Data Augmentation of IMU Signals and Evaluation via a Semi-Supervised Classification of Driving Behavior

Amani Jaafer[1], Gustav Nilsson[2], and Giacomo Como[3]

***Abstract*— Over the past years, interest in classifying drivers' behavior from data has surged. Such interest is particularly relevant for car insurance companies who, due to privacy constraints, often only have access to data from Inertial Measurement Units (IMU) or similar. In this paper, we present a semi-supervised learning solution to classify portions of trips according to whether drivers are driving aggressively or normally based on such IMU data. Since the amount of labeled IMU data is limited and costly to generate, we utilize Recurrent Conditional Generative Adversarial Networks (RCGAN) to generate more labeled data. Our results show that, by utilizing RCGAN-generated labeled data, the classification of the drivers is improved in 79% of the cases, compared to when the drivers are classified with no generated data.**

***Index Terms*— IMU sensor, driving behaviors, data generation, data evaluation.**

I. Introduction

Modern vehicles are equipped with an increasing number of sensing devices, such as Global Positioning System (GPS), Inertial Measurement Units (IMU), and other sensors that communicate through the Controller Area Network (CAN-Bus). This real-time sensed data can be used to detect, analyze, predict, and plan a large variety of issues such as traffic congestion, vehicle energy consumption and emissions, urban mobility, and drivers' behavior. Multiple approaches have been developed and applied to accurately identify driving behavioral patterns, such as driver recognition [1], [2], maneuver recognition [3], [4], and aggressive driving detection [5]. While an accurate classification of the driving behavior can contribute to a better driving experience for the driver, there are also other applications where such classification can be useful. Recently, there are been a growing interest from car insurance companies in designing driver behavior classification systems that could eventually be used to relate their costumers' fees to how they drive. As a part of this solution, it is of interest to accurately classify the level of aggressiveness of their customers' recorded trips. Nevertheless, the large number of trips would not allow to explicitly identify the type of driving for each individual trip. Consequently, several works such as [4], [6], and [7] have been conducted to solve this problem by an unsupervised learning approach. In the mentioned work, the goal is to find clusters from the recorded trip data which can be characterised by different levels of the aggressiveness without relying on the labels.

Since the labels, i.e., the driving style, remain a crucial element in order to apply a supervised algorithm, generating realistic artificial data can be an alternative to increase the size of the training or validation datasets and possibly improve the quality of the classification. Semi-supervised learning is motivated by the availability of large datasets with unlabeled features in addition to labeled ones, and has previously been utilized in different applications [8], [9], [10]. This lack of labeled data can be efficiently addressed through a deep learning pipeline.

Another application of interest for driving behavior classification is the development of autonomous vehicles. A better understanding of how humans drive can indeed allow for both a better functioning on a technical level and, of course, minimizing as much as possible any error, in view of the security of the users. Identifying aggressive drivers is crucial in developing safer autonomous driving techniques and advanced driving assistant systems. This problem has been extensively studied over the past decades in several works [11], [12], [13]. Current autonomous driving systems use a wide range of algorithms to process sensor data. Some work, as [14], uses end-to-end approaches to make navigation decisions from the sensor inputs such as camera images, LIDAR data, etc. A variety of sensors can be useful for cars to extract important information to improve the quality of autonomous vehicles and to learn how to drive safely and efficiently. Nevertheless, data collection can also be expensive and restricted in terms of privacy. Simulating data and exploiting it in the same context as real ones appears as a solution to study. The attention to generative models is increasing due to their capability of modelling underlying patterns in multidimensional data. However, assessing the quality of the synthetic data remains a crucial point to validate.

In this paper, we formulate the problem of generating labeled IMU signals, representing aggressive and normal drivers, of one-minute length for a specific part of a road, using Recurrent Conditional GANs. The generated data will

[1]A. Jaafer is with the Division of Systems Analysis and Economics, Royal Institute of Technology (KTH), Stockholm, Sweden. jaafer@kth.se

[2]G. Nilsson is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. gustav.nilsson@gatech.edu

[3]G. Como is with the Department of Mathematical Sciences, Politecnico di Torino, Italy and the Department of Automatic Control, Lund University, Sweden. giacomo.como@polito.it

This work was partially supported through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA, as well as by MIUR grant Dipartimenti di Eccellenza 2018–2022 [CUP: E11G18000350001], the Swedish Research Council [2015-04066], and Compagnia di San Paolo.

