2024 IEEE Globecom Workshops (GC Wkshps): Workshop on Mobile Network Enabled Synergies of Communication, Sensing and Computing Toward 6G

Opportunistic Sensing with Satellite Communications using a Conditional Ensemble Neural Network

G. Scognamiglio*, G. Bacci†, A. Rucci*, A. Vaccaro*, F. Sapienza†, and F. Giannetti†
*MBI srl, Pisa, Italy ({gscognamiglio, arucci, avaccaro}@mbigroup.it)
†Dip. Ing. dell'Informazione, University of Pisa, Pisa, Italy ({giacomo.bacci, fabiola.sapienza, filippo.giannetti}@ing.unipi.it)

***Abstract*—Rainfall estimation, a key task for enabling sustainable and intelligent living, has traditionally been carried out using dedicated devices. Recently, machine learning (ML) applications exploiting opportunistic satellite-to-earth microwave links (SMLs) have gained traction. However, significant deep learning implementations to perform end-to-end regression from raw signal-to-noise ratio (SNR) to rainfall estimation remain underexplored. Leveraging a proprietary satellite receiver network, we address this shortcoming by building a conditional ensemble neural network (a deep learning classifier-regressor chain) and comparing its performance, both in terms of detection and regression, with a state-of-the-art power-law (PL)-based algorithm. Our neural network shows significant gains in both tasks. Regarding the classification task, the gains are marginal: approximately a 3.5%-increase in both accuracy and F1-score. On the other hand, for the regression task, the performance gains are more substantial: a 34%-decrease in normalized mean absolute error (NMAE) and a 70%-decrease in mean absolute percentage error (MAPE) on event-based cumulative precipitation predictions.**

I. Introduction and motivation

Smart monitoring of the environment has become one of the pillars enabling intelligent living [1]. Real-time, precise precipitation measurements are of great interest to decision makers in a wide range of applications, from agricultural planning to flood control, environmental analysis, and public safety [2]–[4]. Yet, capturing precise real-time rainfall data remains arduous due to extensive spatial and temporal variations [5].

In the past decade, novel techniques focusing on quantifying rain-induced attenuation in existing radio communication networks have gained traction. Most methods leverage the fact that electromagnetic waves are vulnerable to dispersion and absorption by weather elements [6]. These radio networks include both commercial microwave links (CMLs) (cellular backhaul connections) and satellite-to-earth microwave links (SMLs). Most academic work uses a power-law (PL)-based model to describe the connection between rain rate and signal loss in microwave links. CMLs for rain estimation have been extensively studied and showcase both PL- and machine learning (ML)-based methods [7]–[9], with the latter providing superior performance [10], [11]. In the context of SMLs, PL-based applications have been extensively researched [12], while ML-based methods remain less explored.

Over the past few years, dense neural networks (DNNs) have been successfully applied to various sensing challenges, both in terms of improved performance and efficient resource utilization [13], [14]. Recently, ML-based applications on SMLs have emerged. In [15], a DNN is applied to statistics extracted from the raw attenuation signal, in order to distinguish between dry and wet periods. For rainfall estimation, [15] leveraged the well-established PL relationship, learning parameters via empirical risk minimization (ERM). This approach, incorporating a shallow DNN followed by an empirical model for rainfall inversion, was similarly adopted in [16]. In [17], a variety of shallow machine learning models are applied to raw signal statistics for rainfall identification. Ref. [18] suggested using a long short term memory (LSTM) architecture for rain event identification. All these studies mainly used ML to classify dry vs. rain periods, relying on the traditional PL for rainfall estimation regression. Notably, most deployed ML models were shallow nets with a single hidden layer.

In this work, we address the limited ML-based applications for SML-based rainfall estimation, and compare our deep learning model with a state-of-the-art PL-based model. In this contribution, we make use of opportunistic communication signals received from fixed satellite devices. However, the methods derived here are fairly general, and can be applied to any satellite receivers, including mobile ones. We introduce a conditional ensemble neural network (CENN) composed of an ensemble classifier and a lightweight regressor model. We analyse and compare its performance with a state-of-the-art PL-based model in terms of detection capabilities (classification task) and rainfall estimation (regression task). The main contributions of this article include:

- introducing a multihead neural network architecture for rain detection using SMLs;
- introducing a conditional piecewise ensemble model for the rainfall estimation regression task; and
- analyzing the performance of the proposed network architecture in comparison to a PL-based method.

