A Deep Neural Network-based Communication Failure Prediction Scheme in 5G RAN

Mohammad Ariful Islam*, Hisham Siddique*, Wenbin Zhang †, and Israat Haque*
*Department of Computer Science, Dalhousie University, Halifax, Canada – {m.arifulislam;hisham.siddique;israat}@dal.ca †Carnegie Mellon University – wenbinzhang@cmu.edu

***Abstract*—5G networks enable emerging latency and bandwidth critical applications like industrial IoT, AR/VR, or autonomous vehicles, in addition to supporting traditional voice and data communications. In 5G infrastructure, Radio Access Networks (RANs) consist of radio base stations that communicate over wireless radio links. The communication, however, is prone to environmental changes like the weather and can suffer from radio link failure and interrupt ongoing services. The impact is severe in the above-mentioned applications. One way to mitigate such service interruption is to proactively predict failures and reconfigure the resource allocation accordingly. Existing works like the supervised ensemble learning-based model do not consider the spatial-temporal correlation between radio communication and weather changes. This paper proposes a communication link failure prediction scheme based on the LSTM-autoencoder that considers the spatial-temporal correlation between radio communication and weather forecast. We implement and evaluate the proposed scheme over a huge volume of real radio and weather data. The results confirm that the proposed scheme significantly outperforms the existing solutions.**

***Index Terms*—5G, RAN, failure prediction, LSTM, Autoencoder.**

I. Introduction

Emerging networking applications like industry 4.0, intelligent transportation, smart health system, AR/VR, etc., demand high network bandwidth, high reliability, and low communication time [1]. Mobile and wireless devices from these applications usually communicate over radio links and form various types of networks like mesh, sensor, or cellular [2]. Specifically, fifth-generation (5G) cellular networks aim to support the above emerging applications with different service level objectives (SLOs). For example, Mobile Broadband (eMBB), ultra-Reliable Low Latency (uRLLC), and massive Machine Type Communication (mMTC) offer enhanced bandwidth (e.g., AR/VR), low latency (e.g., autonomous vehicles), and low-power massive machine-to-machine communication (e.g., industry 4.0), respectively. Unlike 4G networks with large and high-power cell towers to reflect signals over long distances, a 5G network consists of cells with a small coverage. Specifically, 5G uses millimeter-wave (mmWave) spectrums (between 24GHz and 100GHz) [3] that can travel over short distances; thus, 5G RAN needs to deploy a dense collection of base stations compared to 4G.

However, the use of mmWave frequency is affected by the weather conditions. The mmWave radio frequency suffers from frequency-dependent absorption, refraction, and scatter while passing through the atmosphere and free-space, which causes distortions in the amplitude and phase of signals [3], [4]. Thus, the parameters of signals from the radio antenna of a 5G base station can attenuate to cause the unavailability of the channel transmission. Both water and wind can affect mmWave, e.g., a strong wind can create atmospheric vibration and attenuate key performance indicator (KPI) parameters [5].

Thus, a 5G-based system for emerging IoT applications must carefully incorporate such weather impacts on mmWave to be robust against weather fluctuations. We can think of different design approaches like using learning based algorithms [6], [7] or Reconfigurable Intelligent Surfaces (RIS) [8]–[10]. The former application layer solution can complement the RIS-based scheme deployed in the lower layer of the protocol stack (see further discussion in Section V-D). However, we focus on designing a learning-based scheme in this paper as such a scheme is proven to be efficient in designing automated network anomaly detection (e.g., intrusion [11] or failure [12]) scheme. In the case of failure prediction, it can help service providers learn about a possible failure and then take the necessary measures. For example, service providers can predict radio link failure in the next 24 hours to support uninterrupted latency-critical IoT applications.

Some works considered radio link failure prediction in the presence of weather fluctuations using learning-based algorithms [6], [7]. The authors of [6] used 5G radio performance indicator data and weather forecast data from a Turkish company ITU-Turkcell [13]. Then, they proposed an ensemble learning-based model composed of Random Forest (RF), Light Gradient Boosting Machine (LGBM), and Extreme Gradient Boosting Machine (EGBM). Aktas *et al.* argue that such failure prediction must consider the temporal correlation in data; thus, they proposed a Long Short Term Memory Network (LSTM)-based model [7]. They separated time-dependent features from configuration ones and used the LSTM and dense neural networks, respectively.

