MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

GNSS Multipath Detection Aided by Unsupervised Domain Adaptation

Zawislak, Remy; Greiff, Marcus; Kim, Kyeong Jin; Berntorp, Karl; Di Cairano, Stefano; Mao, Konishi; Parsons, Kieran; Orlik, Philip V.; Sato, Yuki

TR2022-118 October 04, 2022

Abstract

This paper concerns the problem of multipath detection in global navigation satellite system (GNSS) positioning. An unsuper- vised machine learning (ML) approach is developed using a convolutional neural network in an autoencoder framework with k-means clustering in the latent space. Such methods often rely on large amounts of annotated data during training, which are difficult to collect in the context of GNSS applications. To this end, a joint approach is proposed to train the ML method on synthetic data generated by hardware in the loop (HIL) simulations, and use unsupervised domain adaptation (UDA) to reduce any discrepancies between real and simulated data. For this purpose, an UDA facilitated by a cycle-consistent generative ad- versarial network is proposed. It is shown that the proposed method can significantly improve the multipath detection accuracy on experimental data compared to baseline approaches (including both heuristic approaches and ML-methods). It is also shown that UDA can enhance GNSS estimation performance in the absence of annotated training data, achieving a prediction accuracy in hand-labeled experimental data of as much as 99 %.

ION-GNSS+ Conference 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

GNSS Multipath Detection Aided by Unsupervised Domain Adaptation

Rémy Zawislak[1], Marcus Greiff[2], Kyeong Jin Kim[2], Karl Berntorp[2], Stefano Di Cairano[2], Mao Konishi[3], Kieran Parsons[2], Philip V. Orlik[2], and Yuki Sato[3],
[1]*Apple, USA*, [2]*Mitsubishi Electric Research Labs, USA,* [3]*Advanced Technology R&D Center, Mitsubishi Electric, Japan.*

BIOGRAPHY

Rémy Zawislak is a GNSS Firmware Engineer at Apple Inc. He has a MS degree in Aeronautics and Astronautics from Stanford University, and his research interests include machine learning and estimation related to GNSS positioning.[1]

Marcus Greiff is a Visiting Research Scientist at Mitsubishi Electric Research Laboratories, where he works on estimation and control. He has a PhD. in Systems and Control from Lund University. During his PhD he worked on nonlinear control for aerial vehicles and estimation theory, for which he has been awarded multiple paper awards at the IEEE CCTA conference.

Kyeong Jin Kim is a Senior Principal Research Scientist at Mitsubishi Electric Research Laboratories. He works on transceiver design, performance analysis of spectrum sharing systems, and design of cooperative communication systems. He has contributed significantly in areas such as cooperative distributed communications and signal processing for power grid systems.

Karl Berntorp is a Senior Principal Research Scientist at Mitsubishi Electric Research Laboratories. He works on statistical signal processing, Bayesian inference and learning, sensor fusion, and optimization-based control, with applications to automotive, transportation, navigation, positioning, and communication systems.

Stefano Di Cairano is a Distinguished Research Scientist and Senior Team Leader at Mitsubishi Electric Research Laboratories. He works on model predictive control, constrained control, networked control systems, optimization algorithms, stochastic systems, and their applications to automotive, aerospace, logistics, and factory automation.

Mao Konishi is a researcher at the Advanced Technology R&D Center of Mitsubishi Electric Corporation. She works in the field of precise GNSS positioning, with focus on estimation filter development and applications to single-rover systems.

Kieran Parsons is a Senior Principal Research Scientist and Senior Team Leader at Mitsubishi Electric Research Laboratories. He works on optical communications network architecture and digital signal processing algorithms for coherent optical communications, with applications to several wireless and optical technologies.

Philip V. Orlik is Vice President and Director at Mitsubishi Electric Research Laboratories. He has an extensive background in wireless communications and networking, signal processing for communication systems, queuing theory, and analytical modeling.

Yuki Sato is a researcher of Advanced Technology R&D Center of Mitsubishi Electric Corporation. Since 2009, he has engaged in centimeter-level positioning technology utilizing Japanese Quasi-Zenith Satellite System and its relevant applications.

ABSTRACT

This paper concerns the problem of multipath detection in global navigation satellite system (GNSS) positioning. An unsupervised machine learning (ML) approach is developed using a convolutional neural network in an autoencoder framework with k-means clustering in the latent space. Such methods often rely on large amounts of annotated data during training, which are difficult to collect in the context of GNSS applications. To this end, a joint approach is proposed to train the ML method on synthetic data generated by hardware in the loop (HIL) simulations, and use unsupervised domain adaptation (UDA) to reduce any discrepancies between real and simulated data. For this purpose, an UDA facilitated by a cycle-consistent generative adversarial network is proposed. It is shown that the proposed method can significantly improve the multipath detection accuracy on experimental data compared to baseline approaches (including both heuristic approaches and ML-methods). It is also shown that UDA can enhance GNSS estimation performance in the absence of annotated training data, achieving a prediction accuracy in hand-labeled experimental data of as much as 99 %.

[1]The work of R. Zawislak was done while he was working at MERL.

I. INTRODUCTION

Global navigation satellite systems (GNSSs) are used in various outdoor positioning services. In these systems, the unknown position of a receiver is inferred based on various range measurements from a set of satellites, whose positions in space are assumed to be known. These range measurements are corrupted by biases, most of which can be adequately modeled and compensated when there is a line-of-sight (LOS) between the receiver and a satellite. However, when a satellite is non-LOS (NLOS), reflections of the signals cause unpredictable multipath interference, which is notoriously difficult to model. This becomes particularly problematic in urban environments, where unexpected multipath measurements can cause large positioning errors. The key to improving positioning performance in such scenarios lies in predicting when a subset of the measurements are corrupted by unexpected biases, such that they can be handled properly in the GNSS position estimation.

Many techniques have been proposed to detect and mitigate the impact of NLOS measurements by deterministic models, see e.g., Chen et al. (2013); Dai et al. (2014). However, such approaches are difficult to deploy in a wide range of scenarios, receivers, and environments. An alternative method by Osechas et al. (2015) relies on spatial diversity-based signal processing, utilizing multiple antennas for multipath detection and mitigation. While demonstrating clear improvements in positioning performance, such methods require multiple antennas, resulting in significant hardware costs. A related approach is taken by Jiang and Groves (2014), where a dual-polarization NLOS detection scheme is proposed by using two differently polarized antennas. The outputs from the two antennas are compared to determine whether the measurements are NLOS signals or LOS. Apart from hardware re-design, a common approach to multipath mitigation is to use elevation masking, as suggested by Kubo et al. (2020), where the receiver selects a subset of visible satellites with the highest elevation angles for the positioning.

