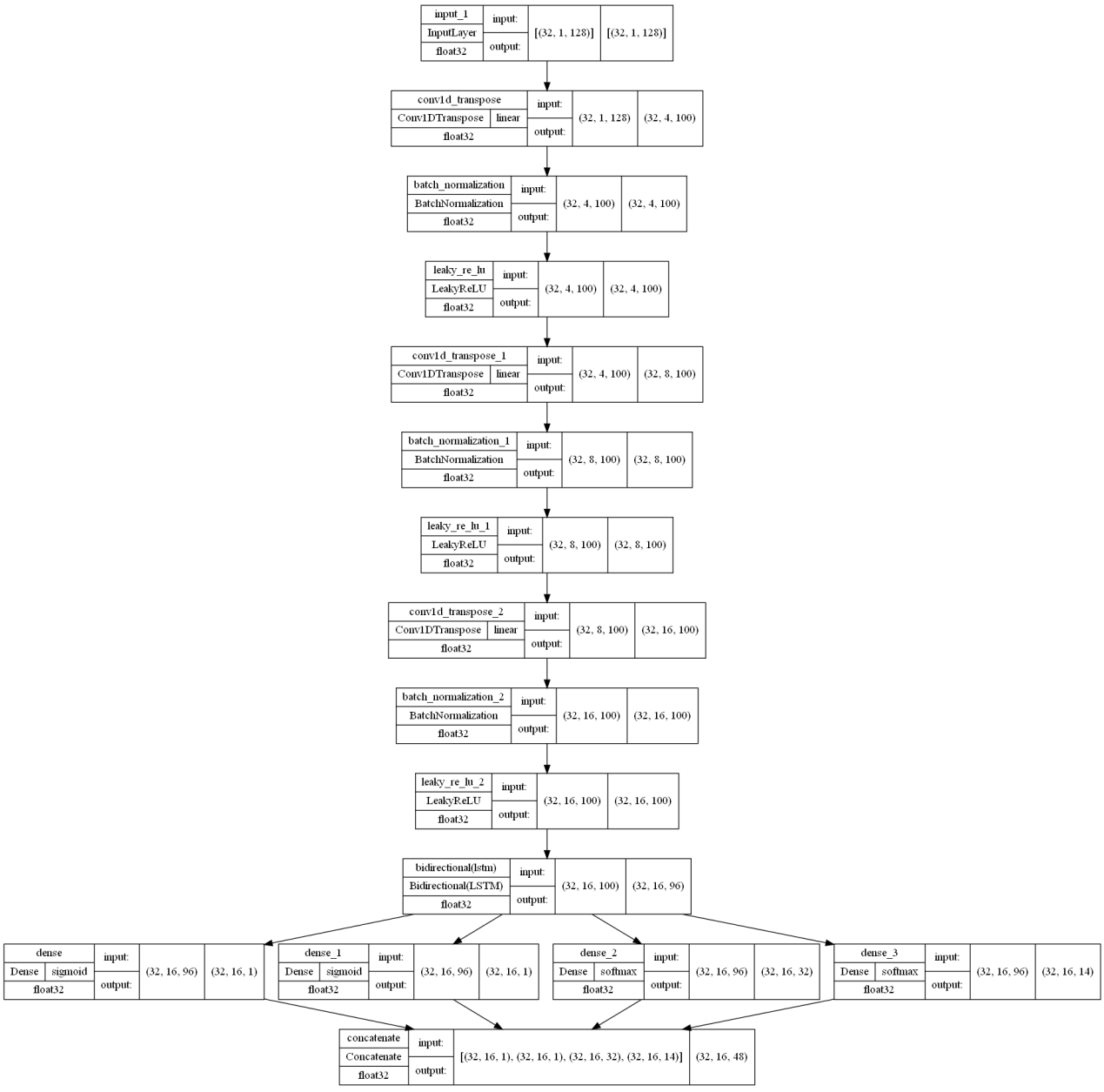


Figure 7: Stateful GAN's generator.

