applied sciences

Article

Semi-Supervised GNSS Scintillations Detection Based on DeepInfomax

Giulio Franzese [1,*], Nicola Linty [2] and Fabio Dovis [1]

1 Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy; fabio.dovis@polito.it
2 Department of Navigation and Positioning, Finnish Geospatial Research Institute (FGI-NLS), 02430 Masala, Finland; nicola.linty@nls.fi
* Correspondence: giulio.franzese@polito.it

Received: 30 November 2019; Accepted: 30 December 2019; Published: 4 January 2020

Abstract: This work focuses on a machine learning based detection of ionospheric scintillation events affecting Global Navigation Satellite System (GNSS) signals. We here extend the recent detection results based on Decision Trees, designing a semi-supervised detection system based on the DeepInfomax approach recently proposed. The paper shows that it is possible to achieve good classification accuracy while reducing the amount of time that human experts must spend manually labelling the datasets for the training of supervised algorithms. The proposed method is scalable and reduces the required percentage of annotated samples to achieve a given performance, making it a viable candidate for a realistic deployment of scintillation detection in software defined GNSS receivers.

Keywords: scintillation detection; GNSS; DeepInfomax; convolutional neural network; semi-supervised learning

1. Introduction

Ionospheric scintillations are defined as phase and amplitude variations of trans-ionospheric radio signals, caused by irregularities and rapid fluctuations of the electron content in the upper layers of atmosphere. In this work, we focus on the effect of scintillation on Global Navigation Satellite Systems (GNSS) signals. Whether the main purpose of the application under consideration is navigation, in which the user demands high quality measurements and scintillation mitigation or rejection, or a scientific application, in which the user is interested in selecting scintillation events within huge datasets, a clean, fast, and precise detection of scintillation events is of paramount importance. The scientific literature on the study of scintillations and their effect on GNSS systems is wide and complete, and therefore is outside the aim of this paper. We refer the reader to several sources to study in depth the scintillation topic [1–4], the receiver design, and signal processing techniques [5–7].

The idea of using machine learning techniques for scintillation detection is not new. The need for machine learning solutions is due to the fact that manual classification of scintillation events based on visual inspection of scintillation indices time series is nowadays unpractical, given the amount of data. Similarly, simple heuristic algorithms, while requiring barely any tuning time from a human expert, have limited performance. Examples are algorithms that decide whether scintillation is occurring or not based on a simple threshold crossings of quantities such as the amplitude scintillation index (S_4), the carrier to noise power spectral density ratio (C/N_0) or the satellite elevation angle. It has been proven that machine learning algorithms can successfully replace and overcome traditional detection techniques.

Such approach was originally proposed in [8,9], where the authors use an automatic scintillation detector based on a supervised machine learning technique called support vector machine (SVM), respectively for amplitude and phase scintillation. This approach provides good detection results but introduces a Fast Fourier Tranform (FFT) operation on the data over windows of 3 min, thus reducing the temporal resolution of the detection results. A second disadvantage is that the input data need to be pre-filtered at an elevation mask of 30° to reduce multipath false alarms, at the expenses of a potential waste of useful and important information. An updated version of the algorithm is presented in [10]. Other works propose using decision tree and random forest algorithms and low level signal observables, such as the in-phase and quadrature correlation outputs of the receiver tracking loop [11,12]. This approach is able to reduce the false alarms rate due to the ambiguity between scintillation and multipath typical of the classical approaches based on the analysis of amplitude scintillation index and provide an early run-time alert. Another advantage is that, by exploiting the high rate correlator outputs, a finer time resolution in detection is obtained [13,14]. A more generic survey of data mining techniques is given in [15]. However, this method does not only rely on GNSS observations, but also includes external data, such as sensors and online forecast services.

The biggest problem with machine learning based classification is that datasets must be labeled manually by experts, wasting precious resources in terms of time, and thus preventing large scale scalability of the method [8,11]. The techniques proposed in the literature and described above partially solve the problem as they are based on fully-supervised learning: a significant portion of input data must be manually labeled to assist the learning process of the algorithm. We here propose to switch to a paradigm centered around semi-supervised learning, in which in contrast to fully supervised learning the label information is not required for all the data, and some entries can be thus classified with an "*unknown*" label.

In recent years, deep learning based solutions [16] for classification tasks have seen tremendous improvements in terms of performance. Neural networks in particular, however, have been rarely used as detection algorithms, one exception being [17]. Nevertheless, they have been successfully deployed for complex modeling of electron content in the ionosphere [18,19].

Usually, deep learning based semi-supervised classifiers (see, for example, [20]), reduce the dimensionality of the unlabeled data X using unsupervised learning, then build a classifier using only the compressed labeled datapoints F for the datapoints where the corresponding label Y is available, as depicted in the scheme of Figure 1.

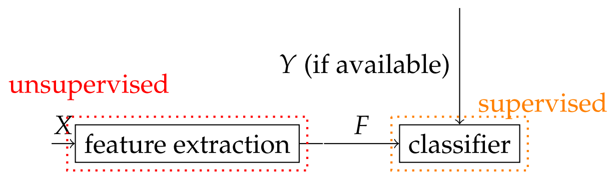

Figure 1. Semi-supervised learning. Feature extraction is performed in an unsupervised fashion, while classification is based on a supervised approach (labels).

In this paper, we make use of the DeepInfomax [21] approach in which a variational representation of mutual information [22] is used to build an unsupervised, information theoretic, feature extractor. Differently from other unsupervised techniques such as Variational Autoencoders [20] that require both an encoder and a decoder, the considered method only needs and encoder, almost halving the complexity. Variational techniques powered by neural networks for mutual information estimation, and in general divergences between probabilities, have been widely used in the literature [22–26]. The complete pipeline we propose consists of firstly compressing the input time series into smaller cardinality vectors, and then feed these vectors to a binary classifier for scintillation detection. During training, the unsupervised compression is performed for all datapoints, whereas, obviously, the training of the classifier is performed only for the labeled datapoints.

The aim of this paper is then to develop a new scintillation detection strategy based on Convolutional Neural Networks and semi-supervised learning. In more detail, we extend the results and approaches presented in [11], in which classification of scintillation events is performed using supervised decision trees. While showing good performance, this method has two fundamental problems. First, it lacks the capability of considering as a whole a single scintillation event sampled at different time instants, as the time feature is not taken into account. Second, it requires to have a fully labeled dataset, where the process of annotation had to be done manually by a team of experts in the field. We instead propose a solution that mitigates both problems by using a system that naturally:

- handles subsequent samples in time, by applying an overlapping time window to all features, thus including the information about the temporal correlation;
- requires only a smaller portion of the dataset to be labeled.