1. Machine Learning for Multipath Detection

In recent years, machine-learning (ML) techniques have led to exceptional improvements in a wide range of applications. In particular, ML has proven successful in learning complex hidden models from data, quickly surpassing state-of-the-art algorithms. As such, several ML approaches have been developed for multipath detection in GNSS data; for example, Phan et al. (2013) proposed a nonlinear regression approach that uses a support vector machine (SVM) for multipath mitigation when the receiver position is constant. Similarly, Hsu (2017) developed a supervised ML-based classifier to classify GNSS pseudorange measurements into three categories: clean, multipath, and NLOS.[2] In the works by Yozevitch et al. (2016), the authors proposed a decision-tree based classifier based on supervised learning, and recent related approaches for supervised ML for multipath detection include the works of Gogliettino et al. (2019); Suzuki et al. (2017); Munin et al. (2020); Dovis et al. (2020). In this paper, we draw special attention to the work by Dovis et al. (2020), where a `k-means`-based ML method was proposed for dynamic receiver scenarios and SVM-based ML method for static scenarios. Inspired by this work, we primarily investigate methods utilizing `k-means` clustering in the forthcoming developments.

2. Unsupervised Domain Adaptation

The aforementioned supervised ML methods require large amounts of correctly labeled training data, which is problematic in the context of GNSS positioning, as the data are nontrivial to annotate. However, there exist sophisticated GNSS simulators that can generate large amounts of synthetic data with typical noise and multipath biases. Even so, when considering the variability of real data due to environments, receiver dynamics, and physical constraints on the receivers, it is clear that there may exist significant discrepancies between the real and synthetic data. While ML-based multipath detection models trained on simulated data (referred to as the source domain) can be used on real data (referred to as the target domain), applying trained models with such a domain shift might impact the model's predictive performance. When considering the relative simplicity of simulated data and the variability of real data, reducing the perceptible difference between data in the two domains can improve the trained model's ability to generalize. Such methods are refereed to as domain adaptation (DA) techniques (see, e.g., Zhang (2019)).

For the special case where there are no labels in the target domain, and potentially no labels in the source domain, it is beneficial to consider unsupervised domain adaptation (UDA). The objective of UDA is to find functions (generators) that map data in the source domain to the target domain, and vice versa (see, e.g., Zhang (2019)). A common learning approach to UDA is to use generative adversarial networks (GANs). When training the GAN to transform data from target to source domain, it is desirable to retain information, in particular the multipath information. This implies constraints on the GAN model used for UDA. In the following, we therefore consider the cycle-consistent adversarial network, or cycle-GAN (CGAN). This model was originally developed by Zhu et al. (2017) for image-to-image translation, and the proposed architecture is interesting for the GNSS multipath detection problem as it enforces conservation of information during unsupervised domain adaptation.

[2] Multipath biases can arise both during LOS and NLOS conditions, but a measurement taken during NLOS is per definition a multipath measurement.

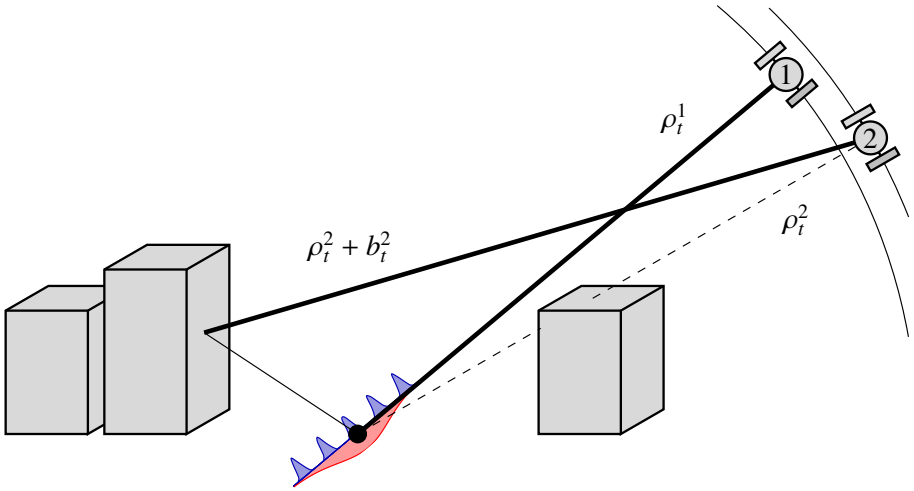

Figure 1: Geometry of a GNSS positioning problem, with the receiver as a black dot. The satellite $i = 2$ is NLOS, and instead of registering the range ρ_t^i, the receiver registers a biased range $\rho_t^i + b_t^i$. The signal bounces on a building to the left, and its signal strength, $\texttt{S1C}_t^i$, decreases as a consequence (indicated by the thickness of the lines). The range measurement from satellite $i = 1$ is LOS, and the measurement likelihood is indicated by red for the pseudorange in the $\texttt{C1C}_t^i$ measurement, and by blue for the phase-range computed from the $\texttt{L1C}_t^i$ measurement.

3. Contributions and Structure

The paper is structured in accordance with its two main contributions:

- We propose an unsupervised ML method based on an autoencoder (AE) and `k-means` clustering in the latent space to detect multipath events on data generated by hardware in the loop (HIL) simulations (see Sec. II).
- We describe how to perform UDA using a CGAN inspired by Zhu et al. (2017) to improve multipath detection accuracy on experimental data by finding an approximate mapping from real data to synthetic data (see Sec. III).

The developed algorithms are subsequently compared with conventional methods for GNSS multipath detection in Sec. IV, showing a substantial improvement of detection accuracy when using the proposed ML and UDA methods.

II. UNSUPERVISED ML FOR MULTIPATH DETECTION

We start by describing the measurement data in Sec. II.1, and then outline the temporal windowing used to generate the inputs for the ML model in Sec. II.2. The network architecture is discussed in Sec. II.3, with specifics given in Appendix A–C.