The remainder is structured as follows. Sect. II provides a relevant background on neural architectures, the dataset used, main engineering features, and a review of PL-based methods chosen for comparison. In Sect. III, we detail our final conditional ensemble model along with the ML methodology used. Experimental results are showcased and discussed in Sect. IV. Finally, conclusions are drawn in Sect. V.

979-8-3315-0567-7/24/$31.00 ©2024 IEEE

II. Background

A. Models

Our conditional ensemble is composed of three sub-models [19]: *i*) a gradient boosting machine (GBM) [20]; *ii*) a DNN [21]; and *iii*) a recurrent neural network (RNN) [22].

i) The GBM builds decision trees in a sequential manner, where each subsequent tree corrects the errors of its predecessor. A decision tree $\mathcal{T}$ consists of nodes $\mathcal{N}$ and edges $\mathcal{E}$. Each node represents a decision point based on a feature x_i from the input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Internal nodes represent tests on features, and each edge $e \in \mathcal{E}$ represents the outcome of a test, leading to child nodes. Leaf nodes $\mathcal{L} \subset \mathcal{N}$ represent the final output value y. At each internal node n, a feature x_i and a threshold θ are chosen to split the data. For a binary split, if $x_i < \theta$, the left child is followed; otherwise, the right child is followed. GBM builds an ensemble of shallow trees by fitting the residual errors of the previous tree rather than the target variable itself. For a mathematical breakdown of GBM, please refer to [20].

ii) The DNN consists of layers of interconnected nodes or neurons. Each connection has an associated weight, which is adjusted during training. When an input is fed into a DNN, it propagates through these layers by undergoing a series of weighted sums and nonlinear activations. The final layer produces the outcome, for instance a classification or a regression value. During the training phase, the network minimizes the error between its predictions and actual true values using an optimization algorithm such as gradient descent [23]. Mathematically, consider an input vector $\mathbf{x} \in \mathbb{R}^n$, where n is the number of features. Each layer l contains neurons (functions) that perform weighted sums of the inputs followed by a nonlinear activation function. For a given l, the output $\mathbf{h}^{(l)}$ is computed as:

$$\mathbf{h}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \tag{1}$$

where $\mathbf{W}^{(l)}$ is the weight matrix for layer l, $\mathbf{b}^{(l)}$ is the bias vector for layer l, and σ is the activation function. The final layer produces the output vector $\mathbf{y}$ using a similar transformation as the hidden layers:

$$\mathbf{y} = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}, \tag{2}$$

where L is the index of the output layer. The DNN learns to make accurate predictions on new data by adjusting the weights at each iterations. Like GBM, the inductive bias of the DNN makes it particularly suitable for tabular data.

iii) An RNN is a class of neural networks designed to process sequential data. Unlike traditional DNNs, RNNs maintain an internal state that captures information about previous inputs, allowing them to model the temporal links. This internal state is updated as the network processes each element of the input sequence. An RNN processes an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ by maintaining a hidden state vector $\mathbf{h}_t$ at each time step t. The hidden state $\mathbf{h}_t$ is updated based on the current input x_t and the previous hidden state $\mathbf{h}_{t-1}$. Mathematically, this is represented as:

$$\mathbf{h}_t = f(\mathbf{W}_{xh}x_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{3}$$

where $\mathbf{W}_{xh}$ is the weight matrix between the input and the hidden state, $\mathbf{W}_{hh}$ is the weight matrix for the hidden state transition, $\mathbf{b}_h$ is the bias term, and f is the activation function. The output y_t at each time step t is given by

$$y_t = g(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y), \tag{4}$$

where $\mathbf{W}_{hy}$ is the weight matrix between the hidden state and the output, $\mathbf{b}_y$ is the bias term for the output, and g is the activation function for the output.