We first analyzed both the radio and weather data sets to understand the feature correlations in this work. We discovered that the features not only have temporal correlation but also have spatial one. Thus, we must incorporate the data in a model in such a way that it can capture that correlation, which also aligns with the claim from [7]. Moreover, we realized that the prediction scheme should incorporate weather forecast data. The rationale for such a choice is that link failure would be correlated to the weather condition that we predict rather than the current one. Specifically, we can assume that radio link KPIs provide the context for failure susceptibility, and

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3229658

the weather forecast offers the anticipated conditions the link will experience. Our rationale also aligns with the state-of-the-art solution [6] and further gets stronger following the performance degradation while using the current weather-based failure prediction in [7]. Thus, we choose weather forecast data to build the proposed model (see Fig. 1).

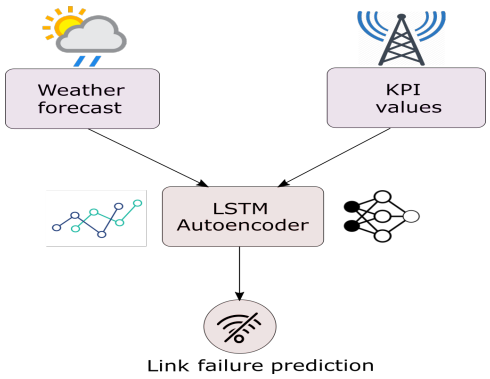

Fig. 1: Forecast weather-based link failure prediction.

This paper proposes an LSTM-autoencoder-based radio link failure prediction scheme considering the spatial-temporal data correlation and weather forecast data. Unlike existing works, we rigorously analyze and pre-process data appropriately for efficient and accurate training, validation, and testing. We also deploy time-series cross-validation due to the time-dependent nature of the data, which is missing in existing works. Finally, we test our model over a diverse set of test data spreading over 2019 and 2020. Note that existing works only trained and tested their solutions over the data from 2019. On the contrary, we trained the proposed model using a portion of data from 2019; then tested it using the remaining 2019 data along with the data from 2020.

The proposed scheme first combines the radio and weather forecast data in the pre-processing phase considering their correlation in the Euclidean space. Then, we apply data encoding to convert the categorical features to binary vectors, followed by PCA-based dimensionality reduction. Next, we perform data balancing as the number of failed links is significantly smaller than the normal operating ones. We train the proposed LSTM-autoencoder model over fail-free links and validate that model with a combination of both types of links, where the model generates higher reconstruction errors in the case of failures. Finally, we test the proposed model on a large set of data to show its effectiveness compared to the existing solutions [6], [7].

We evaluate the performance of the proposed model using the real radio and weather data from [14] and compare it with the existing ensemble and LSTM+ models. We predict failures for the next day and measure the performance in terms of precision, recall, and F1 score. *Precision* refers to the ratio of true positive predictions to the total positive predictions, and *recall* is the ratio of the true positive predictions to the actual positive samples. An *F1 score* is a harmonic mean of precision and recall (see further details in Section V-A). The evaluation results reveal that the proposed LSTM-Autoencoder offers **14%** and **33%** higher precision compared to the ensemble and LSTM+ models, respectively. In summary, the proposed work has the following contributions.

- We rigorously analyze the data and propose a novel pre-processing scheme composed of data merging (radio and weather), encoding, dimensionality reduction, and balancing in a cohesive fashion.
- The model training and validation offer a new time-series cross-validation-based approach, where we learn the classification threshold during the validation.
- We test the model performance over both 2019 and 2020 data while training was conducted on the data from 2019.
- The evaluation results confirm the superiority of the proposed solution compared to existing ones. We also share the code with the community for reproducibility [15].

II. Background and Motivation

This section presents the necessary background on different ML models to understand the proposed LSTM-autoencoder-based prediction scheme. We also lay out the rationale for our design choices.

5G RAN. As the number of users connected to mobile networks has been increasing substantially in the last few years, we have been seeing more and more innovations in this field. One of the main changes is the advent of 5G networks. It uses mmWaves to allow for high bandwidth and low latency communication, making 5G a key enabler for emerging applications like augmented reality, autonomous vehicles, and smart cities. However, mmWaves increase the bandwidth at the cost of coverage size and higher penetration loss. To circumvent these problems, 5G RAN networks tend to have a high density of small cells coupled with larger macrocells [16]. With this deployment, users can connect to the smaller and closer cells, then communicate with the macrocells that send traffic to the core network, connecting the user to the Internet. An outline of this type of deployment is depicted in Fig. 2.