1. Measurement Data

The proposed unsupervised ML (UML) algorithm is trained on simulated data. For this purpose, we use the Orolia GSG-63 GNSS HIL simulator, where multipath injection is done as described by ITU (1997), and generate data in the Receiver Independent EXchange (RINEX) format, see RINEX Specification (2015) for a detailed description. This is done for a receiver that moves along the streets of Boston, USA. The resulting data are very rich, containing information on signal strength and of the geometric ranges between the receiver and satellites, denoted by ρ [m] (see, e.g., RINEX Specification (2015), Section 5). The geometry of the GNSS problem is related to the measured RINEX data as outlined in Fig. 1. In this paper, we only consider measurements on the L_1 frequency band of the GPS constellation, with an associated carrier wavelength of $\lambda_1 = 0.1903$ [m]. From the RINEX files, the following measurements are used to generate features (inputs) to the forthcoming algorithms:

- $\texttt{C1C}_t^i$ - Pseudo-range measurement: the distance from the receiver antenna to the i^{th} satellite, ρ_t^i, including various clock offsets, multipath biases and noise (see Fig. 1).
- $\texttt{L1C}_t^i$ - Carrier-phase measurement: the distance from the receiver antenna to the i^{th} satellite, ρ_t^i, expressed in units of cycles of the carrier frequency, λ_1^{-1}. This measurement is more precise than the pseudo range in terms of the noise variance, but includes additional biases.
- $\texttt{D1C}_t^i$ - Doppler shift: the time-derivative of the receiver's carrier phase, primarily determined by the relative velocity of the i^{th} satellite's and receiver's antennas.
- $\texttt{S1C}_t^i$ - Signal strength: the carrier-to-noise density (C/N_0) in [dbHZ] for the i^{th} satellite and receiver (see Fig. 1).

Here, the sub-index t refers to a time at which the measurements are sampled, and without loss of generality, we assume a sample rate of 1 [Hz]. Multipath biases will be present in both the $\texttt{C1C}_t^i$ and $\texttt{L1C}_t^i$ data, and implicitly the $\texttt{D1C}_t^i$ as well. Furthermore, multipath or NLOS conditions will typically be appear in the $\texttt{S1C}_t^i$ signal. We therefore define three features based on these signals over time windows of length $W \in \mathbb{N}$, and use them as inputs to the multipath detection algorithms.

2. Feature Selection

In this subsection, we explain three features used by our approach. This includes: the centered signal strength (CSS) over time; the centered code-minus-carrier (CMC) over time; and the Doppler rate consistency (DRC). These features are described in relation to the measurements in the RINEX file (see Sec. II.1), and were found through extensive feature engineering.

a) Centered Signal Strength (CSS)

It is necessary to study the relationship between the signal strength and the existence of multipath, as the multipath phenomena generally degrades signal quality in addition to introducing signal delays. The C/N_0 reported in the $\texttt{S1C}_t^i$ measurement is closely linked to the elevation. A common technique for detecting multipath is therefore to evaluate the C/N_0 relative to a threshold that depends on the satellite's relative elevation angle (see, e.g., Kubo et al. (2005)). In this paper, we use successive measurements over a time window to capture its time evolution. These are centered around the origin by taking the mean over the time window,

$$\texttt{CSS}_t^i \triangleq \texttt{S1C}_t^i + B_t, \tag{1}$$

where the offset B_t is chosen such that $\sum_{k=t-W}^{t} \texttt{CSS}_t^i = 0$ for all t and $W \in \mathbb{N}$.

b) Code Minus Carrier signal (CMC)

A discrete jump in the carrier phase measurement may be indicative of a transition from NLOS to LOS and vice versa. We therefore define the CMC for the i^{th} satellite at a time t as done by Blanco-Delgado and de Haag (2011),

$$\texttt{CMC}_t^i \triangleq \texttt{C1C}_t^i - \lambda_1 \texttt{L1C}_t^i + C_t. \tag{2}$$

This feature has been used for multipath detection in recent works by Yunwei et al. (2019); Caamano et al. (2020). Its magnitude may vary significantly, as not all of the biases in the $\texttt{L1C}_t^i$ signal appear in the $\texttt{C1C}_t^i$ signal. Consequently, the CMC is defined with respect to a time-window of length W, where C_t is chosen such that $\sum_{k=t-W}^{t} \texttt{CMC}_k^i = 0$ for all t.

c) Doppler Rate Consistency (DRC)

The consistency between the expected change in pseudorange from the measured Doppler shifts versus the actual change in pseudorange is also considered. The DRC is an effective feature for multipath and NLOS detection, since multipath often impacts the Doppler shifts and pseudorange measurements differently (see Hsu (2017)). The DRC is defined as

$$\texttt{DRC}_t^i \triangleq |(\texttt{C1C}_t^i - \texttt{C1C}_{t-1}^i) - c\Delta_t \lambda_1 (\texttt{D1C}_t^i)|, \tag{3}$$

where c [m/s] denotes the speed of light, and Δ_t [s] is the duration between two consecutive observations.

In the following, the defined features are combined into tensors $\boldsymbol{X} \in \mathbb{R}^{3\times W\times B}$, consisting of $B \in \mathbb{N}$ batches of matrices

$$\boldsymbol{x}_t^i \triangleq \begin{bmatrix} \texttt{CSS}_t^i & \texttt{CSS}_{t-1}^i & \cdots & \texttt{CSS}_{t-W}^i \\ \texttt{CMC}_t^i & \texttt{CMC}_{t-1}^i & \cdots & \texttt{CMC}_{t-W}^i \\ \texttt{DRC}_t^i & \texttt{DRC}_{t-1}^i & \cdots & \texttt{DRC}_{t-W}^i \end{bmatrix} \in \mathbb{R}^{3\times W}. \tag{4}$$

The tensor $\boldsymbol{x}_t^i$ is used as an input to the UML algorithm, for various satellites i and times t. When using the model for multipath prediction, a smaller tensor is formed at the most recent t over all i, and used as input to the prediction model.

3. CNN-AE Network Architecture

We assume that the impact of multipath and NLOS on the features defined in Sec. II.2 is significant enough to make it separable from the LOS measurements. Distinguishing between LOS and NLOS data can then be thought of as a clustering problem, for which the `k-means` algorithm is a natural choice (see Dovis et al. (2020)). However, it might not be optimal to cluster on a distance between different $\boldsymbol{x}_t^i$ constituting the input tensor $\boldsymbol{X}$ in (4). A common approach to improving the `k-means` performance consists of first training an autoencoder (AE) on the data to a set of hidden features, here denoted by $\boldsymbol{F}$, and then apply `k-means` in this latent space. For this purpose, we use a CNN implemented using PyTorch (see, e.g., Paszke et al. (2019)), referred to as a `CNN-AE`. The idea is for the `CNN-AE` to learn a compressed feature representation of the data in $\boldsymbol{F}$, from which it can reconstruct the input as $\hat{\boldsymbol{X}}$ with as high accuracy as possible. The high-level architecture of the `CNN-AE` is illustrated in Fig. 2 for a generic loss $\mathcal{L}(\boldsymbol{X}, \hat{\boldsymbol{X}}, \boldsymbol{F})$. The functional relationships between the network layers is described in Appendix A, consisting of convolutions combined with ReLu activation functions and max pooling operations (refer to Paszke et al. (2019)).