An extended version of this paper is available at http://arxiv.org/abs/2006.09267

be practically assessed based on its capacity to improve the classification of the semi-supervised framework.

II. Related work

Since obtaining real sensor data can be costly, time-consuming, and have privacy issues, there have recently been several studies on sensor modelling for virtual testing, e.g., in [15], [16], [17], which are mostly based on parametric models. In [15], a non-parametric statistical model was developed allowing for the generation of sensor position output. In [16] a radar model is proposed where noise is added to the raw signals, and then filtering is applied to model sensor output. Further, [17] proposed a Variational Autoencoder (VAE) approach in order to model the radar sensor output given some input vector, using object lists and spatial rasters.

Generative Adversarial Networks (GANs) [18] have proven to perform well in generating different types of data. Different research works, from computer vision [19], [20] to natural language processing [21], had shown that the application of this kind of generative models can provide good results. In [22] a Recurrent Conditional Generative Adversarial Network (RCGAN) has been proposed for modelling real-valued time series describing sensor outputs that are used in autonomous driving applications. In [23], the authors augmented the LiDAR sensor data in simulated environments, by employing CycleGANs. Evaluating GANs is a challenging task. Some would rely on a visual assessment by having appealing results that agree with the real distribution. The latter shows a high potential for some data, especially for images. Meanwhile, when time series are inspected visually, this remains an inconsistent method, since it is based on a manual operation to inspect each generated sample. By evaluating a convenient distance metric between the real and fake data distribution, we can assess the trained model and infer how much the model is capturing temporal patterns. Quantitative measures, such as reconstruction loss, Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence can be combined with visual assessment to provide a robust assessment of GAN models. A quantitative extrinsic approach like in [23] and [24] is also an alternative, which mainly relies on an external method to measure the quality of the generated data.

III. Contribution

This paper makes two main contributions to the field of driving behavior classification. First, it addresses the problem of data augmentation of car sensors. In our study, we generate IMU signals of one-minute length in a common portion of the road characterised by the type of driving style, which is either aggressive or normal. We use Recurrent Conditional GANs for the generation of these labeled time series. Second, we build a framework to evaluate the quality of generated data from a practical perspective. In other words, we assess the quality of the data based on the improvement of a semi-supervised model, which identifies the type of driving, by adding different percentage of synthesised data to the classifier's training and/or validation sets. Consequently, the paper investigates how much data should be generated and in which set should be used, to improve the accuracy of the driving behavior's classification.

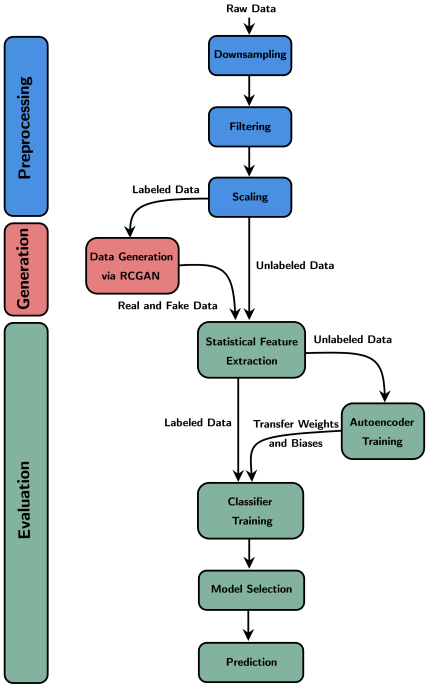

Fig. 1. The overall framework used for classifying the drivers, including data preprocessing, data generation and evaluation through the semi-supervised approach.