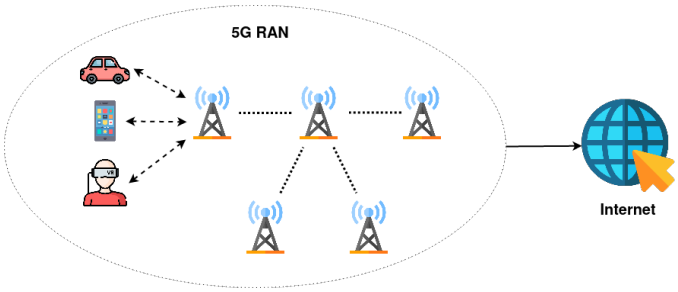

Fig. 2: Example of a 5G RAN.

Encoding and dimensionality reduction. A neural network model requires input and output variables as numeric values [17], i.e., we cannot use categorical data while using deep learning models. There are various encoding approaches, e.g., we can convert categorical data to binary vectors using one-hot encoding [18]. It best fits for categorical data without any ordinal properties [18], [19], which is the case in our data (e.g., rainy, foggy, sunny). Using categorical values as input,

we create new categorical columns and assign binary values to each, i.e., create binary vector representation. For example, instead of representing "weather" features {rainy,cloudy,sunny} as arbitrary numbers like 1, 2, and 3, we create a binary vector of size 3. The presence and absence of a value in this vector are denoted by 1 and 0, respectively. Thus, we can represent rainy, cloudy, and sunny as 100, 010, and 001, respectively.

However, a consequence of such categorical encoding is that it may add more dimensions to the data and impact the learning of models. One way to avoid such issues is to reduce the data dimensions, e.g., using Principal Component Analysis (PCA) [20]. PCA is an unsupervised statistical analysis method for component analysis that constructs relevant features using a linear combination of original features. Specifically, correlated features are linearly transformed into smaller uncorrelated components to construct the most relevant features [20].

For example, consider a dataset with three dimensions. In order to find the principal component of this dataset, we would first have to standardize the data by subtracting each dimension's mean. This standardized dataset is then used to find the covariance matrix, C, according to the equation, $C = \frac{\sum(X_i)(X_i^T)}{n}$, where X is the standardized dataset and n is the sample size. Using the equation, $C - \lambda i = 0$, we solve it for λ, where each value of λ corresponds to a dimension of the dataset. Thus, there would be three values of λ: λ_1=1.45, λ_2=0.675, λ_3=1.09 for three dimensions. The higher the value, the more relevant that dimension is, i.e., the first dimension is the most relevant in this example. These values of λ, called eigenvalues, can then be further used to generate their respective eigenvectors through which the entire dataset can be reconstructed [21].

Data balancing. An important aspect of data preprocessing is balancing minority and majority classes within the data. Our radio dataset has a significantly smaller number of failed events compared to the normal functional links. Usually, a minor imbalance between majority and minority classes (e.g., 60:40) does not impact the learning performance of a model, which is not the case for a significant imbalance (e.g., 90:10), i.e., the model cannot establish a correct decision boundary between classes. In this case, a classifier that always classifies an input as the majority class will have 90% accuracy [22]. To alleviate this imbalance, we can use undersampling [23] or oversampling [24]. Usually, oversampling tends to be preferred better over undersampling as it may remove important information from the data pertaining to classes [24].

We used the synthetic minority oversampling technique (SMOTE) [24], where the synthetic samples are generated for the minority class to avoid the overfitting problem caused by random sampling. By interpolating between existing minority instances, it generates new data points [24]. Specifically, we can identify a total of N minority instances and randomly choose one. Then, we can retrieve its K-Nearest Neighbours (KNN), say for $k = 5$. The approach utilizes a distance metric to calculate the difference between the chosen instance (feature vector) and its k neighbors, which is multiplied by a random value between $[0, 1]$ [25]. The process continues as long as we want to add minority instances.

Autoencoder. An autoencoder consists of an encoder $g(\phi)$, a latent space representation Z_n, and a decoder $f(\theta)$ (see Fig. 3). The encoder learns a compressed vector representation of the input data X_n, whereas the decoder reconstructs the input from the compressed vector Z_n [26]. The goal is to learn the input data structure through the compress-decompress mechanism while minimizing the reconstruction error. Specifically, the error is backpropagated through the model architecture to update network weights in hidden layers. The process continues until the model converges with the minimum error [26]. Thus, the reconstructed output $X_n^{'}$ is similar to the input.