Remark 1. *The use of max-pooling has shown promising results for generalization because it sharpens global variations in the data while merging local ones. This pooling is generally done across time, but for our last layer we perform pooling across features instead. As such, we select the most important feature at each time-step to get a representation of the data. Doing so might reduce the temporal correlations of the features, but makes it easier to detect anomalies caused by multipath. Indeed, we do not care about clustering on when multipath happens in the time-window, only if it occurs.*

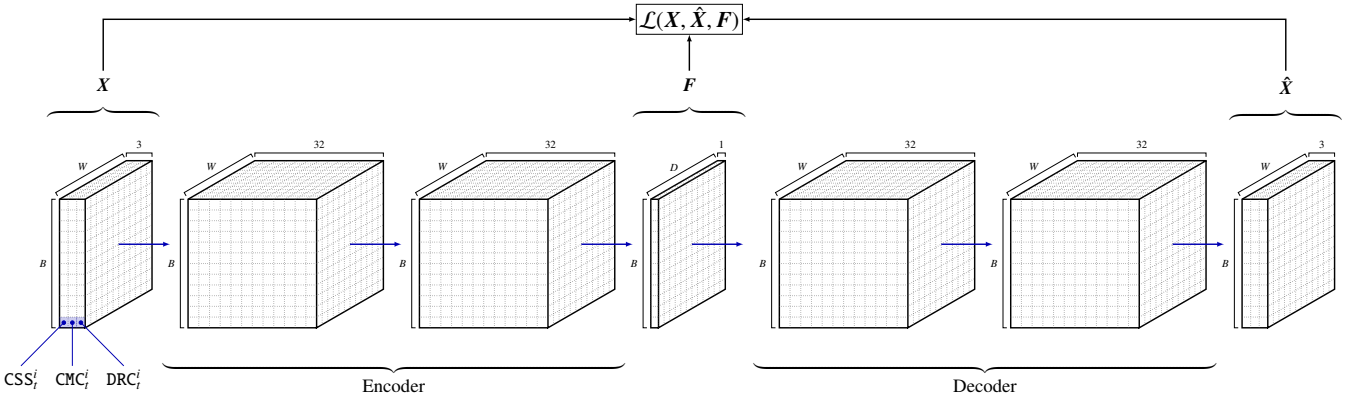

Figure 2: Sketch of the `CNN-AE` architecture, the functional relationships (arrows) defining the layers are given in Appendix A

.

4. CNN-AE Loss Functions

The objective of the `CNN-AE` training is to minimize the element-wise mean-squared error (MSE) loss function, computed as

$$\text{MSE}(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \frac{1}{3 \times B \times W} \sum_{i=1}^{B} \text{MSE}(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i), \tag{5}$$

where $B \in \mathbb{N}$ denotes the batch size. Here, $\boldsymbol{x}^i \in \mathbb{R}^{3 \times W}$ and $\hat{\boldsymbol{x}}^i \in \mathbb{R}^{3 \times W}$ denote the i^{th} batch input and reconstructed input matrices at the output layer of the `CNN-AE`, respectively. Additionally, we consider the loss functions proposed by Tao et al. (2021) to favor the cluster selection. In the work of Tao, an instance separation loss is shown to encourage the network to learn different features for each training example. Furthermore, a feature decorrelation loss encourages the network to learn distinct features. However, unlike the losses defined by Tao et al. (2021), we only apply the loss across each batch of time-windowed data. Thus, we define the batch-based instance separation (BIS) loss and batch-based feature decorrelation (BFD) loss as follows:

$$\text{BIS}(\boldsymbol{F}) \triangleq -\frac{1}{B \times D} \sum_{i=1}^{B} \log\Big(\frac{\exp(\boldsymbol{f}_i \boldsymbol{f}_i^\top)}{\sum_{i=1}^{B} \exp(\boldsymbol{f}_i \boldsymbol{f}_j^\top)} \Big), \qquad \text{BFD}(\boldsymbol{G}) \triangleq -\frac{1}{B \times D} \sum_{i=1}^{D} \log\Big(\frac{\exp(\boldsymbol{g}_i \boldsymbol{g}_i^\top)}{\sum_{i=1}^{D} \exp(\boldsymbol{g}_i \boldsymbol{g}_j^\top)} \Big), \tag{6}$$

where D denotes the feature size of the bottleneck layer, and $\boldsymbol{f}_i \in \mathbb{R}^{1 \times D}$ denotes the i^{th} feature vector extracted at the bottleneck layer, that is, the i^{th} row vector of the matrix $\boldsymbol{F} \in \mathbb{R}^{B \times D}$. For the transposed matrix of $\boldsymbol{G} \triangleq \boldsymbol{F}^\top$, $\boldsymbol{g}_i \in \mathbb{R}^{1 \times B}$ denotes the i^{th} row vector of $\boldsymbol{G}$. The losses in (5) and (6) are weighted equally in an overall loss function, defined as

$$\mathcal{L}(\boldsymbol{X}, \hat{\boldsymbol{X}}, \boldsymbol{F}) \triangleq \text{MSE}(\boldsymbol{X}, \hat{\boldsymbol{X}}) + \text{BIS}(\boldsymbol{F}) + \text{BFD}(\boldsymbol{F}^\top). \tag{7}$$

5. CNN-AE Training

The training and testing of the proposed CNN-AE with k-means clustering is accomplished in three main steps:

1. Train the CNN-AE in an unsupervised manner on simulated data from the HIL simulator, minimizing the loss function $\mathcal{L}(\boldsymbol{X}, \hat{\boldsymbol{X}}, \boldsymbol{F})$ defined in (7).
2. Compute hidden features of the entire training data and train a k-means clustering on the hidden features.
3. Compute hidden features of the testing data, $\boldsymbol{F}$, and predict the cluster using the trained k-means, where the output is a binary vector $\boldsymbol{Y} \in \{0,1\}^B$. If a slice of the tensor $\boldsymbol{x}_t^i$ in (4) is LOS, the corresponding element in the vector $\boldsymbol{Y}$ is set to 0.

Numerical experiments using the above network architecture is presented in Sec. IV. Before showing these results, we first describe how unsupervised domain adaptation (UDA) can be leveraged to improve the model's performance on real data.