RNNs typically have strong limitation when dealing with very long sequences due to vanishing or exploding gradients in the optimization process [23]. We therefore use a particular type of RNN, called gated recurrent unit (GRU), which partially overcomes the limitation of capturing long-range dependencies by using a set of gates to control information flow [24].

B. Data and ML methodology

The data for this work are taken from the NEFOCAST platform [25] (see [26] for further details), developed within the framework of a research project funded by the Tuscany region (Italy) to assess the viability of a satellite-based real-time precipitation monitoring system. NEFOCAST leverages a dense network of next-generation satellite receivers, named SmartLNBs (SLNBs). The SLNBs are interactive satellite terminals which, in addition to their primary design for satellite services, can act as weather sensors due to their ability to measure the signal-to-noise ratio (SNR) of the downlink signal, and to transmit it via the uplink channel. NEFOCAST thus provides a stream of SML-based SNR values that can be used to infer rain detection and estimation.

The satellite SNR time series have been used as the input feature (independent variable) to train our models, with the associated disdrometer measurements being the target feature (dependent variable). We selected a location in Rome, Italy, that features both an SLNB and a disdrometer. Data were available from July 15, 2021, to July 11, 2023. The selected test period spanned from May 10, 2023, to July 11, 2023.

Regarding data preprocessing, the SLNB SNR time series have a granularity of 30 seconds, while the disdrometer provides a rain-rate measurement every 60 seconds. We thus average the SNR time series to resample them at 60 seconds. This reduces the noise in the SNR and makes the input and target variables have the same temporal resolution. We then divide the target values by 60, to get actual rain precipitation in millimeters (mm) over the 60 seconds span. Furthermore, disdrometer values below $0.5\ \mu\mathrm{m}$ are set to 0. The input variable is normalized with a rolling maximum, followed by a rolling median normalization, as the goal for the model is to recognize patterns that are independent of the absolute scale of the data. It is worth noting that the normalization

only applies to the input data: the ML models will thus learn to map normalized SNR data directly to precipitation in millimeters at a 60 seconds frequency, meaning that no post-transformations need to be applied to the models' output. Regarding feature engineering, we extract statistics on multiple time windows similarly to [16], [17]. This includes rolling window calculations for the median, standard deviation, and skewness and kurtosis. Our approach stands out by extracting statistics across multiple window sizes.

The hyperparameters of the neural networks composing the ensemble are obtained by grid search. The grid search process is automated with Keras-Tuner [27] using HyperBand [28], a bandit-based algorithm, to find the best hyperparameters at each iteration. We run a very flexible grid search, making as few assumptions as possible and allowing the neural network to find its most appropriate shape given the underlying task. For each main architecture tested, we define only its essential building blocks (an input layer, a hidden layer, a dropout layer to prevent overfitting [21], [29], and the output layer) and coded everything else as hyperparameters: the number of units per layer, the number of additional hidden layers, dropout rate at each additional layer, activation functions, and the learning rate. To prevent the network size from spiraling out of control, the maximum number of additional layers is capped at 9, and units per hidden layer are kept under 192 and 96 for DNN and GRU layers, respectively. The dropout rate is set between 0% and 50%. The learning rate is bounded between 5e-5 and 1e-3. The optimizer function used for each experiment is RMSProp [30]. For the HyperBand search algorithm, the reduction factor for the number of epochs is set to 3 and the number of times to iterate over the full HyperBand algorithm is set to 20. Inside each model training iteration performed by Keras-Tuner, we set the batch size to 64 and the maximum epochs to 80 with an early stopping callback set at 5 epochs. Temporal hold-out is used as the validation technique. To improve generalization performance and avoid overfitting the validation set given the complexity of the models involved, we select a validation period of one year prior to the test period. Standard area under ROC curve (AUC)-receiver operating characteristic (ROC) and mean square error (MSE) are used as objective metrics for the classification and regression tasks, respectively. Note that our flexible grid search strategy, where everything is considered a hyperparameter, cannot be run with a traditional grid search approach that tries each possible hyperparameter combination, since it would take too much time. Therefore, an efficient search algorithm like the bandit-based HyperBand algorithm is needed. For each architecture, the grid search takes between 4 and 12 hours using an NVIDIA QUADRO RTX 5000 graphics card with 16 GB of memory [31].