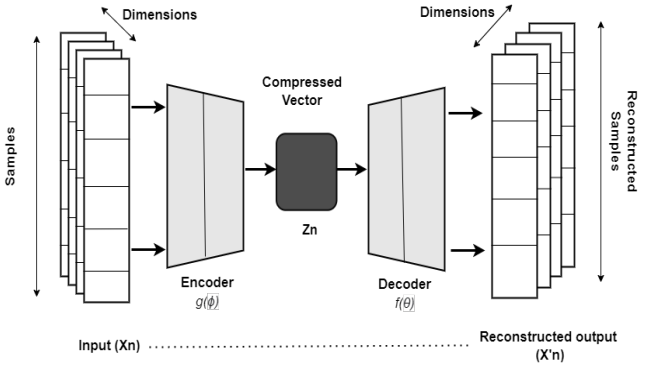

Fig. 3: Autoencoder architecture.

The autoencoder is a great candidate for anomaly detection as it can learn the useful hidden features present in the training data with minimum error while generating a higher error in the case of anomaly [26]–[29]. Thus, in the failure prediction problem, we carry out the link failure prediction by identifying the instances which cannot be correctly reconstructed and classifying them as failed links. We use Mean Absolute Error (MAE) as the loss function to calculate the difference between the original and reconstructed vectors.

LSTM-autoencoder. An LSTM is a Recurrent Neural Network (RNN) that allows a network to retain long-term time dependencies between data points. The LSTM architecture consists of a series of repeating modules containing three control gates: forget, input, and output gates. The former controls which information to pass from one module to the next to maintain time-dependent context [30]. The proposed failure prediction model needs to deal with the correlation of radio and weather time series data and predict possible failure in the next 24 hours, where we can use LSTM. In the following paragraphs, we will present an LSTM-Autoencoder model that combines characteristics from LSTM and autoencoder; thus, a suitable failure prediction model in our problem space. Specifically, the operation of an autoencoder with an LSTM involves reconstructing the input data while maintaining the temporal dependency of the features.

Fig. 4 presents the architecture of an LSTM-Autoencoder. The LSTM-encoder transforms a given input sequence into a fixed-length vector or compressed sequence in a latent space. This fixed-length vector is called *context vector*. The LSTM-decoder takes the context vector as input to predict output sequences. A repeat vector layer is used at the entry of the

decoder to replicate the context vector n times, where n is the number of future steps to generate. Each of these replicas of the context vector is then fed to the LSTM-decoder to generate n outputs one for each time step. The outputs of the decoder are then passed through a *Time Distributed* Layer, where a fully connected dense layer is applied to each of the outputs to generate the final one [31].

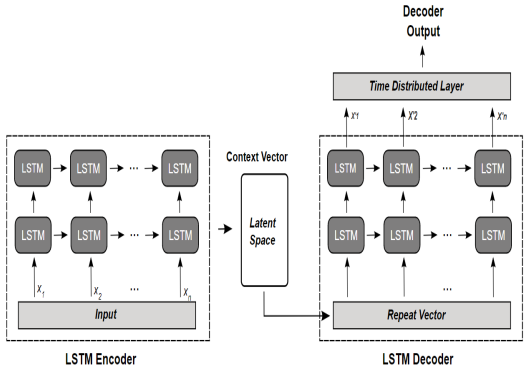

Fig. 4: LSTM-Autoencoder architecture.

III. Dataset Description

The telecommunication service provider Turkcell from Turkey gathered the data we used in this work from regional co-located weather stations and radio towers (sites). The data was collected from radio towers and surrounding weather stations between 2019 and mid-2020 [15]. Due to security and privacy issues, they also offered a pairwise distance matrix between these weather and radio towers without providing the actual geo-locations. Furthermore, some of the KPIs and configuration data is also anonymized. A brief description of the data is given below. The detailed description of various features can be found in [15].

Radio sites. This dataset contains spatial features of radio sites, including the site-no, the ground height (height from the sea level), and the clutter class (environment type at the site location, e.g., dense tree area, industrial, or commercial area). There are 1674 radio sites in this dataset.

Key performance indicators (KPIs). Each sample in the KPI data represents features used to measure the performance of radio links. There are 21 KPIs features; out of these, date-time, polarization, frequency band, link length, severely error seconds, error seconds, unavailable seconds, available time, background bit error (BBE), and modulation are the most effective features to determine the condition of radio links. Some of them contain the sequential properties, i.e., *seasonality*. For example, date-time, severely error seconds, error seconds, unavailable seconds, available time, and BBE are such features. On the other hand, features like polarization, link length, modulation, clutter class, and ground heights represent the spatial properties of the data. Samples for each radio link are measured every day as aggregated values. A summary of the radio data is given in Table I.

Weather station data. Similar to the radio sites' spatial features, this dataset contains spatial features of weather stations, which include station ids, ground heights, and clutter classes.

Subset	Total Features	Total Samples
RL-Sites	3	1674
RL-KPI	21	979063

TABLE I: The summary of radio data.

