Synthetic Sensor Data Generation for Health Applications: A Supervised Deep Learning Approach

Skyler Norgaard*, Ramyar Saeedi+, Keyvan Sasani+, and Assefaw H. Gebremedhin+

Abstract— **Recent advancements in mobile devices, data analysis, and wearable sensors render the capability of in-place health monitoring. Supervised machine learning algorithms, the core intelligence of these systems, learn from labeled training data. However, labeling vast amount of data is time-consuming and expensive. Moreover, sensor data often contains personal information that a user may not be comfortable sharing. Therefore, there is a strong need to develop methods for generating realistic labeled sensor data. In this paper, we propose a supervised generative adversarial network architecture that learns from feedback from both a discriminator and a classifier in order to create synthetic sensor data. We demonstrate the effectiveness of the architecture on a publicly available human activity dataset. We show that our generator learns to output diverse samples that are similar but not identical to the training data.**

I. INTRODUCTION

Rapid advancements in mobile devices, data analysis, and sensors are revolutionizing healthcare services. Use of wearable devices and smart-homes has enabled the shift of healthcare services from hospital-care to remote in-home care [1], [2]. Remote health monitoring (RHM) is being used from preventive care to rehabilitation of patients with diabetes, heart failure, and asthma [3], [4].

Machine learning algorithms, the core intelligence of RHM systems, are generally supervised. Therefore, these algorithms require training data to construct models for data classification, alert generation, and patient/care giver engagement. Notwithstanding the enormous capacity of RHM systems, currently existing algorithms are mostly designed based on a limited amount of training data collected in small trials. Limitations and challenges associated with such data include: 1) data collections are performed in lab settings with specific protocols in order to be more accurate for the context at hand; 2) labeling a vast amount of sensor data is time-consuming and expensive; and 3) the data collected from users may contain sensitive information. Thus, there is the risk of compromising the privacy of users.

One way to address the above limitations is to supplement the available training data with synthetic labeled data. Several data generation methods have been proposed in recent years based on deep neural networks. Goodfellow et. al [5] proposed Generative Adversarial Networks (GANs) for the first time. Since their inception, GANs have found several applications in text and image generation [6], [7]. However, little work has been done to develop GANs for the purpose of generating multivariate time-series data. Alzantot et. al [8] developed an architecture for generating wearable sensor data that features a Gaussian Mixture Density Network connected to a recurrent neural network. However, the generator and the discriminator were trained separately, thus the generator did not actually learn from the feedback of the discriminator. Moreover, the data was not labeled, limiting its effectiveness for supervised learning tasks. Similarly, Esteban et. al [9] developed a recurrent generative adversarial network for generating medical time-series data. However, the training did not feature any form of supervision, making the training less stable and the approach less generalizable to other settings. Moreover, the generation of data for a particular class was only demonstrated with a "serialized" version of the MNIST handwritten digits dataset, leaving the task of generating labeled time-series data for a health context unexplored.

In this paper, we aim to address these limitations and develop a stable and generalizable method for generating synthetic sensor data. An adequate synthetic data generator needs to allot data in such a way that the generated data: 1) is difficult to distinguish from real data; 2) is not simply a duplicate of real training data; and 3) features diversity among each class.

GANs are known to be extremely difficult to train. The performance of the generator depends on the performance of the discriminator, and vice versa. This means that if one of the components converges too quickly, the performance of the other will suffer drastically. Thus finding the proper learning rate and training schedule can be like finding a needle in a haystack, and the configuration that balances the two in one setting will likely not be applicable in other settings. Moreover, maintaining a balance between the generator and discriminator does not necessarily mean that the training is effective. To truly tell, it is necessary to check how realistic the generated data is. Unlike image generation, time-series sensor data can be difficult for a human to assess visually, especially as the number of sensor channels increases.

We present a supervised generative adversarial network architecture. The network features three computational models: 1) the discriminator, which learns to detect whether the data is real or synthetic 2) the generator, which learns to generate realistic data and 3) the classifier, which is pre-trained with real data and provides supervised feedback to the generator.