C. State-of-the-art PL-based method

The PL-based algorithm, to be used for the comparison with our ML-based model, is the one detailed in [26]. This PL-based model uses Kalman filters (KFs) to identify rain events, and then inverts the SNR attenuation with a PL-based algorithm. KFs are a class of filters that efficiently estimate the states of a linear dynamical system from a series of incomplete and noisy measurements. The KF leverages its prediction of the system state along with the new measurements to form a statistically optimal estimate, using Gaussian probabilities and covariance matrices (which provide the weights associated with measurement and process noise) to manage estimation error and uncertainty. The covariance matrices allow us to control the weights associated with measurement and process noise. Higher values in the process noise covariance matrix increase the uncertainty associated with the process model, and lead the filter to be closer to the measured values. Based on these properties, [26] defines a set of two KFs: one with a low process noise covariance matrix and the other with a high process noise covariance matrix. Thus, one filter closely follows the measured SNR time series, while the other is more stable and acts similarly to a long rolling average. Once a statistical difference between the filter values is detected, the PL-based rainfall inversion is started. When it rains, the SNR attenuation is represented by the specific logarithmic attenuation k (in dB/km), which is empirically related to the rain rate R (in mm/h) by a PL, as shown in [6].

III. Conditional ensemble for rainfall estimation

A. Meta models

The meta models for the final conditional ensemble consist of a classifier (which is itself an ensemble of two neural networks, a GRU neural network and a multi-head neural network), and a tree-based regressor.

The model composing the classifier ensemble has the following characteristics: a GRU with $130,000$ parameters distributed over 6 layers; and a multi-head neural network (MHNN) with $107,000$ parameters. This type of model combines the characteristics of the previous neural networks. This model is, in practice, a multi-input model where one part of the model (one "head") takes 60 minutes of the raw SLNB signal as the input, and processes the information with a number of GRU layers; the other "head" takes the 35 available statistics as the input, and processes the information through a series of dense layers. This information is then combined and refined with another series of dense layers, until a final layer outputs the result. We obtain our final ensemble classifier by averaging the outputs of both classifiers with a weighted average, where the weights are learned on the training set. The regressor model is a GBM that takes the extracted statistics of each sample, which include 30 features, as input.

B. Conditional ensemble model

The final model is a conditional ensemble combining the classifier and regressor to obtain a single, improved regression value. This setup involves two main components: for any instance i, *i*) a classifier $y_{\mathsf{c},i}$ that predicts whether the output of the regressor $y_{\mathsf{r},i}$ should be considered or set to zero; and *ii*) a regressor $y_{\mathsf{r},i}$ that predicts a continuous outcome, but its output is used conditionally based on the classifier's output.

		training		*validation*	
model	*# of weights*	*AUC*	*F1*	*AUC*	*F1*
GRU	130,000	0.872	0.730	0.839	0.681
multihead	107,000	0.897	0.768	0.841	0.792
ensemble	237,000	0.899	0.770	0.858	0.798

Table I: Classifiers' individual and ensemble performances (F1 is reported as the harmonic mean between precision and recall at the best identified decision threshold).

	KF	*ML*
accuracy	0.772	0.799
F1-score	0.764	0.792
recall	0.727	0.760
precision	0.804	0.828
FPR	0.182	0.163

Table II: Model assessment on test data of KF and ensemble GRU-MHNN classifier (decision threshold of ensemble model optimized on validation set: 55%; test data imbalance: 50%).

The parametrized piecewise function is

$$f_i(x_1) = \begin{cases} 0, & \text{if } y_{c,i} < x_1, \\ y_{r,i}, & \text{otherwise,} \end{cases} \tag{5}$$

with x_1 being the decision threshold parameter (see below for practical values). In this architecture, the classifier's output is part of a decision rule in the model's inference stage, where the output of the entire model is either the regressor's prediction or zero, based on the classifier's decision.