III. UNSUPERVISED DOMAIN ADAPTATION WITH THE CGAN

The basic idea of domain adaptation is to find a function that maps experimental data in a source domain, $\boldsymbol{X}_R \in \mathcal{D}_R$, into data that more closely resemble the output of the HIL simulator in the target domain, $\boldsymbol{X}_S \in \mathcal{D}_S$. These tensors are constructed as in Sec. II, such that the output from the UDA can be used directly with the CNN-AE. For the domain adaptation, we consider a network structure similar to the cycle-consistent adversarial network (CGAN), originally proposed in Zhu et al. (2017). The CGAN model is composed of two generators: a simulated generator, $\boldsymbol{G}_S : \mathcal{D}_S \mapsto \mathcal{D}_R$, to transform simulated data to real data, and a real generator, $\boldsymbol{G}_R : \mathcal{D}_R \mapsto \mathcal{D}_S$, to transform real to simulated data. It also trains two discriminators to distinguish between true and fake generated data for each domain; with $\boldsymbol{D}_S : \mathcal{D}_R \vee \mathcal{D}_S \mapsto \{0,1\}^B$ for the simulated domain and $\boldsymbol{D}_R : \mathcal{D}_R \vee \mathcal{D}_S \mapsto \{0,1\}^B$ for the real domain. It is trained in a fully unsupervised manner, and can be applied to any real data without the need for labels. The information flow in a high level representation of the CGAN is depicted in Fig. 3.

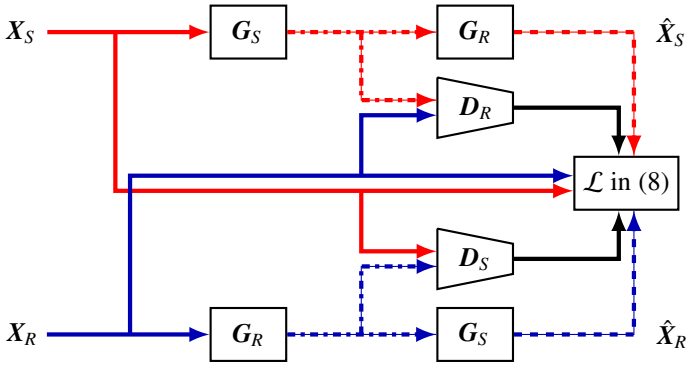

Figure 3: Figure of the CGAN model used for the domain adaptation and its information flow. Red and blue-colored arrows denote data flow for simulated data and real data, respectively. Black colored arrows denote the binary classification from the two discriminators

.

The general idea of the UDA-aided CNN-AE is then to use the generator $\mathbf{G}_R$ in Fig. 3 to transform the real data before passing it to the CNN-AE for multipath detection. This pipeline is illustrated in Fig. 4.

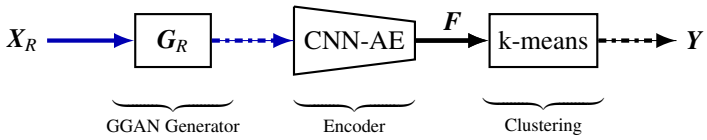

Figure 4: The generator $\boldsymbol{G}_R$ of the CGAN (in Fig. 3) is used for domain adaptation. The encoder of the CNN-AE (in Fig. 2) is used to compute a set of latent variables, $\boldsymbol{F}$, on which the k-means clustering is run to determine if the inputs, $\boldsymbol{X}_R$, are LOS or multipath/NLOS.

1. CGAN Network Architecture

The architectures of the generators and discriminators in Fig. 3 are detailed in Appendix B, and the network is implemented via PyTorch (see Paszke et al. (2019)). In its construction, we make use of residual connections, as suggested by Zhu et al. (2017). Residual connections serve to make the the model focus on data variation and to reduce the problems with vanishing gradients that can occur in deep CNNs (refer to He et al. (2015); Zhang et al. (2017)). In contrast to the network model of `CNN-AE`, the network models in the CGAN employs an instance norm in these residual blocks to stabilize the convergence behaviours (see, e.g., Ulyanov et al. (2017)). In our implementation, residual connections are used both for the generators and discriminators, but in contrast to the work of Ulyanov et al. (2017), we do not include a bottleneck layer in the generator. Finally, it was observed that a small batch size seemed particularly beneficial for good performance in the CGAN when considered for multipath detection. The parameters of the network and its functional relationships are detailed in Appendix B.

2. CGAN Loss Functions

A mean squared error (MSE) loss function is used to train both the generators and the discriminators of the CGAN (refer to the implementation in Paszke et al. (2019)), but an element-wise maximum absolute error (MAE) for the cycle consistency loss, as proposed by Zhu et al. (2017). The model is trained by alternatively minimizing the generator loss function, $\mathcal{L}_G$, and discriminator loss function, $\mathcal{L}_D$, to minimize a combined loss function expressed as their sum, $\mathcal{L}$. These are defined as follows:

$$\mathcal{L}_G(\boldsymbol{X}_S, \boldsymbol{X}_R) \triangleq \mathrm{MSE}(\mathbf{1}, \boldsymbol{D}_S(\boldsymbol{G}_R(\boldsymbol{X}_R))) + \mathrm{MSE}(\mathbf{1}, \boldsymbol{D}_R(\boldsymbol{G}_S(\boldsymbol{X}_S))) + \lambda_S \ \mathrm{MAE}(\boldsymbol{X}_S, \hat{\boldsymbol{X}}_S) + \lambda_R \ \mathrm{MAE}(\boldsymbol{X}_R, \hat{\boldsymbol{X}}_R), \tag{8a}$$

$$\mathcal{L}_D(\boldsymbol{X}_S, \boldsymbol{X}_R) \triangleq \mathrm{MSE}(\mathbf{0}, \boldsymbol{D}_S(\boldsymbol{G}_R(\boldsymbol{X}_R))) + \mathrm{MSE}(\mathbf{0}, \boldsymbol{D}_R(\boldsymbol{G}_S(\boldsymbol{X}_S))) + \mathrm{MSE}(\mathbf{1}, \boldsymbol{D}_R(\boldsymbol{X}_R)) + \mathrm{MSE}(\mathbf{1}, \boldsymbol{D}_S(\boldsymbol{X}_S)), \tag{8b}$$

$$\mathcal{L}(\boldsymbol{X}_S, \boldsymbol{X}_R) \triangleq \mathcal{L}_G(\boldsymbol{X}_S, \boldsymbol{X}_R) + \mathcal{L}_D(\boldsymbol{X}_S, \boldsymbol{X}_R). \tag{8c}$$

Here, $\mathbf{1}$ and $\mathbf{0}$ denote tensors of ones and zeros, respectively, of appropriate dimensions; $\boldsymbol{X}_S$ and $\boldsymbol{X}_R$ are sample tensors from the simulated and real data domains; $\hat{\boldsymbol{X}}_S$ and $\hat{\boldsymbol{X}}_R$ are reconstructed tensors from the simulated and real data domains (see Fig. 3); and λ_S and λ_R denote weighting factors for the cycle consistency loss function in both domains. The implementation and choice of loss functions are inspired by Zhu et al. (2017), but adjustments were made for the GNSS application (such as the batch size).