We show that to reach the same level of accuracy, just about 20% of datapoints must be manually labeled. For a faithful comparison, we used the same dataset as in [11] and we performed statistical measurements of testing classification performance using the same splitting of datasets for training and testing. Moreover, when the amount of labeled datapoints is augmented, the proposed method reaches extremely high performance with a top accuracy on the test set of 0.9993.

The overview of the paper is then the following: in Section 2, we discuss the need for unsupervised compression; in Section 3, we give a brief overview of the DeepInfomax approach that we further expand in Appendix A; in Section 4, we describe the considered datasets and their properties; in Section 5, we give an overview of the system architecture and training and testing procedure; and, in Section 6, we show the results of the proposed classification method. Finally, in Section 8, we discuss on the results and suggest possible future work directions. We also include a complete Appendix B to briefly describe the neural network architectures and training technicalities.

2. The Need for Unsupervised Compression

As depicted in Figure 1, one possible semi-supervised classification scheme is composed of two main blocks: an unsupervised feature extractor and a supervised classifier that is trained using as input the features. Notice that for some features the label correspondence is missing. The attentive reader might wonder why the first portion of the deep learning based scheme, the unsupervised encoding part, depicted in Figure 1, is necessary at all and a simple training of a classifier on the labeled portion of the dataset is not sufficient. The reason is that, as explained in the introduction (Section 1), in this work, we improved on two weak points of the previous decision tree based classification algorithms: being able to consider a collection in time of the features and to reduce the number of labeled training points. Generally speaking, in machine learning, these two requirements are in conflict: when increasing the cardinality of the inputs, as a rule of thumb, to maintain a good level of performance the number of training points must be increased. This is typically solved when dealing with high-dimensional inputs either by performing some form of manual feature engineering and reducing the dimensionality of the input or by increasing the amount of labeled data by acquiring larger labeled datasets. The approach of semi-supervised deep learning is instead different. Given a large dataset in terms of number of samples, with a large input cardinality, with a small number of labeled datapoints, an automatic, unsupervised, end-to-end feature extraction (we use interchangeably the term feature extraction or compression.) is performed on the whole dataset. A classifier is then trained using as inputs the small feature vectors thus resolving the problem of having few labeled datapoints, having the input be a small cardinality, a small number of labeled samples is sufficient to maintain the required level of performance. It will be shown in Section 5 in this work we consider a compression ratio of 32:1. In this work, we decided to use an information theoretic based feature extraction scheme [21].

3. The DeepInfomax Approach

In this section, we give a brief overview of the DeepInfomax approach [21]. Appendix A provides mode details about the technique. Figure 2 schematically depicts the idea.

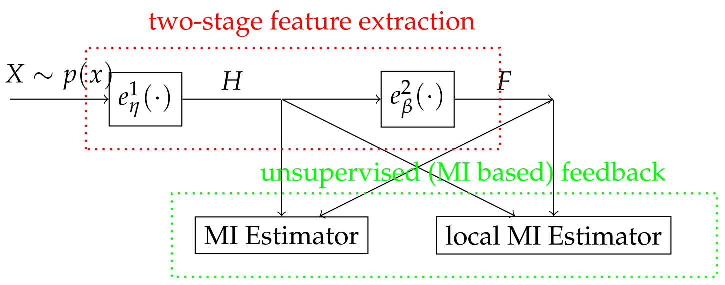

Figure 2. Deep Infomax—high level scheme.

The main idea is to build a two stage compression, using a cascade of two neural networks $e^1_\eta(\cdot)$ and $e^2_\beta(\cdot)$, of an input X, respectively generating a first equivalent feature $H = e^1_\eta(X)$ and a second one $F = e^2_\beta(H)$, such that the Mutual Information (MI) between H, F is maximized. In practice, as also explained in [21], a proxy to the mutual information that worked better in practice, the Jensen Shannon Divergence, was used. The authors in [21] introduce, and also maximize, a new metric: the Local Mutual Information. It is defined as the sum of all mutual information between each element $H^{(z)}, \quad z = 1, \cdots |H|$ of the variable H, that is, a multidimensional random vector and the whole feature vector F. In equations, $MI_{loc} = \frac{1}{|H|} \sum_{z=1}^{|H|} \mathrm{I}\left(H^{(z)}; F\right)$, where $I(A, B)$ is the mutual information (MI) between A and B.

The sketch of the architecture we consider in our work is depicted in Figure 3. We found that, for this work, considering only the local Mutual Information was the best choice in terms of performance. Moreover, a *Uniform Prior regularizer*, similarly to the one described in [21], was included to ensure that the features F cover evenly the latent space.

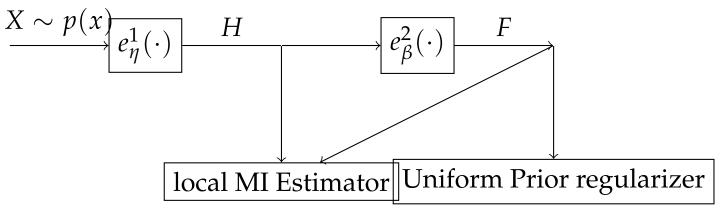

Figure 3. High level scheme of architecture used in this work.

It is possible to show ([21,22], and also Appendix A) that a Local MI estimator can be built as depicted in Figure 4.

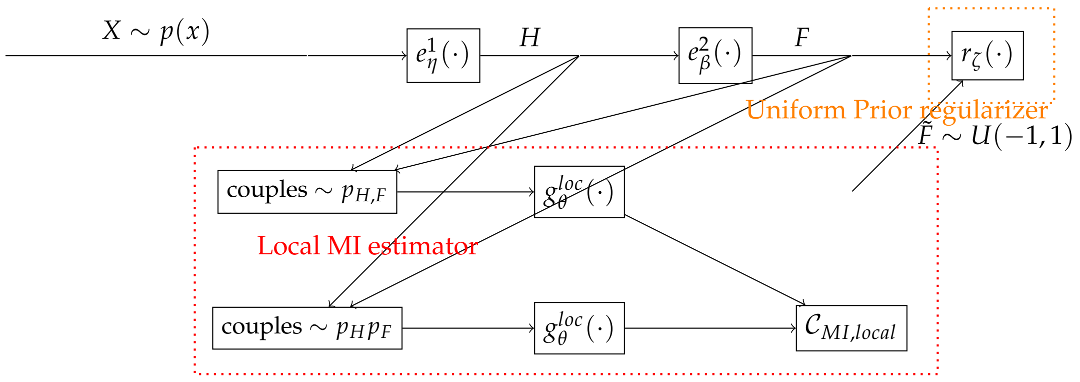

Figure 4. Detailed representation of high level scheme depicted in Figure 3.