IV. Experimental results

A. Detection analysis

We first investigate the detection capabilities of various models on the task at hand, using multiple ML algorithms. For model selection, we primarily use AUC, a metric that describes a classifier's ability to discriminate between two classes, and F1 score (hereafter referred to as F1), which corresponds to the harmonic mean of precision and recall, reported here at a 50% decision threshold. The ensemble of classifiers is created by simple averaging. The best-performing models on the validation set, specifically the GRU and a MHNN model with 130,000 and 107,000 parameters respectively, are described in Sect. III. In Table I, we report the performance results in terms of AUC and F1-score for both the training and validation sets of the aforementioned models, as well as the performance achieved by their ensemble. We observe that ensembling provides a slight improvement in the validation performance for both AUC and F1.

We then test the ensemble classifier and the Kalman classifier (described in Sect. II-C) on the same test data. The test arrays consist of rain events from the dataset referenced in Sect. II-B, with an additional 10-minute lag added before and after each event to achieve a 50% class balance. The comparative performance of the models is displayed in Table II, which contains recall, precision, and false positive rate (FPR). The ML (ensemble) classifier shows a mild advantage, exhibiting slightly higher accuracy and F1-score by about 3.50%.

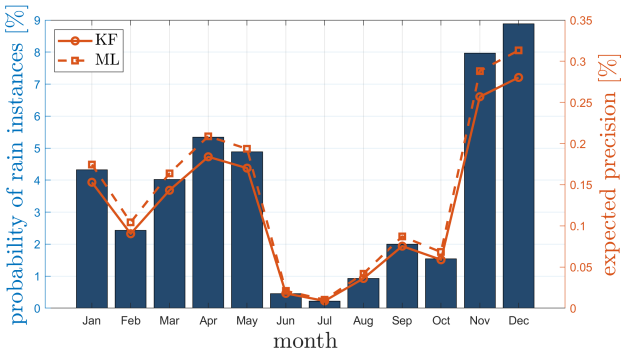


Fig. 1: Expected precision per month for KF and ML models (left: monthly probability of observing a positive instance in the population; right: expected precision of the classifiers).

The true class imbalance over which the models would be used in production settings is considerably more skewed (around 95% class imbalance) than our test set. We thus analyze the performance of both detectors (KF and ML ensemble) at varying levels of probability of rain, so as to estimate the performance of the models in a production imbalance. Precision is a class-balance sensitive metric, since it is computed by conditioning on predicted positives. This conditioning negatively affects precision in case of high imbalance. Indeed, applying the Bayes theorem to the precision metric, we obtain

$$P(\mathsf{AP}|\mathsf{PP}) = \frac{P(\mathsf{PP}|\mathsf{AP})\,|P(\mathsf{AP})}{P(\mathsf{PP})}, \tag{6}$$

where AP and PP represent the *actual positive* and the *predicted positive* events, respectively. Otherwise stated, the lower the probability of observing a positive instance, the lower the value of $P(\mathsf{AP})$, which results in a lower precision value $P(\mathsf{AP}|\mathsf{PP})$.

Using historical data, we calculate the expected month-by-month probability of observing a positive rain event. By plugging these probabilities into (6), we can obtain the results shown in Fig. 1, which displays the monthly probabilities along with the expected precision of both models over the course of the year. The probability of rainfall ranges from as low as 0.02% in July, to nearly 9% in December. Given this significant class imbalance, the models generally exhibit low precision. Again, this precision reduction in highly imbalanced conditions is somewhat expected. In Fig. 1, we can see that the ML model consistently outperforms the KF model, except for June, July, and August, where the performance of both models converges due to extreme class imbalance.

B. Regression analysis

Using the same training set as for the classification component, we develop an ML regressor. After multiple trials with various architectures, a tree-based GBM model has emerged as the best-performing one. This GBM has then been conditionally ensembled with the ML classifier to build the CENN.