IV. NUMERICAL EXPERIMENTS

The proposed `CNN-AE` is evaluated both on the HIL simulated data and publicly available experiments logs provided by Narula et al. (2020), here referred to as the TEX-CUP data set. In the former case, the presence or absence of multipath in the data is known, and in the latter case, there is no ground truth data available.[3] Here we manually label each element of the batch in (4) based on the elevation angle of the satellites and the $\texttt{S1C}_t^i$ values reported in the RINEX files, omitting any ambiguous cases. As such, we expect that the proposed method to perform well on the simulated data, as both the training and testing data come from this domain, and worse on the TEX-CUP data set.

1. Methods used in the Comparison

To assess the performance of the `CNN-AE` and the impact of domain adaptation, we compare the proposed algorithms to three heuristic approaches for multipath detection based on simple thresholding:

`Heuristic 1` Compare the minimum measured $\texttt{S1C}^i$ value versus the average over a time window of length $W = 16$, to detect sudden drops or high variations in $\mathrm{C/N_0}$. By this heuristic approach, multipath is detected if the average minus the minimum is greater than a defined threshold.

`Heuristic 2` Compare the minimum and maximum measured $\texttt{S1C}^i$ values to detect variations of high amplitude. We detect multipath if the maximum minus the minimum over a time window of length $W = 16$ is greater than a threshold.

`Heuristic 3` Compare the minimum measured $\texttt{S1C}^i$ value versus the expected value from an model based on the satellite elevation angle, as proposed in Tokura and Kubo (2017). Here, detect multipath if the absolute value of the expected (modeled) value minus the minimum is greater than a defined threshold.

For these heuristic approaches, we use a threshold of 5 [dBHz], which seems to yield good classification results. It is worth noting that for the considered data, this threshold works better with `Heuristic 3` than the threshold of 10 [dBHz] proposed in Tokura and Kubo (2017). In addition to these heuristic approaches, we consider a simple classifier implemented as a CNN with residual connections, trained in a supervised manner on the simulated data, with and without domain adaptation implemented using a CGAN in Sec. II. The details of this model are discussed in Appendix C, and it is referred to as the `Naïve Classifier`.

[3] This form of evaluation is common when considering multipath detection in experimental data, as the labeling of data is ambiguous. However, the accuracy results should be viewed with this in mind, and may look slightly different.

2. Prediction Accuracy

In this section, we provide a comparison of prediction accuracy[4] with the three baseline heuristic approaches, as well as the `Naïve Classifier` and the `CNN-AE + k-means`, the latter used both with and without domain adaptation (see Table 1).

Table 1: Detection accuracy of the heuristics, the `Naïve Classifier`, and the `CNN-AE +k-means` on the simulated and the TEX-CUP data.

Model used for detection	Simulated dataset	TEX-CUP dataset
`Heuristic 1`	0.83	0.78
`Heuristic 2`	0.97	0.80
`Heuristic 3`	0.74	0.78
`Naïve Classifier`	0.86	0.89
`Naïve Classifier` + UDA	-	0.94
`CNN-AE + k-means`	1	0.93
`CNN-AE + k-means` + UDA	**–**	**0.99**

From the results reported in Table 1, we note that while the heuristic approaches certainly work better than random predictions, the performance is not perfect. While the `Heuristic 2` works well on the simulated data, performance is less impressive on the experimental dataset. On the other hand, the ML-based approaches perform well both on the simulated and real data, outperforming the heuristic techniques in this respect. This demonstrates the potential of ML for multipath detection, even when using simulated data for training and using an unsupervised approach. We also observe clear improvements when performing UDA with a CGAN (see Sec. III) to first transform the data such that its statistical properties more closely resembles the simulated data used during training. For both of the `Naïve Classifier` and the `CNN-AE +k-means`, the domain adaptation greatly improves prediction performance on the real TEX-CUP dataset.

3. Training the `CNN-AE + k-means`

Next, we provide an analysis of the `CNN-AE + k-means` training accuracy as a function of the training epoch, which indicates that the model learns a representation with smaller clusters in time. This reduces the performance of the model, which should ideally learn only two clusters. We hypothesize that at the beginning of training, the model starts with the two most natural clusters, i.e., multipath and no multipath which are certainly partly discernible in the considered features (recall, the Heuristic 1 and 2 can be computed directly from the `CSS` feature). To demonstrate this and the propensity for the `CNN-AE + k-means` to over fit, Table 2 shows the training accuracy of `CNN-AE + k-means` for 50, 100, 150, 200 and 500 training epochs.

Table 2: Multipath detection accuracy of `CNN-AE + k-means` for various numbers of training epochs.

Training epochs	Simulated	TEX-CUP
50	1	0.86
100	**1**	**0.93**
150	1	0.92
200	1	0.91
500	0.93	0.53

In Fig. 5, we also provide the t-distributed stochastic neighbor embedding (t-SNE) visualization[5] of the hidden features of autoencoder for the simulated and TEX-CUP data sets at 0, 100 and 500 training epochs. Here, we note that at 100 epochs, two clusters are quite distinct and correspond to the label distribution in simulated data. However after 500 epochs, we observe many smaller clusters, which makes a `k-means` clustering operating in the latent space with two clusters far less accurate.

[4] Simulated dataset and TEX-CUP dataset respectively represent imbalanced distributions, 47% and 24% for the minority class (NLOS and multipath). Thus, we use the detection accuracy as the performance metric of the classifier, as motivated in Brownlee (2020).

[5] t-SNE is a statistical technique for visualizing and analyzing high-dimensional data, see, e.g., Van der Maaten and Hinton (2008).