Couples extracted from the joint and marginal distributions of H, F are built (respectively $p(H, F)$, $p(H)p(F)$). These couples are fed to two copies of the same network $g_{\theta}^{loc}(\cdot)$ whose output is the input of a cost function described by Equation (A7). Notice that, to build couples sampled from the joint distribution $p(H, F)$, it is sufficient to observe joint realizations of variables H, F, while, to build samples whose distribution is the product of the marginals $p(H)p(F)$, we can instead randomly shuffle one of the two sets of variables H or F and build new couples with the scrambled versions.

To perform regularization instead, always Figure 4, a discriminator network is fed with both the random variables uniformly generated and the features F, and trained to distinguish among the two. The rest of the system is trained adversarially [27] to fool the discriminator, and it is possible to show that, in an equilibrium regime, the distribution of F is the multivariate uniform distribution.

The whole process serves the purpose of extracting high quality and informative feature vectors F in an unsupervised way (i.e., no external labels are needed), which can be used for other downstream tasks such as classification or clustering.

4. The Data Collections

In this work, we use the same datasets as in [11]. The dataset is composed by a multivariate 10-dimensional time series composed of the following features: I and Q correlator outputs, sometimes referred as GNSS raw data; Signal Intensity (S_I), computed as $S_I = I^2 + Q^2$; the raw phase ζ of the received signal; the amplitude scintillation index S_4; the C/N_0; the satellite elevation and azimuth angles; the de-trended measure of carrier raw phase ϕ; and the phase scintillation index σ_{ϕ}. I, Q, the raw phase and consequently S_I are provided at 50 Hz; the other observables are provided at 1 Hz. The overall dataset is then built by performing a moving average with integration time of 60 s and downsampling the high-rate features at 1 Hz. Before processing the data with neural networks a standard normalization, mean removal and division by standard deviation feature by feature has been performed.

Data have been collected in Hanoi, Vietnam (11° 20′ N geo-magnetic latitude on 26 March and 2 April 2015. The measurements have been recorded by means of a Software Defined Radio (SDR) receiver and data grabber, as detailed in [28,29]. L1 C/A signals of 20 Global Positioning System (GPS) satellites were considered, corresponding to 169955 entries, recorded over a time interval of approximately 6 h, with a scintillation rate of 1:4. The ground truth label has been determined manually, based on the visual inspection of the observables.

To avoid confusion, it is important to underline that all the datasets used in this work have been manually annotated, as a ground truth is needed to measure classification accuracy performance. In our experiments, we artificially remove some of the labels to simulate a semi-supervised learning scenario.

5. Overview of Functioning and System Description

As mentioned earlier, one of our goals is to be able to harvest the information contained in the temporal correlation among features. For this reason, we consider as input of the system sliding windows of a multivariate serie X_n built as described in Section 4. The input multivariate time series has thus cardinality equal to 10, or, in neural network language, the number of input channels is 10.

If we denote the sliding window size as N_{win}, we can formally define the input as the $[10 \times N_{win}]$ matrix :

$$X_n = \begin{bmatrix} X^1_{n-N_{win+1}} & X^1_{n-N_{win+2}} & \dots & X^1_{n-1} & X^1_n \\ \dots & \dots & \dots & \dots & \dots \\ X^{10}_{n-N_{win+1}} & X^{10}_{n-N_{win+2}} & \dots & X^{10}_{n-1} & X^{10}_n \end{bmatrix}^T, \tag{1}$$

where n is the last time instant of the window and X^j is the j_{th} feature We decided to use a causal time window such that the method could be applied to a real-time detection scenario. We allow for a delay, such as in a post processing application, and define the window as

$$X_n = \begin{bmatrix} X^1_{n-\left(\frac{N_{win}-1}{2}\right)} & \dots & X^1_n & \dots & X^1_{n-\left(\frac{N_{win}+1}{2}\right)} \\ \dots & \dots & \dots & \dots & \dots \\ X^{10}_{n-\left(\frac{N_{win}-1}{2}\right)} & \dots & X^{10}_n & \dots & X^{10}_{n-\left(\frac{N_{win}+1}{2}\right)} \end{bmatrix}^T . \tag{2}$$

The performance will improve. Notice that in terms of coding the change is trivial with respect to what we implemented, a simple offset in the indexing is sufficient to switch between the two modalities. Notice also that for this second implementation N_{win} must be odd. In this work, we consider the first variant and select $N_{win} = 64$. To test the performance of the proposed method, the entire dataset is split into a training part (90%) and a testing part (10%), using the same procedure as in [11].

During training, the sliding windows are processed and compressed according to a variation of DeepInfomax scheme [21] (highlighted in Section 3 and expanded in Appendix A) that produces a feature vector. As previously introduced, this first stage of training is completely unsupervised and, for our application, the purpose of this process is to create the lower cardinality equivalent feature vector that will be used as input of the classifier. To make the picture clearer, the cardinality of the raw input is the size to the 10-dimensional sliding windows, i.e., $10 \times N_{win} = 640$, while, in this work, we found that a well performing feature vector cardinality ($|F|$) is 20. Thus, for all the time instants for which the scintillation label is available, a classifier is trained with the corresponding feature vector. Notice that the training of compression scheme and classifier is jointly performed end to end using backpropagation.