	KF	*GBM*	*CENN*
MAE	0.025	0.020	0.018
NMAE	0.993	0.823	0.767
NMAE-dry	0.715	0.524	0.298
NMAE-rain	2.236	1.905	1.920

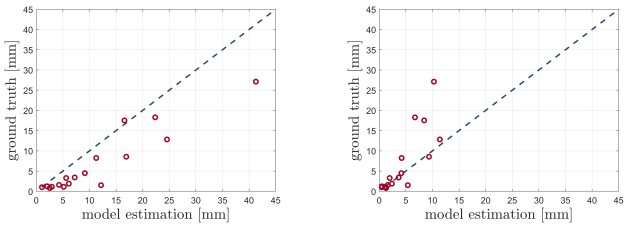
Table III: Model assessment on test data for both KF and ML models with a granularity of minute-by-minute measurements.

	KF	*GBM*	*CENN*
MAE	4.656	3.350	3.049
NMAE	0.694	0.500	0.454
correlation	0.934	0.752	0.819
RMSE	6.130	5.270	5.309
NMB	0.677	−0.136	−0.350
MAPE (%)	147.8	71.2	44.4

Table IV: Model assessment on event-based cumulative precipitation of test data for both models.

We analyze the regression performance of the KF PL regressor model compared to our ML model, also providing the performance against both our single GBM regressor and the CENN. The minute-by-minute regression metrics on the test set are reported in Table III. Over most metrics, the GBM shows a slight advantage over the KF model. This gap is further increased by the CENN, which achieves a 28% reduction in overall mean absolute error (MAE) and a more than 58% decrease in normalized mean absolute error (NMAE) on dry instances. Given that the CENN is forced to 0 when predictions by the classifier ensemble are too low, this approach effectively leverages classifier predictions, notably improving the accuracy in avoiding false rain predictions compared to both KF and GBM models. Note that the KF model slightly outperforms the ML models with high-intensity rains.

We further analyze the cumulative performance of both classifiers on test events. Figs. 2(a) and 2(b) display scatterplots of measured versus predicted cumulative precipitation for each test event, for the KF model and CENN, respectively. Regression performance metrics calculated on these cumulative values are collected in Table IV, which also reports root mean squared error (RMSE) and normalized mean bias (NMB). To highlight the benefits of using the CENN over the GBM regressor alone, we provide performance comparisons for both models. Similar to the minute-by-minute performance, the CENN shows a 34% reduction in overall MAE compared to the KF and and 9% reduction compared to the GBM model. The maximum correlation value for the KF model cumulatives is 0.93, indicating a strong linear relationship between its predictions and the targets, as can be visually confirmed in Fig. 2(a). However, these predictions are strongly positively biased, meaning that the model systematically overestimates the total precipitation of the events. This is reflected in the highest overall MAE and NMAE, and more tangibly, in the very high mean absolute percentage error (MAPE) of 147% on average for all events. On average, for a given event, the KF model has an absolute error of around 150% of the value of that event. In contrast, the CENN, while having a significantly



(a) KF model. (b) CENN.

Fig. 2: Scatterplot of measured versus predicted event-based cumulative precipitation.

lower correlation of 0.82, exhibits less bias and a drastically lower MAPE of 44% on average for all events.

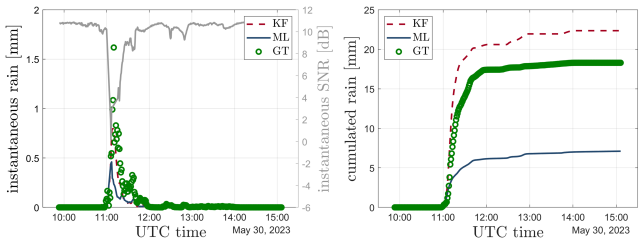
Finally, in Fig. 3, we provide a visualization of two selected rain events from the test set, where the ground truth (GT) is reported with green dots: one favoring the KF model (dash-dotted red lines), and another favoring the CENN (solid orange lines). The event from May 30th, 2023, has a total measured precipitation of $18.3\,\mathrm{mm}$, where the KF model manages to approximate this substantial rain event significantly better. In contrast, during the more mild event on June 2nd, 2023, which has a total measured precipitation of $4.5\,\mathrm{mm}$, the CENN achieves near-perfect accuracy in estimating the precipitation both minute-by-minute and cumulatively. In this instance, the KF model significantly overestimates the total amount.