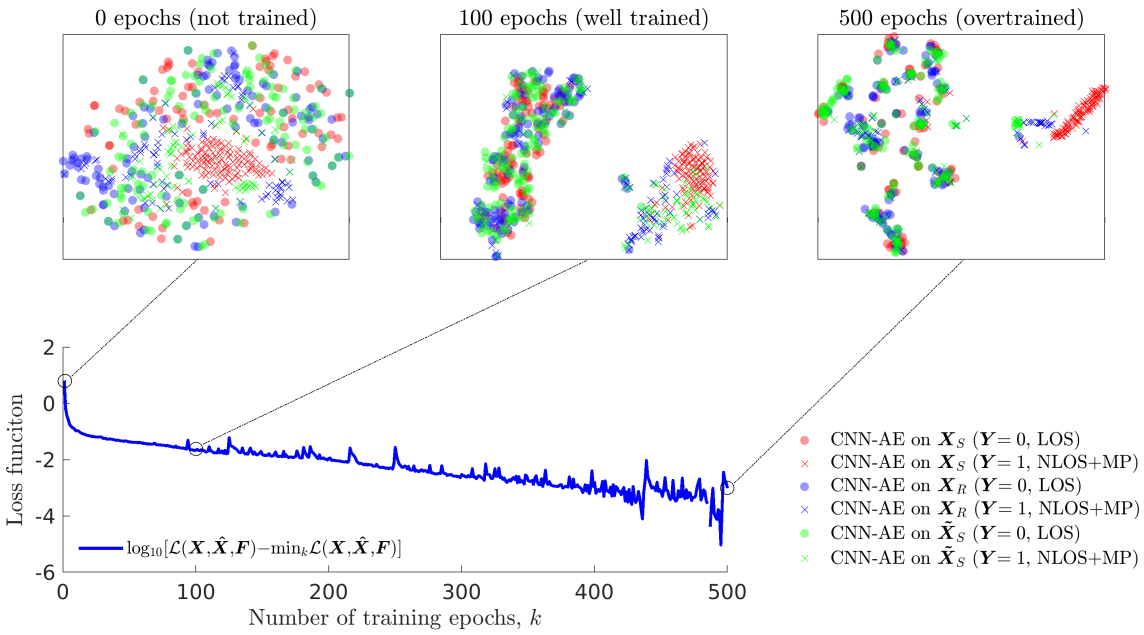

Figure 5: Depiction of $\mathcal{L}(\boldsymbol{X}, \hat{\boldsymbol{X}}, \boldsymbol{F})$ in training, with a snapshot of the resulting `CNN-AE` applied to simulated data (red), experimental TEX-CUP data (blue), and TEX-CUP data transformed into the simulated domain (green). The circles indicate LOS, and crosses a NLOS/multipath prediction, both shown in clusters using a t-distributed stochastic neighbor embedding. The training at epoch 100 is used in the evaluation.

V. CONCLUSIONS

In this paper, we have propose novel ways of detecting multipath and NLOS conditions in GNSS positioning applications, using only the data provided in RINEX navigation files in the context of moving receivers. For classification of synthetic data from a sophisticated hardware-in-the-loop simulator, near perfect accuracy was achieved by very simple heuristic approaches operating on C/N_0. In particular, a time-windowed heuristic approach on the range of the $\texttt{S1C}^i$ measurement over a time-window performed exceptionally well. Indeed, it outperformed the `Naïve Classifier` on the synthetic hardware-in-the-loop data. However, performance of the heuristic methods was reduced significantly when tested on the real TEX-CUP data set.

As for the ML-based approaches, a model defined by a `CNN-AE` + `k-means` clustering in the latent space, similar to that proposed in Dovis et al. (2020) was shown to perform very well when compared to the simpler `Naïve Classifier`. This was the case both on the simulated and the real TEX-CUP data. Indeed, the `CNN-AE` + `k-means` yielded perfect training accuracy on the simulated data (down to rounding errors), and even without domain adaptation, this method outperformed all of the tested heuristic approaches and the `Naïve Classifier` by a wide margin.

Finally, we have shown that domain adaptation with a cycle-consistent GAN, trained in an unsupervised manner, greatly enhanced prediction accuracy of the ML-based classifiers when operating on the real TEX-CUP dataset. We conclude that training an ML-based classifier on large amounts of annotated simulation data from a sufficiently sophisticated HIL simulator can work well in practice, and that the performance of such a system may be enhanced further by UDA.

In our future work, the proposed approach of UDA, combined with the `CNN-AE` + `k-means` classifier, will be used as a measurement mask in the GNSS position estimator. In this forthcoming work, performance will not be assessed by prediction accuracy, which can only be as good as the hand labeling of the experimental data. This is a point of critique in most works on ML-based multipath detection involving experimental data. Instead, the effects of the proposed method will be measured in the statistics of the positional estimation error, providing a measure of performance that is free of any hand-labeling and directly related to the goal of the multi-path detection.

REFERENCES

Blanco-Delgado, N. and de Haag, M. U. (2011). Multipath analysis using code-minus-carrier for dynamic testing of GNSS receivers. In *Proceedings of the 2011 International Conference on Localization and GNSS (ICL-GNSS)*, pages 25–30.

Brownlee, J. (2020). *Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*. Machine Learning Mastery, 1.2 edition.

Caamano, M., Crespillo, O. G., Gerbeth, D., and Grosch, A. (2020). Detection of GNSS multipath with time-differenced code-minus-carrier for land-based applications. In *Proc. 2020 European Navigation Conference (ENC)*, pages 1–12.

Chen, X., Dovis, F., Peng, S., and Morton, Y. (2013). Comparative studies of GPS multipath mitigation methods performance. *IEEE Transactions on Aerospace and Electronic Systems*, 49(3):1555–1568.

Dai, W., Huang, D., and Cai, C. (2014). Multipath mitigation via component analysis methods for GPS dynamic deformation monitoring. *GPS solutions*, 18(3):417–423.

Dovis, F. et al. (2020). Opportunistic use of GNSS signals to characterize the environment by means of machine learning based processing. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9190–9194.

Gogliettino, G. et al. (2019). A machine learning approach to GNSS functional safety. In *Proc. 32nd Int. Tech. Meeting of the Satellite Division of The Institute of Navigation*, pages 1738–1752.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *ArXiv*, abs/1512.03385.

Hsu, L.-T. (2017). GNSS multipath detection using a machine learning approach. In *Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6.

ITU (1997). Guidelines for evaluation of radio transmission technologies for IMT-2000. *Report ITU-R M.1225*.

Jiang, Z. and Groves, P. D. (2014). NLOS GPS signal detection using a dual-polarisation antennas. *GPS Solutions*, 18(1):15–26.

Kubo, N., Kobayashi, K., and Furukawa, R. (2020). GNSS multipath detection using continuous time-series C/N_0. *Sensors*, 20(14):1–25.

Kubo, N., Suzuki, T., Yasuda, A., and Shibazaki, R. (2005). An effective method for multipath mitigation under severe multipath environments. In *Proc. 18th Int. Tech. Meeting of the Satellite Division of The Institute of Navigation*, pages 2187–2194.