Figure 5 depicts the global architecture of the system used during training. In the spirit of focusing on the important high level description of the method, all the details related to actual networks architectures and optimizer details are relegated to Appendix B. As anticipated, the first part of the scheme is related to the unsupervised feature extraction, while the second part is devoted to classification of labeled datapoints. The whole procedure can be summarized as follows, considering a batch based training:

1. A batch of input time windows $\{X_r\}$ is is sampled from the training set.
2. The inputs are processed by a first convolutional neural network $e^1_\eta(\cdot)$ and a first level compression is performed, generating variables $\{H_r\}$.
3. Through $e^2_\beta(\cdot)$, a second neural network with both convolutional and fully connected layers, the final features $\{F_r\}$ are built.
4. Starting from this point of the computation chain, three different parallel processes are performed:

   (i) **classification** of features $\{F_r\}$ to determine whether a given time instant r corresponds to scintillation. To be completely precise, as explained in Appendix B, the actual input to the classifier is the concatenation of feature vector F and the result of skip connections taken from the layers of the first convolutional network $e^1_\eta(\cdot)$. This is done only for the time instants for which there is a reference. For all time instants for which a reference signal exists, the corresponding feature F_r is fed to fully connected classifier $c_\psi(\cdot)$ that receives as reference signal Y_r, the corresponding label scintillation/no scintillation for time instant r.
   (ii) **local Mutual Information (MI)** computation: starting from the collections of $\{(H_r, F_r)\}$ couples extracted from the joint and marginal distributions are built. These couples are fed to the network $g^{loc}_\theta(\cdot)$ whose output is the input of the $\mathcal{C}_{MI,local}$ cost function, defined in (A7).
   (iii) **adversarial matching** of features F distribution to a target uniform distribution. Random variables $\tilde{F} \sim U(-1, 1)$ are generated. A discriminator is fed with both the random variables generated and the F (the rest of the system is trained adversarially [27] to fool the discriminator). This is used as a regularized to ensure that the features F cover evenly the latent space.

5. Standard backpropagation is performed, the cost function considered is $\mathcal{C} = \mathcal{C}_{MI,local} + \mathcal{C}_{adversarial} + \mathcal{C}_{classification}$, where the subscripts represent the various component of the cost function.

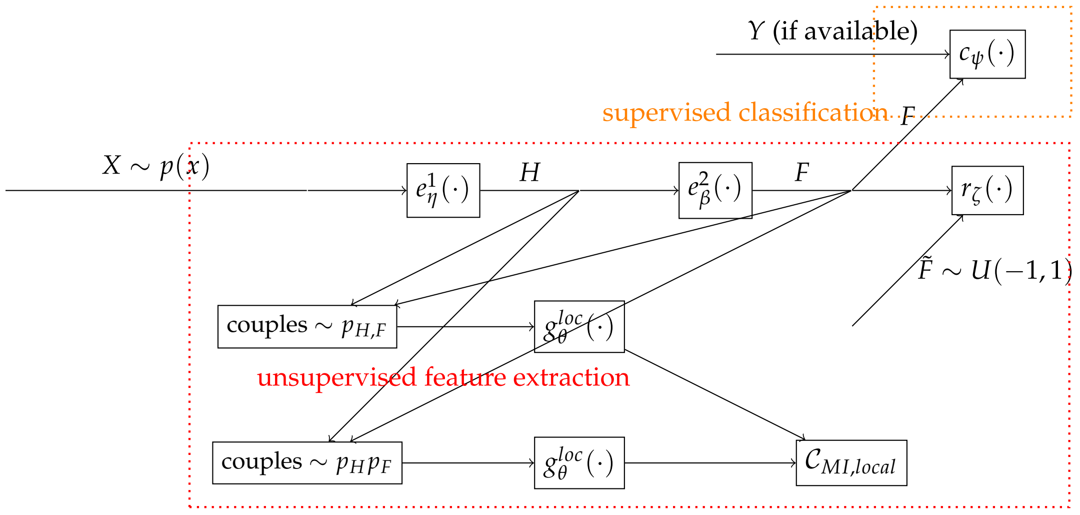

Figure 5. Neural Network architecture (Training). The dashed red box contains the unsupervised feature extraction part, while the orange one contains the supervised classification.

During testing, the only portion of the network that is used is the one related to classification, for simplicity depicted in Figure 6 that is clearly much simpler and lighter. In a realistic scenario, the training could be then performed offline on powerful machines while the lighter scintillation detection system could be subsequently uploaded on SDR receivers where the computational capabilities and the maximum power consumption are stricter.

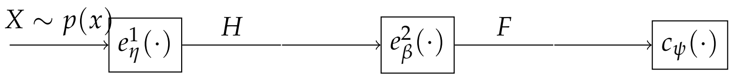

Figure 6. Neural Network architecture (Testing).

6. Experiments

In this section, we present experimental results of the DeepInfomax method, compared with the Decision Tree proposed in [11], varying the number of labeled datapoints, by comparing the classification accuracy with respect to the baseline.The dataset used for the experiments is the same as in [11], but we neither train or test the method on the first 63 s of each data collection, due to the necessity of having a full window of samples. For each of the experiments, the labeled datapoints are randomly extracted from the training set. In the following, we present results in terms of standard machine learning performance indicators that we hereafter define. Denote as C the true class to which a datapoint belongs (0 for non scintillation, 1 for scintillation) and as $\hat{C}$ the estimated class, i.e., the output of the classifier. We can define four joint probabilities:

- true positive: $T_p = P\left(\hat{C} = 1, C = 1\right)$,
- false positive: $F_p = P\left(\hat{C} = 1, C = 0\right)$,
- true negative: $T_n = P\left(\hat{C} = 0, C = 0\right)$,
- false negative: $F_n = P\left(\hat{C} = 0, C = 1\right)$.

Derived from these four quantities are the following ones:

- accuracy: $A = T_p + T_n$,
- precision: $P = \frac{T_p}{T_p+F_p}$,
- recall: $R = \frac{T_p}{T_p+F_n}$,
- F-score: $F_{sc} = 2\frac{RP}{R+P}$.

Figure 7 reports the main result of this paper: a comparison between the proposed method and the decision tree based one, varying the number of labeled datapoints. For sake of completeness, it is furthermore necessary to clarify that, when dealing with the decision tree classification algorithm, the number of labeled datapoints (x-axis of Figure 7) coincides with the size of the dataset used for training, since there is no possibility to include unlabeled datapoints in the learning pipeline. Instead, with our approach, the size of the training dataset is always the same (90% of the complete dataset, roughly 150,000 datapoints) but only a portion of the datapoints is labeled.

The improvement in terms of performance is evident. First, it is worth noticing that the proposed DeepInfomax algorithm outperforms the Decision Tree technique, both in terms of accuracy and of F-score, for the same number of labeled elements. Second, a much lower number of labeled datapoints is required when using DeepInfomax to attain similar levels of accuracy and F-score. As the number of labeled points increases, the accuracy reaches a top value on the test set of 0.9993 for DeepInfomax, compared to the value 0.9824 for the Decision Tree.

It is important to notice that, even if from a performance point of view, the results are excellent, to attest to the usefulness of the method in a realistic application tests on many different datasets should be performed. It is possible, in fact, that bias is present in the considered data collection on which the networks base their decisions. However, this problem is complex and outside the scope of this paper; see [30].