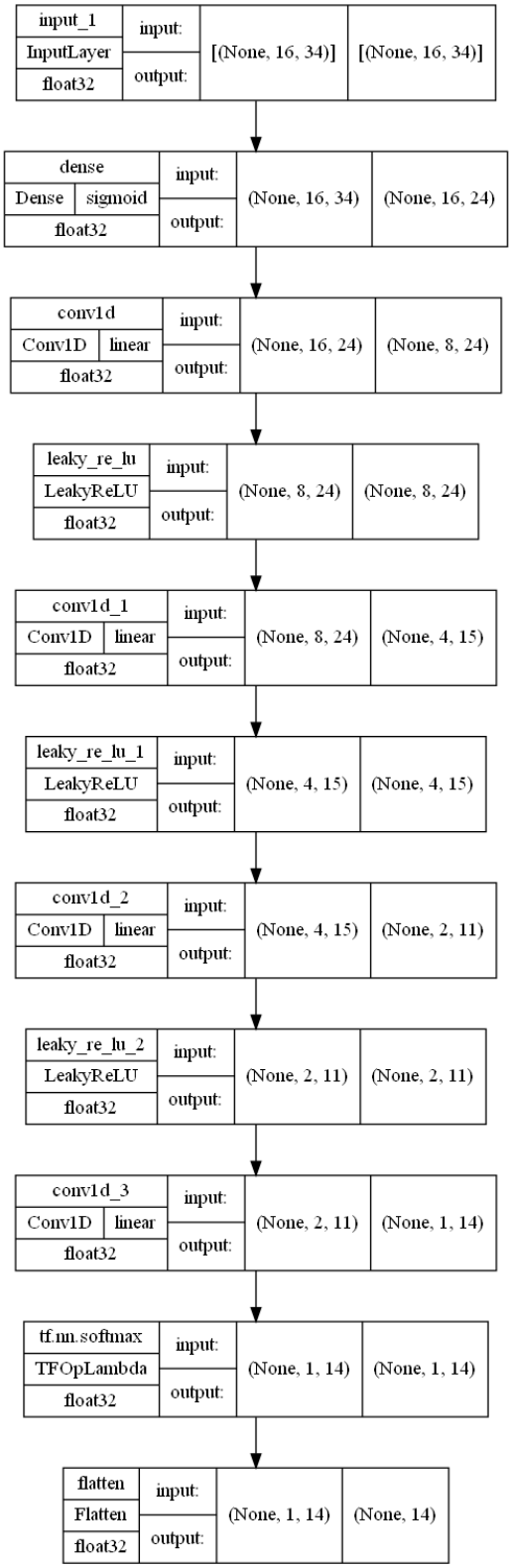


Figure 8: CNN architecture (classifier)

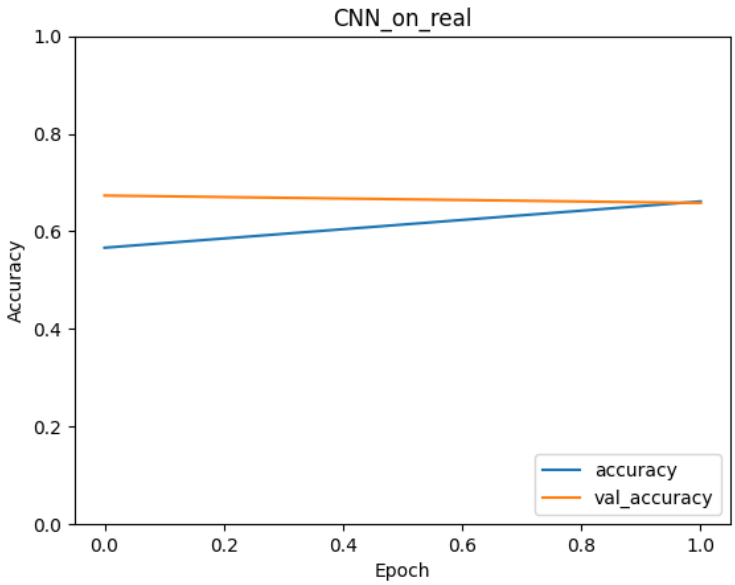


Figure 9: Accuracy of CNN model trained on real data and tested on real data (val_accuracy)