Munin, E., Blais, A., and Couellan, N. (2020). Convolutional neural network for multipath detection in GNSS receivers. In *Proc. Int. Conf. on Artificial Intelligence and Data Analytics for Air Transportation*, pages 1–10.

Narula, L. et al. (2020). TEX-CUP: The university of Texas challenge for urban positioning. In *Proc. IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 277–284.

Osechas, O., Kim, K. J., Parsons, K., and Sahinoglu, Z. (2015). Detecting multipath errors in terrestrial GNSS applications. In *Proceedings of the 2015 International Technical Meeting of The Institute of Navigation*, pages 465–474.

Paszke, A. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035.

Phan, Q., Tan, S., and McLoughlin, I. (2013). GPS multipath mitigation: a nonlinear regression approach. *GPS Solutions*, 17(3):371–380.

RINEX Specification (2015). RINEX: The Receiver Independent Exchange Format, Version 3.03. Technical report, International GNSS Service.

Suzuki, T., Nakano, Y., and Amano, Y. (2017). NLOS multipath detection by using machine learning in urban environments. In *Proc. 30th Int. Tech. Meeting of the Satellite Division of The Institute of Navigation*, pages 3958–3967.

Tao, Y., Takagi, K., and Nakata, K. (2021). Clustering-friendly representation learning via instance discrimination and feature decorrelation. *ArXiv*, abs/2106.00131.

Tokura, H. and Kubo, N. (2017). Efficient satellite selection method for instantaneous RTK-GNSS in challenging environments. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 60(4):221–229.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Instance normalization: The missing ingredient for fast stylization.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11).

Yozevitch, R., Moshe, B. B., and Weissman, A. (2016). A robust GNSS LOS/NLOS signal classifier. *Navigation: Journal of The Institute of Navigation*, 63(4):429–442.

Yunwei, L. et al. (2019). Estimation of snow depth using pseudorange and carrier phase observations of GNSS single-frequency signal. GPS Solutions. Art. No.118.

Zhang, J., Zheng, Y., and Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proc. Thirty-First AAAI Conference on Artificial Intelligence*, pages 1655–1661.

Zhang, L. (2019). Transfer adaptation learning: A decade survey. *ArXiv*, abs/1903.04687.

Zhu, J., Park, T., Isola, P., and Efros, A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232.

A. DETAILS OF THE CNN-AE IMPLEMENTATION

Let `c5s1-k` denote a Convolution- ReLU layer with k input channels, defined using a kernel size of 5 and a stride of 1 (`Conv1d`).[6] Let `bn` denote a batch normalization (`BatchNorm1d`) and `mp` denote max pooling (`MaxPool1d`) of size 3 and stride 1. Similarly, let `t5s1-k` denote a transposed Convolution-ReLU layer with k input channels, with a kernel size of 5 and a stride of 1 (`ConvTranspose1d`). Finally, let `norm` denote a euclidean normalization (`Normalize`). The `CNN-AE` is defined as

$$\text{Encoder} : \texttt{c5s1-3} \to \texttt{bnmp} \to \texttt{c5s1-32} \to \texttt{bnmp} \to \texttt{c5s1-32} \to \texttt{norm}. \tag{9a}$$

$$\text{Decoder} : \texttt{t5s1-1} \to \texttt{bn} \to \texttt{t5s1-32} \to \texttt{bn} \to \texttt{t5s1-32}. \tag{9b}$$

The dimensions are defined with a window $W = 16$, a latent space characterised by $D = 16$, and a batch size of $B = 512$. It was trained with an ℓ_2-regularized ADAM optimizer (`ADAM`), using a learning rate of $\gamma = 10^{-5}$ and a weight decay of $\lambda = 0.01$.

B. DETAILS OF THE CGAN IMPLEMENTATION

The generators of the CGAN model are defined by a succession of strided convolution and residual blocks. A residual block (see e.g., Zhang et al. (2017)) is defined by a direct feed-trough of the the argument, which is summed with the output of a function. As in Zhu et al. (2017), the function in our residual block is formed by sequential application of an Instance norm (`InstanceNorm`), a Convolution with kernel size 3 and stride 1 (`Conv1d`), a Leaky ReLU (`LeakyReLU`) activation function, and by dropout (`Dropout`) with probability $p = 0.05$. Here, `Rk` denotes a residual block with k input channels. The CGAN generators are formed as

$$\text{Generator} : \texttt{c5s1-3} \to \texttt{R256} \to \texttt{R256} \to \texttt{R256} \to \texttt{R256} \to \texttt{R256} \to \texttt{t5s1-3}. \tag{10a}$$

The CGAN discriminators are defined similarly, adding a linear transformation in the final layer (`Linear`), referred to as `lin`,

$$\text{Discriminator} : \texttt{c5s1-3} \to \texttt{R32} \to \texttt{R32} \to \texttt{lin}. \tag{10b}$$

Similart to the `CNN-AE`, the CGAN is defined with a window $W = 16$, a latent space characterised by $D = 16$, but a much smaller batch size of $B = 16$. It was trained with an ℓ_2-regularized ADAM optimizer (`ADAM`), using a learning rate of $\gamma = 10^{-5}$ and a weight decay of $\lambda = 0.01$, and the parameters of the CGAN loss function were defined with $\lambda_R = \lambda_S = 20$.

C. DETAILS OF THE NAIVE CLASSIFIER IMPLEMENTATION

Let `c5s1-k` denote a Convolution- ReLU layer with k input channels, defined using a kernel size of 5 and a stride of 1 (`Conv1d`). Just as with the CGAN generator, the naive classifier is constructed using residual blocks (see e.g., Zhang et al. (2017)). However, for the classifier, the function in these residual blocks is formed by sequential application of an batch norm (`BatchNorm1d`), a Convolution with kernel size 5 and stride 1 (`Conv1d`), a ReLU activation function (`ReLU`), and by dropout (`Dropout`) with probability $p = 0.1$. Here, `Rk` denotes a residual block with k input channels, and the classifier is defined by

$$\text{Classifier} : \texttt{c5s1-3} \to \texttt{R32} \to \texttt{R32} \to \texttt{lin}, \tag{11}$$

adding a linear transformation in the final layer with two classes (`Linear`), and taking the argmax over each two-dimensional channel in the output. Unlike the `CNN-AE`, training was done using a cross-entropy loss (`CrossEntropyLoss`) in a supervised manner using an ℓ_2-regularized ADAM optimizer (`ADAM`), with a learning rate of $\gamma = 5 \cdot 10^{-5}$ and a weight decay of $\lambda = 0.01$.

[6] Here, the notation (`XXX`) refers to a specific function in TensorFlow, refer to the documentation associated with Paszke et al. (2019).