Table 1 reports the results of the method for all the performance metrics introduced above. As expected, the performance increases with the increase of number of labeled datapoints, rapidly saturating to excellent levels. It is interesting to notice that the amount of false positives and false negatives is almost equivalent.

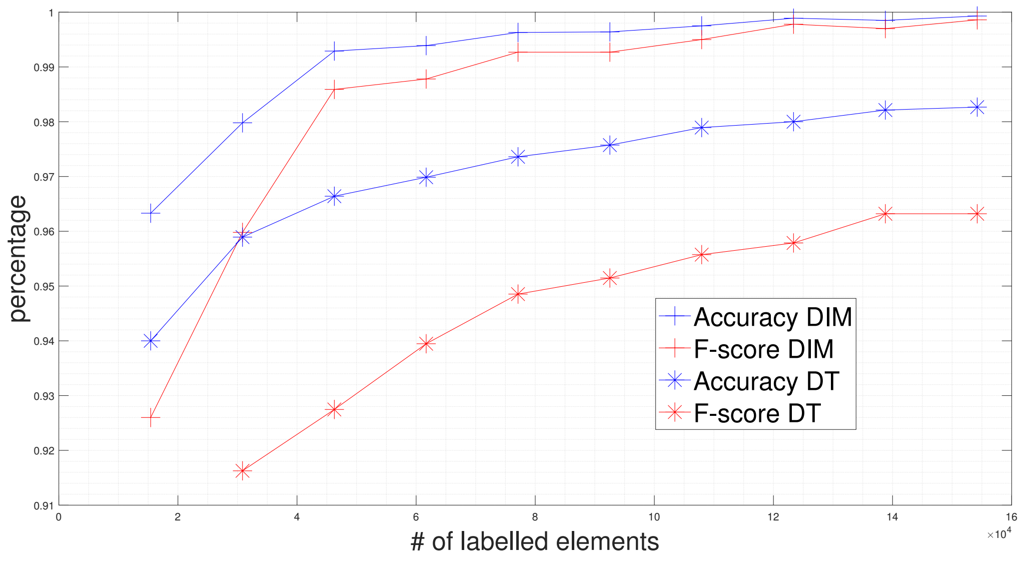

Figure 7. Accuracy and F-score of Deep Infomax (DIM) method and Decision Tree (DT) [11] method.

Table 1. Varying of the considered performance metrics versus percentage of labeled datapoints (% labeled).

% Labeled	Accuracy	Precision	Recall	F-Score	T_n	F_p	F_n	T_p
10	0.9633	0.9254	0.9267	0.9260	0.7339	0.0185	0.0182	0.2294
20	0.9798	0.9757	0.9444	0.9598	0.7389	0.0060	0.0142	0.2409
30	0.9929	0.9858	0.9860	0.9859	0.7463	0.0036	0.0035	0.2466
40	0.9939	0.9825	0.9932	0.9878	0.7453	0.0044	0.0017	0.2486
50	0.9963	0.9926	0.9928	0.9927	0.7459	0.0019	0.0018	0.2504
60	0.9964	0.9924	0.9929	0.9927	0.7511	0.0019	0.0018	0.2452
70	0.9975	0.9951	0.9949	0.9950	0.7485	0.0012	0.0013	0.2490
80	0.9989	0.9984	0.9972	0.9978	0.7494	0.0004	0.0007	0.2494
90	0.9985	0.9957	0.9984	0.9970	0.7438	0.0011	0.0004	0.2546
100	0.9993	0.9986	0.9986	0.9986	0.7521	0.0004	0.0004	0.2472

7. A Qualitative View on Features Importance

Neural networks are known for being black box models, in which obtaining human understandable information from the inner functioning is extremely complex, if not impossible. Several techniques have been proposed to determine and inspect feature importance on networks decisions, usually with specific methods tailored to particular experiments and particular network architectures.

In this subsection, we present results on features importance based on the simple but powerful features random permutation technique (similarly to the method introduced in [31]). The idea is to consider a trained classifier, randomly shuffle the variables one by one, and record the decrease in terms of performance. The measured decrease for all the shuffling experiments with respect to the baseline (the unshuffled dataset) is used as a proxy to determine the shuffled feature importance. Notice that this investigation strategy somehow lacks the capability of measuring joint higher order importance of features. In fact, two or more features can be extremely correlated and important for the considered task, but shuffling just one of them possibly does not impact performance in a substantial way, producing thus an underestimation of the feature importance.

In Figure 8, we report the results of permutation tests on the classifier trained with percentage of labeled examples equal to 10%. Despite the possible limitations of the considered measurement technique, we obtain results that are consistent with the physical knowledge of the problem as well as previous results [11].

The features in order of importance (computed as the decrease in terms of accuracy with respect to the baseline) are: $[S_I, \theta_{el}, S_4, C/N_0, \zeta, \theta_{az}, I, Q, \sigma_\phi, \phi]$. In the considered data collections, amplitude scintillations' events are observed. It is thus reasonable to assess that Signal Intensity, S_I, is the most important feature, with a drop in accuracy of more than 40% when the feature is randomly permuted. The three features $\theta_{el}, S_4, C/N_0$ when permuted induces basically the same drop in terms of accuracy. Elevation angle θ_{el} is informative for classification: signal statistics for low elevation satellites greatly vary from the high in sky ones. The amplitude scintillation index S_4 is clearly an important feature since amplitude scintillations events are considered. Finally, the C/N_0 is yet another indirect measurement of signal intensity. Notice, moreover, that $\theta_{el}, S_4, C/N_0$ are the three features used for the so-called semihard detection rule [11]. The raw phase and the azimuth angle appear to contain less information about the scintillation events and the features I, Q, σ_ϕ, ϕ have the lowest impact when randomized. It is possible that the relevant information contained in I, Q is already summarized in S_I, while being the scintillation events linked to amplitude fluctuations the scintillation phase information is not that important.

A complete analysis of feature importance would require to retrain from scratch different versions of the classifier by completely excluding some of the features from the beginning of training, but, since this is computationally expensive and outside the scope of this paper, we leave this analysis for future works.