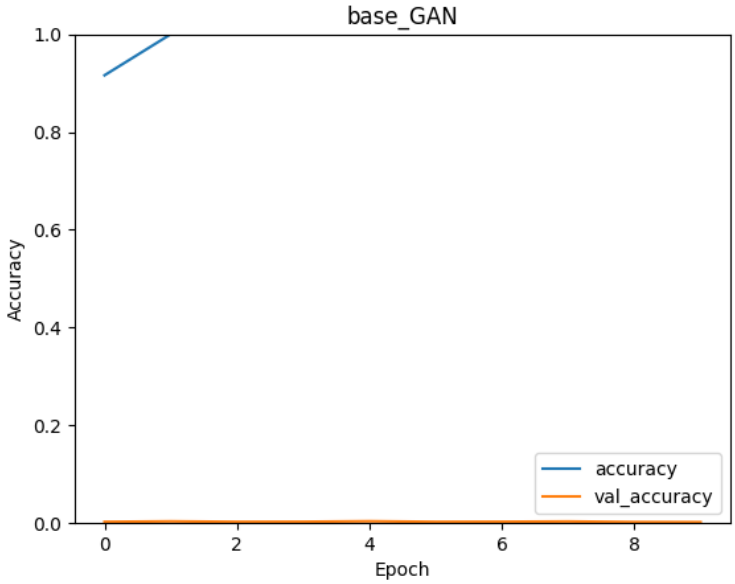


Figure 10: Accuracy of CNN model trained on base GAN data and tested on real data (val_accuracy)

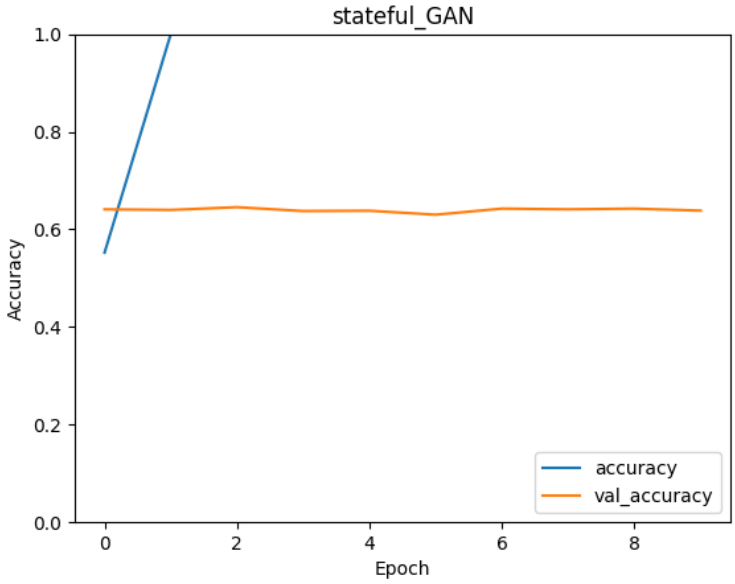


Figure 11: Accuracy of CNN model trained on stateful GAN data and tested on real data (val_accuracy)