+Saeedi, Sasani, and Gebremedhin are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, 99164. E-mails: {rsaeedi,ksasani,assefaw}@eecs.wsu.edu

*Norgaard is with the Department of Computer Science, Kalamazoo College, Kalamazoo, MI 49006, USA Email: skyler.norgaard14@kzoo.edu

978-1-5386-3646-6/18/$31.00 ©2018 IEEE

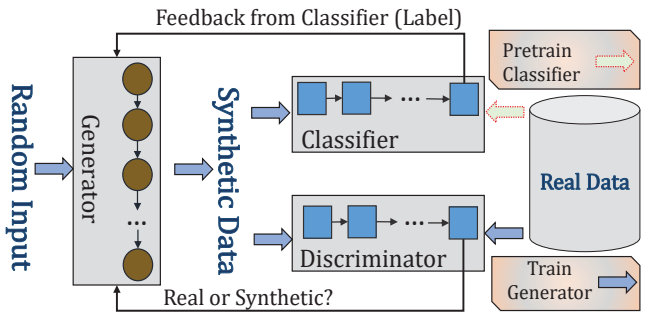


Fig. 1: High-level overview of the proposed Supervised Generative Adversarial Network structure.

We aim to solve the issue of training stability by updating the generator with feedback from both a discriminator and a pre-trained classifier. To further improve the stability, we train a generator for each class separately. Inspired by [9], we also provide easily-interpretable metrics for determining whether the generated data is actually realistic and how suitable it is for a given context. With these metrics, we attempt to check for two common problems: 1) the generator collapsing and generating only one sample 2) the generator simply memorizing or being too similar to the training data. In doing so, we hope to minimize the amount of visual assessment required during training while providing experts with as much information as possible to properly assess the generated data.

II. Supervised Generative Adversarial Network Architecture

In this section, we provide an overview of our proposed supervised GAN architecture, whose central idea is the collaborative training of a generator and a discriminator.

A. Generative Adversarial Networks

In a traditional GAN, there are two competing networks. The generator takes a vector of random values z as input and generates synthetic data while the discriminator takes both real and generated data as input and learns to discern between the two. The networks are trained simultaneously. The hope is that as the discriminator improves, the generator will learn to generate better samples, which will force the discriminator to improve, and so on and so forth. This competition can be represented by the minimax equation from Game Theory:

$$\min_G \max_D V(D,G) =$$

$$\mathbf{E}_{x\sim p_{data}(x)}[\log D(X)] + \mathbf{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

where V is the value function of the minimax game, D is the discriminator function, G is the generator function, z is a vector of random values, p_{data} is the prior probability distribution of the real data and p_z is the prior probability distribution of the random input vectors z.

B. Recurrent Neural Networks

Deep learning has been found effective in a multitude of tasks. However, a standard feedforward neural network makes the assumption that each input is independent. This is not true in the case of time-series data. To overcome this, recurrent neural networks (RNNs) were developed. In an RNN the activation of each time step depends on the activation from the previous time-step. To solve the vanishing or exploding gradient problem, Long Short Term Memory (LSTM) networks were developed. LSTMs have been found effective in recognizing activity from raw sensor data [10] and were used in both [8] and [9]. Thus we also employ them for our work.

C. Supervised Generative Adversarial Network

Here, we provide the architectural specifications of each component of our deep learning architecture as well as an overview of the algorithm that we employ. A high-level schematic of our proposed supervised GAN framework is provided in Fig. 1.

1) Architecture: The Generator (G), Discriminator (D) and Classifier (C) all feature a recurrent LSTM layer and a feedforward layer. All LSTM layers use the hyperbolic tangent function as activation. The number of units in the LSTM layers of G, D and C are 128, 100, and 100 respectively. The feedforward layer of G has 3 units corresponding to the three sensor channels. The feedforward layer of D has 1 unit and uses the sigmoid function as activation. This function outputs the probability that the given segment is real. The feedforward layer of C employs the soft-max activation function, which outputs the most probable class that the segment belongs to. Since LSTM networks tend to become more informed over time, we make our prediction using the activation from the last time step of both the discriminator and classifier.