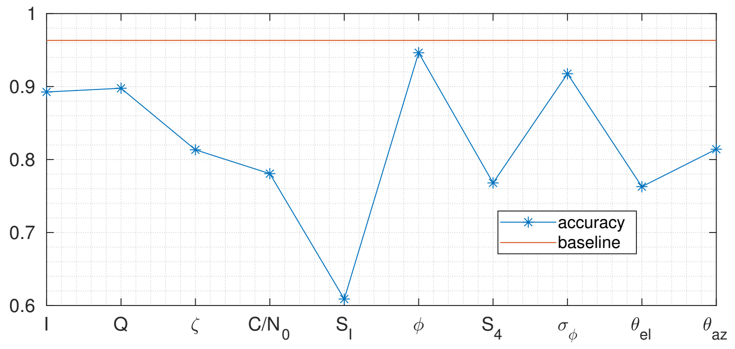

Figure 8. Feature importance analysis using random permutation method. The lower the accuracy with respect to the baseline, the higher the feature importance.

8. Conclusions

In this paper, we propose a new semi-supervised architecture for GNSS scintillations detection based on the DeepInfomax approach. We showed that it is possible to reduce by a large amount the number of labeled datapoints to achieve a target performance, decreasing the amount of time human researcher must spend manually labelling datasets. Thanks to the considered architecture, it is possible to process temporal windows of features empowering the capabilities of the model. Furthermore, the overall accuracy and F-score are improved when compared to other machine learning techniques. Future directions include testing of the trained system on datasets as different as possible, in order to verify the generalization capabilities of the proposed method. Moreover, we will optimize the system to further reduce the amount of supervision in the view of switching to an almost fully unsupervised algorithm.

Author Contributions: Conceptualization, G.F., N.L. and F.D.; methodology, G.F., N.L. and F.D.; software, G.F., N.L. and F.D.; validation, G.F., N.L. and F.D.; formal analysis, G.F., N.L. and F.D.; investigation, G.F., N.L. and F.D.; resources, G.F., N.L. and F.D.; data curation, G.F., N.L. and F.D.; writing–original draft preparation, G.F., N.L. and F.D.; writing–review and editing, G.F., N.L. and F.D.; visualization, G.F., N.L. and F.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank Rayan Imam for the technical help with the preprocessing of the dataset.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Variational Representation of Divergence and Deep Infomax

In this appendix, we briefly review the concept of convex divergence between probability distributions, the particular case of mutual information (MI), and its usage for the construction of a feature extractor.

The class of f-convex divergences between distributions of two random variables $X_1 \sim p(x)$, $X_2 \sim q(x)$ is defined as

$$\mathrm{D}(p||q) = \int p(x) f\left(\frac{q(x)}{p(x)}\right) dx. \tag{A1}$$

Besides from very special cases, such as multivariate Gaussian distributions, and simple f functions, no closed form exists for the computation. Moreover, in general, not even a parametric model about the distributions p, q is known, but all the information we have about the density functions is related to same empirical samples $\{x_1^i\}_{i=1}^{N_1}, \{x_2^j\}_{j=1}^{N_2}$ drawn from the corresponding distributions.

The main idea of variational representation is that, roughly speaking, thanks to convex analysis, we can write the divergence as

$$\mathrm{D}(p||q) = \sup_{g(x)\in\mathcal{G}} \left\{ \int p(x)g(x)dx - h\left(\int q(x)t(g(x))dx \right) \right\}, \tag{A2}$$

where h, t are analytically known functions that can be computed once f is fixed. Thanks to this representation, we can estimate the divergence without knowing the actual distributions p, q using Monte Carlo methods:

$$\mathrm{D}(p||q) \simeq \sup_{g(x)\in\mathcal{G}} \left\{ \sum_{i=1}^{N_1} g(x_1^i) - h\left(\sum_{j=1}^{N_2} t(g(x_2^j)) \right) \right\}. \tag{A3}$$

An attentive reader might notice that since there is the problem of extremization $\sup_{g(x)\in\mathcal{G}}$, we just moved the complexity of the problem to another point. The main idea discussed in [22], is that, however, what holds however is that

$$\mathrm{D}(p||q) = \sup_{g(x)\in\mathcal{G}} \{\cdot\} \geq \sup_{g(x)\in\mathcal{N}} \{\cdot\},$$

where the space $\mathcal{N} \subset \mathcal{G}$ is chosen such that we can perform "easily" optimization (such as the space of Neural Networks, i.e., $g_\theta(x)$ is a neural network with parameters θ). We can finally estimate the divergence with the following approximation by optimizing (i.e., training) the network and using Monte Carlo techniques

$$\mathrm{D}(p||q) \simeq \sup_{\theta} \left\{ \sum_{i=1}^{N_1} g_\theta(x_1^i) - h\left(\sum_{j=1}^{N_2} t(g_\theta(x_2^j)) \right) \right\}. \tag{A4}$$

If g_θ is sufficiently flexible, we can get arbitrarily close to the true value of the divergence. Notice that, since it is possible to compute the gradients of the functions g, h, t, end to end training is possible. In the particular case where the considered divergence is the mutual information (MI) between two random variables X_1 and X_2, and that thus the dataset is composed $\{(x_1, x_2)^i\}_{i=1}^N$, this corresponds to considering

$$\mathrm{I}\,(X_1; X_2) = \mathrm{D}(p(x_1, x_2)||p(x_1)p(x_2)) = \int p(x_1, x_2) \log\left(\frac{p(x_1, x_2)}{p(x_1)p(x_2)} \right) dx_1 dx_2. \tag{A5}$$

That can be approximated as

$$\mathrm{I}\,(X_1; X_2) \simeq \sup_{\theta} \left\{ \sum_{i=1}^{N} g_\theta((x_1, x_2)^i) - h\left(\sum_{j=1}^{N} t(g_\theta((x_1, x_2)^j_{scrambled})) \right) \right\}, \tag{A6}$$

where $(x_1, x_2)^i$ are simply samples taken from the dataset of joint variables (and thus extracted from $p(x_1, x_2)$) while $(x_1, x_2)^j_{scrambled}$ are derived by randomly scrambling the indices of one of the two random variables in the dataset (thus having from a practical point of view statistical distribution equal to $p(x_1)p(x_2)$). Remember that the functions $h(\cdot), t(\cdot)$ are known. Actually, in practice [21], maximizing instead the Jensen–Shannon Divergence (JSD) between $p(x_1)p(x_2)$ and $p(x_1)p(x_2)$ works better. After some manipulations, the final result is that the divergence can be simply written as:

$$\mathrm{JSD}\,(X_1; X_2) \simeq \sup_{\theta} \left\{ \sum_{i=1}^{N} -s_p\left(-g_\theta((x_1, x_2)^i) \right) - \left(\sum_{j=1}^{N} s_p\left(g_\theta((x_1, x_2)^j_{scrambled}) \right) \right) \right\}, \tag{A7}$$

where $s_p(z) = \log(1 + \exp(z))$ is called soft-plus function.