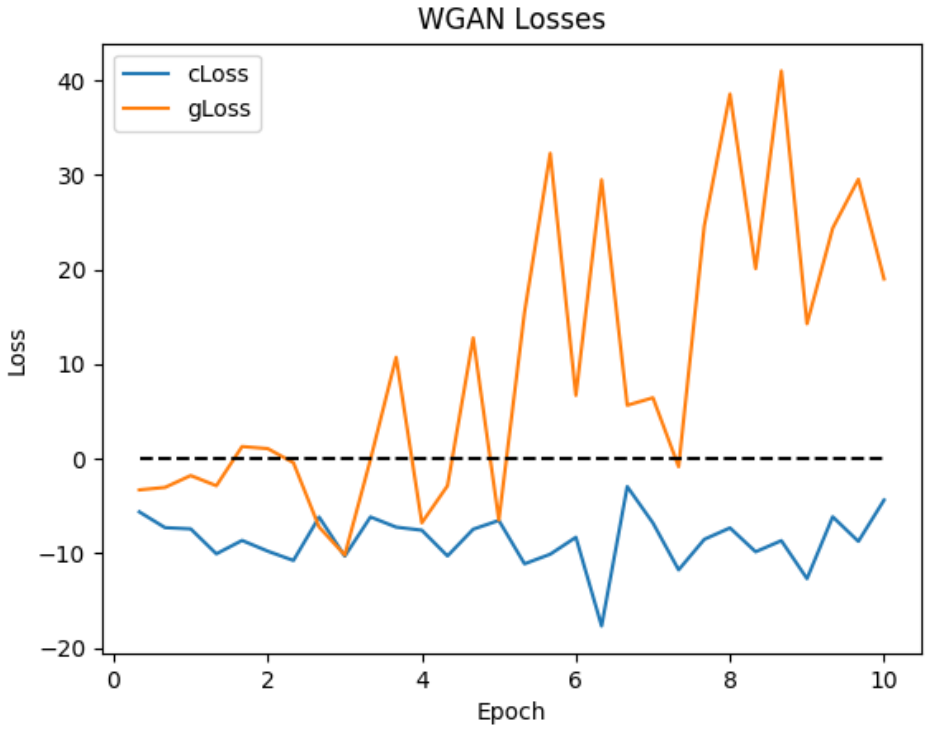
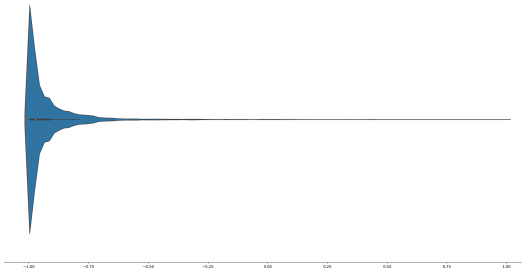
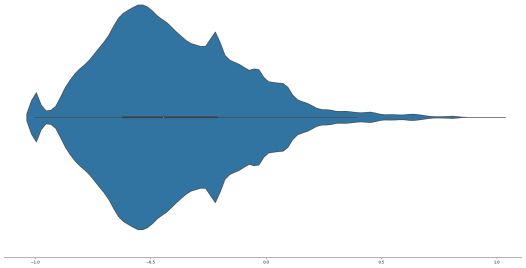


Figure 12: Losses for the stateful GAN using earth-mover's distance.