IV. Approach

In this section, we firstly present the experimental setting used to collect the labeled data. Then, we present the data preprocessing tasks, followed by the generative model used to synthesize the multidimensional time series. Finally, we present the extrinsic assessment framework used to evaluate the generated data. The entire pipeline is shown in Fig. 1.

The dataset being used in this paper was collected from a vehicle simulator. The experiment consisted of 40 drivers driving separately using different cars, in the same circuit. The drivers had been asked to drive both in a normal and in an aggressive way. By doing so, we have close to a ground-truth about which recorded trips that are normal or aggressive. The simulator was collecting the same signals as a real IMU unit, i.e., longitudinal acceleration, lateral acceleration, pitch, yaw, and roll. All the signals had the same sampling frequency, namely 1000 Hz. In total, the dataset consists of $n = 238$ simulation drives.

A. Data preprocessing

For computational reasons, we down-sampled the IMU signals to 1 Hz, by taking each 1000^{th} observation. Although this down-sampling is done mostly for computational convenience, it is also very likely that in practical applications the hardware will have a more limited sampling frequency compared to the simulator. Since the signals may contain

artifacts, we filtered them by applying a moving average filter with a sliding window of ten samples. We limited our study only on the first one minute of each trip, both for computational reasons, but also since in practical applications, it would be favorably to classify the driver without too much history. A similar choice time-window has previously been suggested in [5].

All features were normalized using a MinMax scaler. Our dataset was split into the labeled data used in training the RCGAN and the unlabeled one used in the semi-supervised part. The RCGAN was trained only on a dataset of 60 trips, equally balanced between aggressive and normal.

B. Data generation

RCGANs were originally developed and implemented in [24] for medical applications. Our paper was inspired from this work to synthesize IMU signals for a normal and an aggressive trip. We will start this section by give a brief introduction to Recurrent Neural Networks (RNN). Next, we will introduce long short-term memory RNNs, which is an extension of the RNN framework. This subsection ends with a description of the RCGAN model, which is using long short-term memory RNNs.

1) Recurrent Neural Networks: RNNs are mostly used for sequential modeling and learning. They process one element of input data at a time t and implicitly store previous information using cyclic connections of hidden units. Given a sequence of vectors, $x = (x_1; :::; x_T)$, where $x_t \in \mathbb{R}^{d_{\text{in}}}$, the RNN outputs a representation, that is a sequence of vectors $h = (h_1; :::; h_T)$, where $h_t \in \mathbb{R}^H$. The sequence h is determined iteratively through:

$$h_t = g(Wx_t + Uh_{t-1} + b); \quad 8t \in \{1; :::; T\}; \tag{1}$$

where $W \in \mathbb{R}^{H \times d_{\text{in}}}; U \in \mathbb{R}^{H \times H}; b \in \mathbb{R}^H$, and $h_0 = \mathbf{0}$. The function g is a non-linear mapping and often chosen as $\tanh$ applied component-wise.

The output vector $p_t \in \mathbb{R}^{d_{\text{out}}}$ transforms the current hidden state $h_t \in \mathbb{R}^H$ in a way that depends on the final task. For classification, it is computed as

$$p_t = \text{softmax}(W_p h_t + b_p): \tag{2}$$

Note that $W \in \mathbb{R}^{H \times d_{in}}; U \in \mathbb{R}^{H \times H}; b \in \mathbb{R}^H; W_p \in \mathbb{R}^{d_{\text{out}} \times H}; b_p \in \mathbb{R}^{d_{\text{out}}}$ are network parameters determined through gradient descent. The scalars $H; d_{\text{in}}$ and d_{out} are the dimensions of the hidden layer, the input, and the output, respectively. For example, in the case of 2-category classification, $d_{\text{out}} = 2$ and the probability vector p_t refers to the probabilities of each input element x_t belonging to each category.