2) Training: Here, we present the way in which we train C, G and D. To train C, we minimize the categorical cross-entropy loss function. To train D, we minimize the log loss function. We simultaneously train G by maximizing the log loss function of $D(G(z))$ and minimizing the categorical cross-entropy loss of $C(G(z))$. This can be done practically by flipping the labels of the real and synthetic data and then minimizing $D(G(z))$ and $C(G(z))$ instead.

3) Algorithm: The pseudocode of the algorithm is outlined in Algorithm 1. The algorithm takes as input the full database of labeled data $DB = \{X_1, X_2, \cdots X_n\}$ where X_j is the set of data for label j, the set of labels $L = \{l_1, l_2 \cdots l_m\}$ where each $l_i \in L$ corresponds with instances in $X_i \in DB$, the pre-training validation threshold p_{th}, the adversarial training validation threshold a_{th}, an initialized classifier C, an initialized generator G_0, an initialized discriminator D_0 and a batch size k. As output, the algorithm returns a trained generative model for each class. For training each component, the algorithm calls functions TrainC, TrainD, and TrainG, which train the respective models in the way described in the previous section. In the algorithm, the classifier is first trained until the validation accuracy is

Algorithm 1 SUPERVISED GENERATIVE ADVERSARIAL NETWORK

Input: DB, L, p_{th}, a_{th}, C, G_0, D_0, k
Output: $G_1, G_2, \cdots G_n$

$p_{acc} \leftarrow 0$
while $p_{acc} < p_{th}$ **do**
  $C \leftarrow$ TRAINC(DB, L, C) /*update C*/
  $p_{acc} \leftarrow$ accuracy of C
end while
for $i = 1 \ldots n$ **do**
  $a_{acc} \leftarrow 0$; $G_i \leftarrow G_0$; $D_i \leftarrow D_0$
  while $a_{acc} < a_{th}$ **do**
    $X_{synt}, L_{synt} \leftarrow$ generate k instances with $G_i(z)$
    $X_{real}, L_{real} \leftarrow$ choose k instances from $X_i \in DB$
    $D_i \leftarrow$ TRAIND$(\{X_{synt}, X_{real}\}, \{L_{synt}, L_{real}\}, D_i)$
    $G_i \leftarrow$ TRAING$(G_i(z), L_{synt}, L_i, G_i)$
    $a_{acc} \leftarrow$ accuracy of C for generated data $G_i(z)$
  end while
end for
return $G_1, G_2, \ldots G_n$

above a certain threshold. For each class, a generator is then trained until the classification accuracy of the generated data $G(z)$, where z is a random input vector, goes above a given threshold.

III. EXPERIMENTAL VALIDATION

To evaluate our supervised generative adversarial algorithm, we used a publicly available human activity recognition (HAR) dataset. The algorithm was implemented in Python. In the remainder of the section, we describe the dataset and present the results of our algorithm.

A. Dataset Description

We use the publicly available Daily and Sports Activities dataset [11]. The dataset features eight subjects (4 male, 4 female) who perform 19 different activities for 5 minutes each. The 5 minute segments are then divided into 5 second windows. This results in 480 samples per activity. To compute our data evaluation metrics, we generate 100 synthetic samples per class. For our task, we select 9 sports activities which are shown in the leftmost column of Table I. We use the data from the accelerometer placed on the subjects' right leg. We then normalize the data so that all values fall in the range $(-1, 1)$.

B. Training Configurations

We first pre-train the HAR classifier until we reach 0.97 F1-score. For the adversarial training, we train the generator with the Adam optimizer with learning rate 0.001 and train the discriminator with the SGD optimizer with learning rate 0.1. We use a batch size of 25 for both configurations and employ dropout of 0.5 before each layer in both the generator and discriminator. For each label, we train a generator until the pre-trained HAR classifier reaches 80% validation accuracy. We choose 80% because we found that the diversity of generated samples begins to decrease after this point.