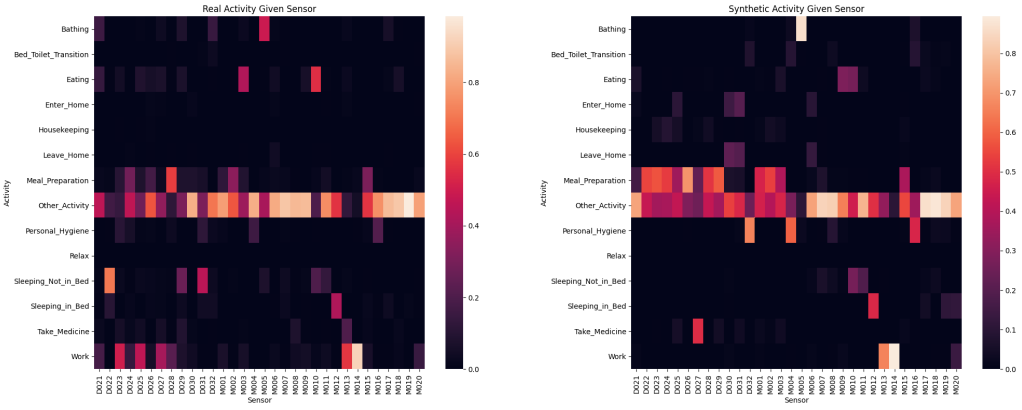


(a) Original



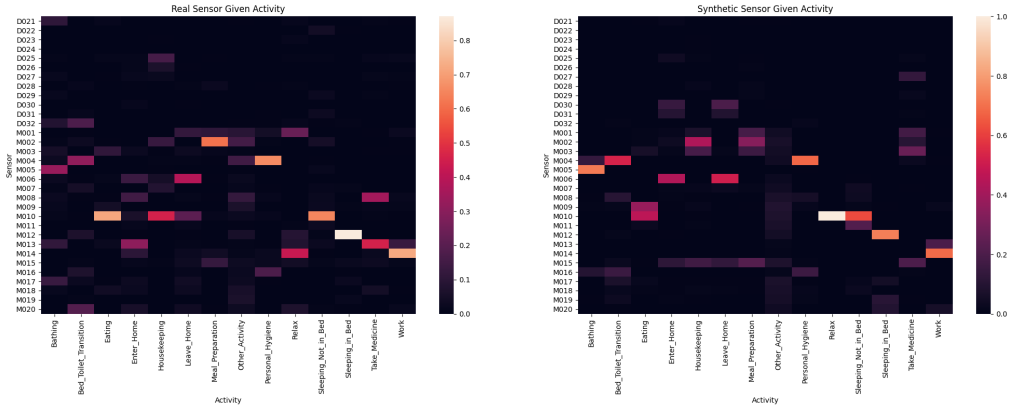
(b) Mapped

Figure 13: Time Differential Distributions



(a) Real (b) Synthetic

Figure 14: Heat maps showing joint probabilities of activity given sensor for the WGAN.



(a) Real (b) Synthetic

Figure 15: Heat maps showing joint probabilities of sensor given activity for the WGAN.

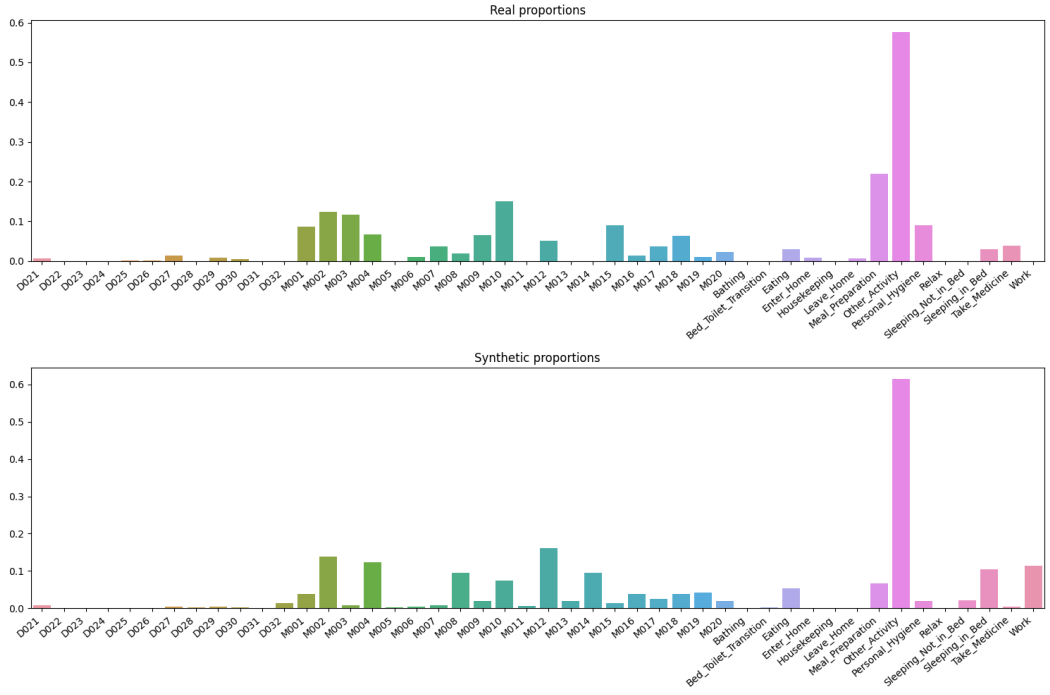


Figure 16: Histogram of data for real data (top) versus WGAN-generated data (bottom).