V. Discussion and Conclusion

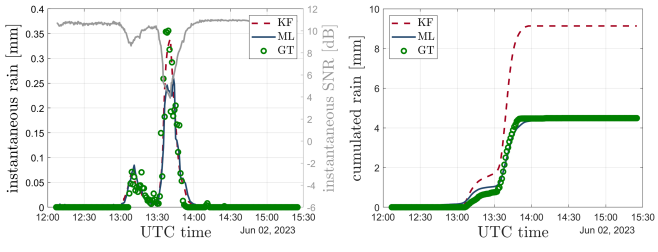
As illustrated by the two rain events in Fig. 3 and the scatter plots in Fig. 2, both the KF model and the CENN demonstrate merits in providing reasonable estimates. The underlying KF trackers ensure that the model identifies all events, and rarely underestimates cumulative precipitation. However, this also makes it more susceptible to falsely identifying rain instances, leading to a general overestimation of cumulative precipitation.

On the other hand, the CENN, which leverages both a regressor and a classifier, appears to be more capable of accurately estimating both light and heavy rain events. This is in general true for rain detection. Furthermore, it performs particularly well with small-to-medium events (less than $10\,\mathrm{mm}$), but it tends to underestimate high/extreme events, which are of most interest to researchers and stakeholders. From Fig. 2(b), we see that most events are close to the bisector, indicating perfect prediction; however, few outlier events are noticeably underestimated by the CENN. From a graphical analysis of these outlier events (one of which is reported in Fig. 3), we see that this happens when the SNR signal abruptly collapses. Since in the training set such SNR collapses are not always related to rain attenuation (e.g., during sun fading), the ML model is uncertain in identifying such instances as rain, which leads to the underestimation at inference time.

Supported by these results, we believe that ML models–particularly our conditional ensemble implementation–prove to be a valid alternative to well-established traditional algorithms for rain inversion from satellite SNRs. As a fu-



(a) intense rain (total: 18.3 mm), May 30th, 2023.



(b) light rain (total: 4.5 mm), June 2nd, 2023.

Fig. 3: Model predictions (left: minute-by-minute inference; right: cumulative predictions).

ture research direction, we will explore potential signal preprocessing methods to reduce the noise at the heart of ML model underestimation errors. We will also investigate possible combinations of KFs with ML models. Finally, we aim to extend this analysis to other locations, and train ML models to achieve a wider geographic generalization.

Acknowledgements

This paper is supported by the project SCORE (Smart Control of the Climate Resilience in European Coastal Cities), funded by EU Horizon 2020 research and innovation program under G.A. 101003534, by COST Action CA20136 OPENSENSE (Opportunistic Precipitation Sensing Network), by the project FoReLab (Departments of Excellence), funded by the Italian Min. of Education and Research (MUR), and by the EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - RESTART, structural project 6GWINET, cascade call SPARKS).

References

[1] U. N. Sustainable Development Group, "Foundational primer on the 2030 agenda for sustainable development," U. N., Tech. Rep., 2019.

[2] S. Lindersson, L. Brandimarte, J. Mård, and G. di Baldassarre, "A review of freely accessible global datasets for the study of floods, droughts and their interactions with human societies," *WIREs Water*, vol. 7, 2020.

[3] J. M. Buttle, D. M. Allen, D. Caissie, B. Davison, M. Hayashi, D. L. Peters, J. W. Pomeroy, S. Simonovic, A. St-Hilaire, and P. H. Whitfield, "Flood processes in canada: Regional and special aspects," *Can. Water Resour. J.*, vol. 41, pp. 7–30, 2016.

[4] D. R. Gergel, B. Nijssen, J. T. Abatzoglou, D. P. Lettenmaier, and M. R. Stumbaugh, "Effects of climate change on snowpack and fire potential in the western USA," *Climatic Change*, vol. 141, pp. 287–299, 2017. [Online]. Available: https://doi.org/10.1007/s10584-017-1899-y

[5] C. Chwala and H. Kunstmann, "Commercial microwave link networks for rainfall observation: Assessment of the current status and future challenges," *WIREs Water*, vol. 6, p. e1337, 2019.