TABLE I: Average Real to Real (RTR), Real to Synthetic (RTS), Synthetic to Synthetic (STS) Similarity, and Maximum RTS (Max-RTS) for each activity.

Activity	RTR	RTS	STS	Max-RTS
Walking	0.392	0.230	0.904	0.402
Walking (incline)	0.461	0.593	0.896	0.745
Running	0.220	-0.016	0.546	0.162
Stepper	0.507	0.478	0.605	0.774
Cross-trainer	0.456	0.425	0.982	0.670
Cycling (horizontal)	0.292	0.191	0.134	0.566
Cycling (vertical)	0.631	0.597	0.689	0.919
Rowing	0.365	0.186	0.179	0.672
Jumping	0.208	0.005	0.442	0.293
Average	0.392	0.299	0.598	0.578

C. Assessing Generated Data

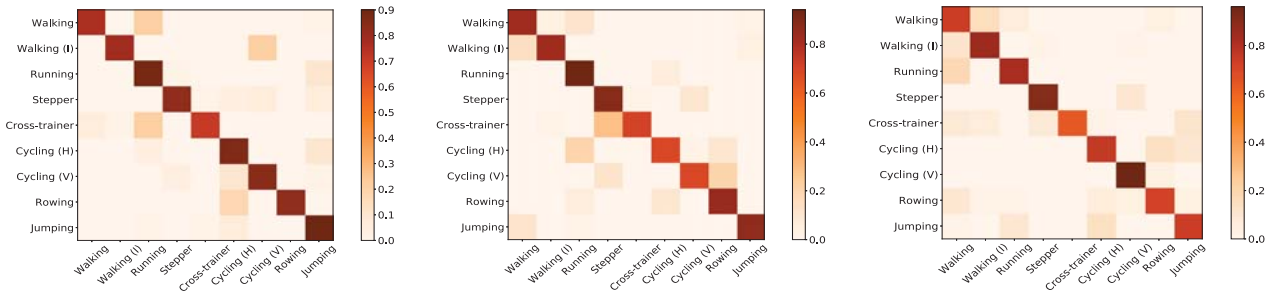
Assessing the quality of synthetic multivariate time-series data is a challenging task since there is not currently a widely-accepted mathematical evaluation framework. Moreover, determining if the data is satisfactory is subjective and depends heavily on the context. For example, highly similar synthetic data may not be a concern when we have limited labeled data and thus make use of the generator to increase the size of the dataset, but would be problematic in a setting where we employ the generator to avoid utilizing data containing sensitive information. Here, we present 4 easily interpretable, computationally efficient, and adjustable metrics which would provide experts with the information necessary to properly analyze the suitability of synthetic data for their particular problem.

1) Synthetic to Synthetic Similarity: We compute the Synthetic to Synthetic (STS) similarity to better understand the diversity of the generated data for the given class and to ensure that the generator is not collapsing. We do this by selecting a random synthetic segment and calculating its average cosine similarity to 5 other synthetic segments. We limit the number of similarities calculated because, during training, we are concerned with checking for generator collapse rather than extensively studying the distribution of similarities between generated instances. However, the number of similarities could be increased if that information was desired.

2) Real to Synthetic Similarity: To compute the Real to Synthetic (RTS) similarity, we calculate the average cosine similarity between each generated segment and 10 random real segments. The RTS value is used in conjunction with the Real to Real Similarity value which is described below.

3) Real to Real Similarity: To better interpret the RTS value, we also compute all possible Real to Real (RTR) similarities and take the average. Since the real data does not change during training, we can pre-compute this value and thus further reduce the number of computations done in real time. Ideally, we want the RTR and RTS values to be similar since this would suggest that the real and synthetic data come from similar distributions (although cases more complicated than this can also arise).

4) Maximum Real to Synthetic Similarity: At the end of training, we also compute the maximum similarity (Max-



(a) Trained on subject 1 (b) Trained on subject 2 (c) Trained on subject 3

Fig. 2: Performance of pre-trained classifier on synthetics data for different subjects.