Upon this idea in [21], they built an unsupervised compression tool of structured high dimensional random variables X (such as images or time series such as voice, etc...), called by the authors *Deep*

Infomax. The (oversimplified) description of the system, schematically depicted in Figure A1 is that, using two neural network encoders $e^1_\eta(\cdot), e^2_\beta(\cdot)$ and two discriminator networks $g^{glob}_\theta(\cdot), g^{loc}_\theta(\cdot)$, it is possible to produce two latent representations H, F, respectively computed as $H = e^1_\eta(X), F = e^2_\beta(H)$, such that both the MI between H, F and the Local Mutual Information $MI_{loc} = \frac{1}{|H|}\sum_{z=1}^{|H|} \mathrm{I}\left(H^{(z)}; F\right)$ (a new metric introduced by the authors of [21]) are maximized. We moreover clarify the role of $r_\zeta(\cdot)$ that is a discriminator network that tries to distinguish between samples drawn from F and samples drawn from a multivariate uniform distributions ($\tilde{F} \sim U(-1,1)$). The system is trained adversarially [27], where $r_\zeta(\cdot)$ and the rest of the network are the two competing agents (at equilibrium F has uniform distribution and this property is used to ensure that the latent variable covers evenly the latent space as a form of regularization). Notice that, in our implementation (see Section 5 and Appendix B), we only considered local mutual information and did not implement the portion of the system linked to the global, i.e., classical, mutual information.

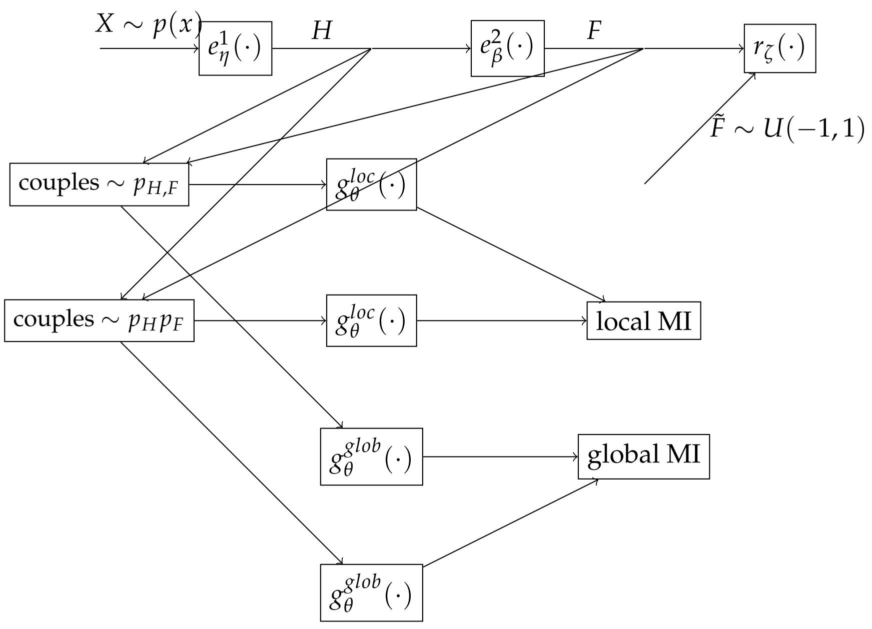

Figure A1. Deep Infomax architecture.

Appendix B. Neural Networks' Architectures

In this appendix, we describe the details of the various networks as well as the optimizer settings. All the software has been written using MATLAB® (R2019a, The MathWorks, Inc.). In our notation, a single line of a table represents a layer. The convention is the following: at the beginning of the row, we have the type of layer as well as the nonlinearity associated with the layer (if present) and the indication of presence of BatchNorm (if the BN acronym is present). We have five types of layers in our network:

1. Input layer. It simply indicates the input of the system, the matrix X_n.
2. 1d convolutional layer (indicated as conv_1d). Its parameters are the number of taps of the filters (Ntaps), the number of filters (Nch) as well as the stride (the decimation period at the output of the layer).
3. residual 1d convolutional layers: as standard conv_1d layers but with a residual connection (another set of filters that bypass the nonlinearity and whose output is summed to the output of the nonlinearity).

4. flatten layer. It simply indicates that the input of this layer (a multivariate time serie) is flattened (vectorized).
5. fully connected layers, whose argument is the dimension of the output.

We consider as nonlinearities the rectified linear unit (ReLu), the sigmoid (Sigm), and the hyperbolic tangent (Tanh). In the first convolutional network ($e^1_\beta(\cdot)$), we included skip connections of the last element of the various sequences at the two layers that are fed to the fully connected classifier $c_\psi(\cdot)$.

All networks are trained using Adam optimizer with default settings and learning rate 0.001 for 30 epochs with batch size equal to 64.

Table A1. Architecture of $e^1_\beta(\cdot)$.

input layer (size 64 × 1 × 10) (layer corresponding to the input X)
res_conv1d (Ntaps = 4, Nch = 64, stride = 2), ReLu, BN
res_conv1d (Ntaps = 4, Nch = 128, stride = 2), ReLu, BN (the output is H)

Table A2. Architecture of $e^2_\beta(\cdot)$.

res_conv1d (Ntaps = 4, Nch = 64, stride = 2), Relu, BN
res_conv1d (Ntaps = 4, Nch = 128, stride = 2), Relu, BN
flatten layer
fully conncected (512), Relu, BN
fully conncected (512), Relu, BN
fully conncected (20),Tanh (the output is F)

Table A3. Architecture of $g^{loc}_\theta(\cdot)$.

conv1d (Ntaps = 1, Nch = 64, stride = 1), Relu
conv1d (Ntaps = 1, Nch = 64, stride = 1), Relu
conv1d (Ntaps = 1, Nch = 1, stride = 1), Sigm

Table A4. $r_\zeta(\cdot)$.

fully conncected (1024), Relu, BN
fully conncected (512), Relu, BN
fully connected (1), Sigm

Table A5. Architecture of $c_\psi(\cdot)$.

fully conncected (512), ReLu, BN
fully conncected (512), ReLu, BN
fully conncected (1), Sigm (the output is $\hat{C}$)