2) Long Short-Term Memory (LSTM): In practice, the vanilla RNN encounters numerical computation difficulties. One reason presented in [25] is that it would cause the gradient to vanish and explode while computing the back-propagation through time, on data with long term dependencies. The vanilla RNNs only consider short term dependencies. The Long Short-Term Memory (LSTM) technique was therefore introduced to mitigate this kind of risk. The latter incorporate a memory cell c together with an input gate i, an output gate o and a forget gate f. The memory cell enables the network to remember its state over time, and by doing so it is possible for the full network to capture long-term temporal dependencies present in the training data. The evolution of LSTM states are determined by:

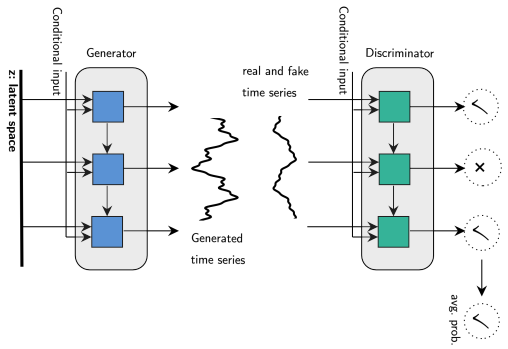

Fig. 2. Architecture of RCGAN composed of an RNN generator (bottom) and RNN discriminator (top). Both RNNs in the generator and discriminator, respectively, are LSTM based. The generator takes input from a latent space as well as a conditional input at each time frame. The discriminator takes either a real or fake time series together with the conditional input as input at each time frame.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1} + b_o) \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

where $W; V; U$ and b are learnable parameters. The function σ denotes sigmoid activation function, that is applied element-wise. The quantities i_t, f_t, and o_t stand for the input, forget and output gates respectively. The output of the LSTM cell is o_t and $\odot$ denoting point-wise vector products, i.e., Hadamard product.

3) Recurrent Conditional Generative Adversarial Networks: RCGANs are generative recurrent neural networks that aim at generating real-valued time series subject to a conditional information. In the RCGAN architecture there are two different LSTM-RNNs trained simultaneously, a generator G and a discriminator D, which have conflicting objectives. The generator learns over the training data, whereas the goal of the discriminator is to discriminate between the synthetic data generated by G and the real data, as depicted in Fig. 2. We denote by $k; l$ and m the feature dimensions of the data, the conditional information and the latent/noise space, respectively. Let L be the length of the time series and $x_t \in \mathbb{R}^k; y_t \in \mathbb{R}^l; z_t \in \mathbb{R}^m$.

In practice, the min-max game problem is described as

$$\begin{aligned} &\min_G \max_D \mathbb{E}_{x \sim p_{\text{real}}(x)}[\log(D(x|y)] \\ &+ \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y))]; \end{aligned} \tag{8}$$

where p_{real} is the distribution of the real data and p_z is a prior distribution over the input noise variables. These latter,

i.e., the m sequences of L points, are drawn independently from $N(0;1)$.

In our case, the input consists of five-dimensional time series data, i.e., the signals for $X = 60$ trips, with a binary condition attributing the type of driving (normal/aggressive). The length of all time series is equal to 60. More details about the architecture of our trained RCGAN are parented in the extended version of the paper[1]. Our RCGAN generates an example from a specific class. In other words, if we ask for the aggressive class, the generator produces one aggressive trip. Thus, after training the model, the number of generated trips per each class, should be defined. We fed the RCGAN with the training set, and then generated $0.5X$; $1X$; and $1.5X$ new trips from the data.

C. Data evaluation

In order to evaluate the quality of the RCGAN model, we used a semi-supervised framework to classify whether a trip is aggressive or normal. Firstly, we extracted statistical features from the real and fake data. Nine statistical features were calculated out of the five time series to measure different properties of that variable, namely: mean, median, mode, standard deviation, skewness, kurtosis, 25 percentile, 75 percentile and interquartile range. A further description of a few of the statistical features is given below.

We denote by $t = (t_1; \ldots; t_L)$ a real-valued series with and its mean and standard deviation, respectively.

1) Mode: The mode is the most frequently appeared value in the serie.

2) Skewness: Skewness is used to measure the asymmetry of the data. Let $^{(n)}$ be the nth moment, i.e.,

$$^{(n)} = \frac{1}{L}\sum_{i=1}^{L}(t_i \quad)^n : \tag{9}$$

The skewness is then calculated with the third moment as $s = \frac{^{(3)}}{^3}$.