RTS) between all generated and real segments. This provides us with an understanding of how similar our generated data is to the real data in the most similar case, and would be valuable in ensuring that the privacy of users is protected since certain instances could be omitted if they were above some similarity threshold. In the case where privacy is not important, this metric can be used to ensure that the generator has actually learned to generate data and has not learned to simply copy the training data.

In Table I, we see that the generator is not simply memorizing the training set (Max-RTS never equals 1), the average RTS and RTR values across all activities are similar, and, on average, the generator is producing diverse samples. We also see that a lower STS tends to result in a larger difference between RTR and RTS. Lastly, we see that our system struggles most with generating similar samples for the Running and Jumping classes, suggesting that more dynamic activities may be more difficult to replicate.

D. Classifier Performance

To better understand the classifier errors and the ambiguity of certain generated samples, we present heat maps (corresponding to a confusion matrix) for three subjects in Fig. 2. Across all subjects, the precision and recall values range from $59.7\% - 100\%$ and $64\% - 96\%$ respectively. We see that there is no apparent pattern in the classifier errors. This suggests that our generator is outputting diverse samples that are ocassionally misclassified rather than systematically failing for certain classes. On the diagonal, we see that the accuracy for the different classes is between 80% and roughly 95%. We do not necessarily want 100% classification accuracy because it will likely mean that the generator overtrained for the given classifier. It is also important to note that although we chose 80% as the a_{th} value from Algorithm 1, certain classes have higher a_{acc} values because of a large increase in classifier accuracy between the second to last and last training iteration.

IV. Conclusion

In this work, we proposed a novel supervised GAN for generating labeled sensor data. We also provided simple yet informative metrics for analyzing the generated data. We showed that our generated data is diverse yet similar to the real data. Overall, we provided a generalizable framework for producing and assessing sensor data which allows for the integration of expert knowledge and could be applied to a variety of RHM tasks. We evaluated the effectiveness of our approach with a publicly available human activity dataset. In future work, we plan to further experiment with GAN architectures and also plan to test the effectiveness of the generated data for transfer learning tasks.

Acknowledgment

This work was supported in part by the National Science Foundation award IIS 1553528.

References

[1] J. H. Abawajy and M. M. Hassan, "Federated internet of things and cloud computing pervasive patient health monitoring system," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 48–53, 2017.

[2] A. Shahrokni, S. Mahmoudzadeh, R. Saeedi, and H. Ghasemzadeh, "Older people with access to hand-held devices: Who are they?" *Telemedicine and e-Health*, vol. 21, no. 7, pp. 550–556, 2015.

[3] M. Elgendi, A. Al-Ali, A. Mohamed, and R. Ward, "Improving remote health monitoring: A low-complexity ECG compression approach," *Diagnostics*, vol. 8, no. 1, p. 10, 2018.

[4] M. M. Davis, M. Freeman, J. Kaye, N. Vuckovic, and D. I. Buckley, "A systematic review of clinician and staff views on the acceptability of incorporating remote monitoring technology into primary care," *Telemedicine and e-Health*, vol. 20, no. 5, pp. 428–438, 2014.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *ArXiv e-prints*, Nov. 2015.

[7] H. Wang, Z. Qin, and T. Wan, "Text Generation Based on Generative Adversarial Nets with Latent Variable," *ArXiv e-prints*, Nov. 2017.

[8] M. Alzantot, S. Chakraborty, and M. B. Srivastava, "Sensegen: A deep learning architecture for synthetic sensor data generation," *CoRR*, vol. abs/1701.08886, 2017. [Online]. Available: http://arxiv.org/abs/1701.08886

[9] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs," *ArXiv e-prints*, Jun. 2017.

[10] R. Saeedi, S. Norgaard, and A. H. Gebremedhin, "A closed-loop deep learning architecture for robust activity recognition using wearable sensors," in *Big Data (Big Data), 2017 IEEE International Conference on*, 2017, pp. 473–479.

[11] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recogn.*, vol. 43, no. 10, pp. 3605–3620, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2010.04.019