[6] R. Olsen, D. Rogers, and D. Hodge, "The aRb relation in the calculation of rain attenuation," *IEEE Trans. Antennas Propag.*, vol. 26, no. 2, pp. 318–329, 1978.

[7] M. Fencl, J. Rieckermann, P. Sýkora, D. Stránský, and V. Bareš, "Commercial microwave links instead of rain gauges: Fiction or reality?" *Water Sci Technol*, vol. 71, no. 1, pp. 31–37, 2015.

[8] W. Wolff, A. Overeem, H. Leijnse, and R. Uijlenhoet, "Rainfall retrieval algorithm for commercial microwave links: Stochastic calibration," *Atmos. Meas. Tech.*, vol. 15, pp. 485–502, 2022.

[9] B. Lian, Z. Wei, X. Sun, Z. Li, and J. Zhao, "A review on rainfall measurement based on commercial microwave links in wireless cellular networks," *Sensors*, vol. 22, no. 12, p. 4395, 2022. [Online]. Available: https://doi.org/10.3390/s22124395

[10] H. V. Habi and H. Messer, "Recurrent neural network for rain estimation using commercial microwave links," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 3672–3681, 2021.

[11] J. Pudashine, "Deep learning for an improved prediction of rainfall retrievals from commercial microwave links," *Water Resour. Res.*, vol. 56, no. 7, pp. 0–3, 2020.

[12] F. Giannetti and R. Reggiannini, "Opportunistic rain rate estimation from measurements of satellite downlink attenuation: A survey," *Sensors*, vol. 21, p. 5872, 2021.

[13] M. Delamou, A. Bazzi, M. Chafii, and E. M. Amhoud, "Deep learning-based estimation for multitarget radar detection," in *Proc. IEEE Veh. Technol. Conf.*, Aug. 2023.

[14] W. Njima, A. Bazzi, and M. Chafii, "DNN-based indoor localization under limited dataset using GANs and semi-supervised learning," *IEEE Access*, vol. 10, pp. 69 896–69 909, 2022.

[15] A. Gharanjik, B. Mysore, F. Zimmer, and B. Ottersten, "Centralized rainfall estimation using carrier to noise of satellite communication links," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 5, pp. 1–1, May 2018.

[16] M. Xian, X. Liu, and T. Gao, "An improvement to precipitation inversion model using oblique earth–space link based on the melting layer attenuation," *IEEE Trans. Geosci. Remote Sens.*, pp. 6451–6465, 2020.

[17] C. Gianoglio, A. Alyosef, M. Colli, S. Zani, and D. Caviglia, "Rain discrimination with machine learning classifiers for opportunistic rain detection system using satellite micro-wave links," *Sensors*, vol. 23, no. 3, p. 1202, 2023.

[18] B. He, X. Liu, S. Hu, K. Song, and T. Gao, "Use of the C-band microwave link to distinguish between rainy and dry periods," *Adv. Meteorol.*, vol. 2019, p. 3428786, 2019.

[19] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, 1990.

[20] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[21] F. Chollet, *Deep learning with python*. New York, NY: Manning Publications, 2017.

[22] R. M. Schmidt, "Recurrent neural networks (RNNs): A gentle introduction and overview," 2019.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS Workshop on Deep Learning*, Dec. 2014.

[25] NEFOCAST project. (accessed Oct. 4th, 2024). [Online]. Available: https://www.nefocast.it/

[26] F. Giannetti, M. Moretti, R. Reggiannini, and A. Vaccaro, "The NEFOCAST system for detection and estimation of rainfall fields by the opportunistic use of broadcast satellite signals," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, pp. 16–27, 2019.

[27] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

[28] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Machine Learning Research*, pp. 1–52, 2018.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Machine Learning Research*, pp. 1929–1958, 2014.

[30] G. Hinton. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[31] NVIDIA website. (accessed Aug. 2nd, 2024). [Online]. Available: https://www.nvidia.com/