3) Kurtosis: Kurtosis is used to measure the peakedness of the probability distribution of the data and calculated as $k = \frac{^{(4)}}{^4}$, where $^{(4)}$ is the 4^{th} moment

4) Percentiles: A percentile is the value of a variable below which a certain percent of observations fall. In other words, the p^{th} percentile is a value l such that at most $(100 \quad p)\%$ of the measurements are less than this value and $100 \quad (1 \quad p)\%$ are greater.

5) Interquartile Range: Interquartile Range (IQR) is a measure of statistical dispersion. It is defined as the difference between the 75^{th} and the 25^{th} percentiles, called the upper and lower quartiles.

The unlabeled part of the dataset were used for training an Autoencoder (AE), see Fig. 1, in order to transfer its weights and biases to the DNN classifier. The Autoencoder is a neural network, which aims to reconstruct the input, i.e., the target output is the input. It is composed of two main parts, an encoder that serves to compress the data in a lower dimensional space and a decoder which reproduces the input out of the bottleneck. The AE is trained in order to minimize the error between the real input and the constructed one. More formally, let s be the input, where $s \in \mathbb{R}^d$, the compressed representation $q \in \mathbb{R}^p$, $(p < d)$ mapped by g ,

$$q = g(s) = (Ws + b); \tag{10}$$

where , W, and b, are respectively the activation function of the encoder, the weight matrix, and the bias vector. The function g is parameterized by $= \{W; b\}$. The decoder part reconstructs the input from the hidden representation q by the function f ,

$$s' = f(q) = (W'q + b'); \tag{11}$$

where , W', and b' are respectively the activation function of the decoder, the weight matrix, and the bias vector. f is parameterized by $= \{W'; b'\}$.

Each training input vector $s^{(i)}$ is mapped to a corresponding $q^{(i)}$ which is then mapped to a reconstruction $s'^{(i)}$ such that $s^{(i)} \quad s'^{(i)}$. The parameters and of the model are optimized to minimize the average reconstruction error such that

$$\begin{aligned}(\quad ; \quad) &= \underset{;}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L(s^{(i)}; s'^{(i)}) \\ &= \underset{;}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L(s^{(i)}; f(g(s^{(i)}))); \end{aligned} \tag{12}$$

with L the loss function and is given by $L(x;y) = \|x \quad y\|_2^2$.

After training the Autoencoder on the unlabeled dataset, i.e., the 178 trips from the simulator that was not used to train RCGAN, we use the weights and biases to initialize a supervised deep neural network (DNN) model and then fine-tune the DNN model using the labeled dataset to classify the type of driving. To measure how generated data can improve the data classification, we run various groups of experiments. In the first group which is our baseline, the classifier was trained and validated using only the real labeled dataset. In the following groups, we made all the combination of the training and the validation sets containing labeled real/fake/real+fake datasets. All the classifiers were trained only with the selected features.

We evaluate the classifier's performance by measuring the Area Under Receiver Operating Characteristic (AUROC). This criterion is one of the most widely used metric to score the goodness of a predictor in a binary classification task. It ranges in value from 0 to 1. The higher the AUROC, the better the classifier is at predicting the classes, which is the type of driving in our case. The AUROC is computed on the test set containing all the real data.

V. Results and Discussion

In order to investigate the quality of the generated fake data and see whether it can be useful on a practical level, we applied our semi-supervised framework as an extrinsic evaluation. The generated fake data were used in both the training and the validation set of the classifier. All combinations of real and generated fake data is covered in Table II.