References

1. Moreno, B.; Radicella, S.; De Lacy, M.; Herraiz, M.; Rodriguez-Caderot, G. On the effects of the ionospheric disturbances on precise point positioning at equatorial latitudes. *GPS Solut.* **2011**, *15*, 381–390. [CrossRef]
2. Skone, S.; Knudsen, K.; De Jong, M. Limitations in GPS receiver tracking performance under ionospheric scintillation conditions. *Phys. Chem. Earth Part A Solid Earth Geod.* **2001**, *26*, 613–621. [CrossRef]
3. Yeh, K.C.; Liu, C.H. Radio wave scintillations in the ionosphere. *Proc. IEEE* **1982**, *70*, 324–360.
4. Kintner, P.; Ledvina, B.; De Paula, E. GPS and ionospheric scintillations. *Space Weather* **2007**, *5*. [CrossRef]

5. Lee, J.; Morton, Y.; Lee, J.; Moon, H.; Seo, J. Monitoring and Mitigation of Ionospheric Anomalies for GNSS-Based Safety Critical Systems: A review of up-to-date signal processing techniques. *IEEE Signal Process. Mag.* **2017**, *34*, 96–110. [CrossRef]
6. Taylor, S.; Morton, Y.; Jiao, Y.; Triplett, J.; Pelgrum, W. An improved ionosphere scintillation event detection and automatic trigger for GNSS data collection systems. In Proceedings of the Proceedings of the 2012 International Technical Meeting of The Institute of Navigation, Newport Beach, CA, USA, 30–31 January 2012.
7. Linty, N.; Dovis, F. An Open-Loop Receiver Architecture for Monitoring of Ionospheric Scintillations by Means of GNSS Signals. *Appl. Sci.* **2019**, *9*, 2482. [CrossRef]
8. Jiao, Y.; Hall, J.J.; Morton, Y.T. Automatic equatorial GPS amplitude scintillation detection using a machine learning algorithm. *IEEE Trans. Aerosp. Electron. Syst.* **2017**, *53*, 405–418. [CrossRef]
9. Jiao, Y.; Hall, J.J.; Morton, Y.T. Performance Evaluation of an Automatic GPS Ionospheric Phase Scintillation Detector Using a Machine-Learning Algorithm. *Navig. J. Inst. Navig.* **2017**, *64*, 391–402. [CrossRef]
10. Liu, Y.L.; Morton, Y.J.; Jiao, Y.J. Application of machine learning to the characterization of GPS L1 ionospheric amplitude scintillation. In Proceedings of the 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, CA, USA, 23–26 April 2018; pp. 1159–1166.
11. Linty, N.; Farasin, A.; Favenza, A.; Dovis, F. Detection of GNSS Ionospheric Scintillations based on Machine Learning Decision Tree. *IEEE Trans. Aerosp. Electron. Syst.* **2019**, *55*, 303–317. [CrossRef]
12. Favenza, A.; Farasin, A.; Linty, N.; Dovis, F. A machine learning approach to GNSS scintillation detection: automatic soft inspection of the events. In Proceedings of the 30th International Technical Meeting of the Satellite Division Of the Institute of Navigation (ION GNSS+ 2017), Portland, OR, USA, 25–29 September 2017; pp. 4103–4111.
13. Savas, C.; Dovis, F. Comparative performance study of linear and Gaussian kernel SVM implementations for phase scintillation detection. In Proceedings of the IEEE International Conference on Localization and GNSS (ICL-GNSS), Nuremberg, Germany, 4–6 June 2019; pp. 1–6. [CrossRef]
14. Savas, C.; Dovis, F. The Impact of Different Kernel Functions on the Performance of Scintillation Detection Based on Support Vector Machines. *Sensors* **2019**, *19*, 5219. [CrossRef]
15. Rezende, L.; De Paula, E.; Stephany, S.; Kantor, I.; Muella, M.; de Siqueira, P.; Correa, K. Survey and prediction of the ionospheric scintillation using data mining techniques. *Space Weather* **2010**, *8*, 1–10. [CrossRef]
16. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 30 December 2019).
17. de Lima, G.; Stephany, S.; de Paula, E.; Batista, I.; Abdu, M. Prediction of the level of ionospheric scintillation at equatorial latitudes in Brazil using a neural network. *Space Weather* **2015**, *13*, 446–457. [CrossRef]
18. Uwamahoro, J.C.; Habarulema, J.B. modeling total electron content during geomagnetic storm conditions using empirical orthogonal functions and neural networks. *J. Geophys. Res. Space Phys.* **2015**, *120*, 11000–11012. [CrossRef]
19. Habarulema, J.B.; McKinnell, L.A.; Opperman, B.D. Regional GPS TEC modeling; Attempted spatial and temporal extrapolation of TEC using neural networks. *J. Geophys. Res. Space Phys.* **2011**, *116*. [CrossRef]
20. Kingma, D.P.; Mohamed, S.; Rezende, D.J.; Welling, M. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2014; pp. 3581–3589.
21. Hjelm, R.D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Trischler, A.; Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv* **2018**, arXiv:1808.06670.
22. Belghazi, M.I.; Baratin, A.; Rajeswar, S.; Ozair, S.; Bengio, Y.; Courville, A.; Hjelm, R.D. Mine: Mutual information neural estimation. *arXiv* **2018**, arXiv:1801.04062.
23. Nguyen, X.; Wainwright, M.J.; Jordan, M.I. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Trans. Inf. Theory* **2010**, *56*, 5847–5861. [CrossRef]
24. Ruderman, A.; Reid, M.; García-García, D.; Petterson, J. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv* **2012**, arXiv:1206.4664.
25. Nowozin, S.; Cseke, B.; Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2016; pp. 271–279.

26. Poole, B.; Ozair, S.; Oord, A.v.d.; Alemi, A.A.; Tucker, G. On variational bounds of mutual information. *arXiv* **2019**, arXiv:1905.06922.
27. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2014; pp. 2672–2680.
28. Curran, J.T.; Bavaro, M.; Morrison, A.; Fortuny, J. Developing a multi-frequency for GNSS-based scintillation monitoring receiver. In Proceedings of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+), Tampa, FL, USA, 8–12 September 2014; pp. 1142–1152.
29. Cristodaro, C.; Dovis, F.; Linty, N.; Romero, R. Design of a configurable monitoring station for scintillations by means of a GNSS software radio receiver. *IEEE Geosci. Remote. Sens. Lett.* **2018**, *15*, 325–329. [CrossRef]
30. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
31. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]