[1] http://arxiv.org/abs/2006.09267

TABLE I

PERFORMANCE OF THE CLASSIFIER

Training Set	Validation Set	Ratio Fake/Real	AUROC[1]
R[2]	R	0%	0.823
R + F[3]	R + F	50%	**0.858**
		100%	**0.851**
		150%	0.805
R	F	50%	**0.846**
		100%	0.774
		150%	**0.845**
F	R	50%	0.799
		100%	**0.858**
		150%	**0.851**
R + F	R	50%	**0.851**
		100%	**0.832**
		150%	**0.844**
F	F	50%	0.776
		100%	**0.846**
		150%	**0.833**
R	R + F	50%	**0.841**
		100%	**0.851**
		150%	0.813
F	R + F	50%	**0.841**
		100%	0.805
		150%	0.805
R + F	F	50%	**0.849**
		100%	0.805
		150%	**0.841**

[1] This measure is computed on the test set, containing all the real data, namely the 238 trips.
[2] The set R consists of the 60 real trips which were used to train the RCGAN.
[3] The set F consists of the fake data which were generated from the RCGAN.

We ran the experiments 200 times. After each trained RCGAN, we generated different amount of data and we utilized them in the validation or training set of our classifier. In 79% of the simulations the RCGAN reached at least an AUROC strictly higher than the baseline value, for at least one combination of real and fake data in every of the 200 runs. Table II shows the performance of the classifier of the semi-supervised framework, trained and validated on different sets consisting of combinations of real and fake data for a simulation outperforming the baseline. AUROC is measured on the test set which contains all the real trips. Bold depicts the AUROC superior to the baseline values. We can see that for most simulations the AUROC exceeds the baseline, for a variety of sets and ratios of real and generated fake data.

Since we varied the percentage of real and generated fake data in both training and validation sets of the classifier, it is of interest of how much generated fake data that is needed and how it should be utilized by the classifier. Table III highlights the summary over the set of simulations which outperform the baseline, i.e., the number of recorded AUROCs that exceeds the baseline value, for each combination set and ratio fake.

On a first glance, we can divide the Table III into three groups. First one containing the combination set which have the highest total number. Training on the real, while validating on both the real and the fake seems to be the best option in order to ensure a better classification.

TABLE II

SET OF SIMULATIONS WHICH OUTPERFORM THE BASELINE

Sets		Counts[1]			AUROC	
Training	Validation	50%	100%	150%	Mean	SD
R+F	R+F	21	14	14	0.834	0.009
R	F	76	76	86	0.843	0.007
F	R	3	1	2	0.833	0.004
R+F	R	24	14	15	0.833	0.008
F	F	0	1	0	0.830	–
R	R+F	106	101	104	0.840	0.008
F	R+F	2	0	1	0.835	0.003
R+F	F	12	26	17	0.835	0.009

[1] The recorded number for each sets and percentage of fake data.

Training on the real data and validating only on the fake data can be also a good way to use the generated data. The second group contains the following combinations; training on the real data and fake data, while validating on the real data, training and validating on both the fake data and real data, and lastly training on real data and fake data while validating on the fake data. This group is characterised by a lower number of records comparing to the first one. This underlines the fact that incorporating the fake data in the training set is less likely to improve the classifier. The third group contains the remaining combinations. This group is characterised by the fact that the training set is only composed of fake data. The negligible number of this group excludes the possibility of using only the fake data to improve the classifier accuracy. This result can be justified by the fact that the generation of data is done on the basis of the real ones, therefore substituting the content of the classifier's training set from real data to fake data, would not guarantee an improvement. The generative model had to learn from the real data to end up having new ones close enough to the original, but still different.

Fig. 3 shows that the classifier can perform well by training merely on the generated fake data. By training and validating on the fake data, we can have an AUROC slightly lower than the baseline, which still guarantees a good prediction of the type of driving.

The first group also reached the highest average of AUROC between its elements, comparing to the other ones. Consequently, we capture the importance of incorporating the fake data only into the validation set.

On the other hand, we want to see whether the size of the generated data would affect the performance of the classifier. Since we know from the previous results, in which combination sets the fake data worth to be used, we limited the scope only on the first group, which only train the classifier on the real data. We can see in Table III, that increasing a ratio fake to 150% would give in overall, higher chances to improve the model. In this case, it means that synthesising more data than the size of the original one, can give a better classification of driving behavior.

VI. CONCLUSION

In this paper, we outlined our experiences of using Recurrent Conditional GANs for generating IMU signals, which