Real weather data. It has samples from 117 weather stations. The most impactful features out of 16 are date-time, measure-date, measure-hour, temperature (max, min, and current), wind direction (max, min), wind-speed (max, min), humidity, precipitation, precipitation-coefficient, pressure, and pressure-sea-level. Several features outlined above contain sequential properties of the weather data, whereas clutter classes and ground heights represent the underlying spatial properties of the weather stations.

Forecast weather data. The weather forecast data contains forecast reports for the five upcoming days. The useful features include date-time, report-time, temperature, humidity, wind directions, and wind speed, which exhibit sequential properties of the data. A summary of the weather data is given in Table II.

Subset	Total features	Total Samples
Met-Forecast	9	39370
Met-Real	16	629217
Met-Stations	3	117

TABLE II: The summary of weather data.

Distance matrix. The exact location of the radio and weather stations is kept anonymous due to security and privacy reasons. However, their geo-relation is represented as pairwise distances, which reflect correlations among radio and weather stations. We summarize all the above dataset description in Table III.

Data Table	Contents
Radio sites	Spatial properties and environment of RL sites.
Radio link KPIs	RL performance indicators, e.g., BBE and unavailable seconds.
Weather station	Spatial properties and environment type of weather stations.
Real weather	Real time weather data, e.g., wind speed and temperature.
Weather forecast	Next five days weather forecast data.
Distance	Pairwise distances among weather and radio stations.

TABLE III: Dataset description.

Impact of weather on radio data. We performed an initial test to demonstrate the weather impact on radio data over time. Multiple weather features can impact the failure, but for simplicity and demonstration, we chose precipitation that is claimed to have the most impact [32]. The results in Fig. 5 show that heavy precipitation affects the mmWave transmission, resulting in link failures. The precipitation rate is highest during three months in the middle of the year.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3229658

Consequently, we observe frequent link failures during that time.

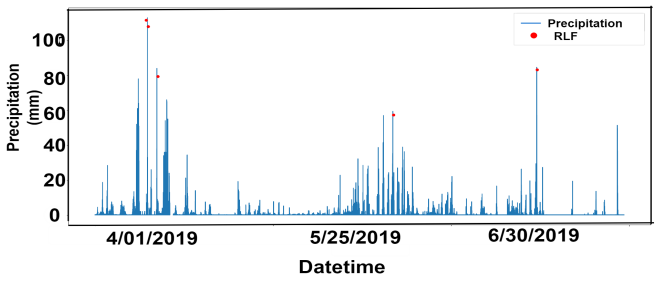

Fig. 5: Impact of weather (e.g., precipitation) on radio link failures.

IV. Design and Methodology

This section presents the proposed LSTM-autoencoder based failure prediction scheme composed of three phases: data preprocessing, model training and validation, and model testing. The workflow of the prediction scheme is depicted in Fig. 6, and each phase is detailed in the subsections below. At a high-level, the data *preprocessing* consists of correlating radio and weather data, encoding categorical features, reducing dimensionality, balancing samples from minority and majority classes, and splitting time-series data. Then, the next phase *trains* and validates the proposed LSTM-autoencoder and two existing models. Finally, we *test* the performance of all three models.

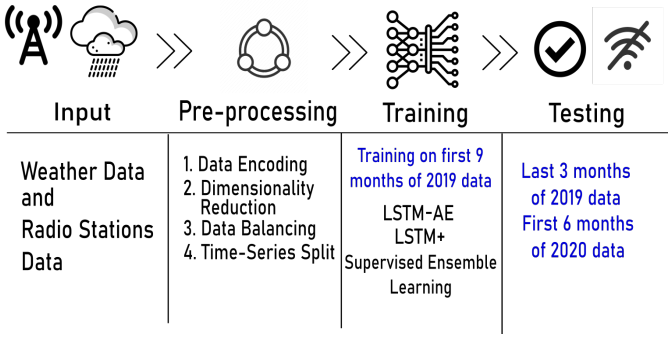

Fig. 6: Prediction workflow.

A. Data Preprocessing

Distance correlation and noise elimination. The primary focus of this work is to predict radio link failure due to the surrounding weather conditions. Thus, understanding the radio and weather station correlation is critical in this failure prediction task. For instance, our initial investigation revealed that in the provided distance matrix, there are some pairwise distances that are significantly larger than the remaining ones and can easily be detected as outliers (see Fig. 7). These distances add noise in the prediction process instead of adding any useful information. Thus, we propose a simple heuristic to choose the right set of weather stations associated with the radio stations, which is missing from previous works [6].

We first compute the distance between each radio tower and its closest weather station. Suppose there are r radio towers, so

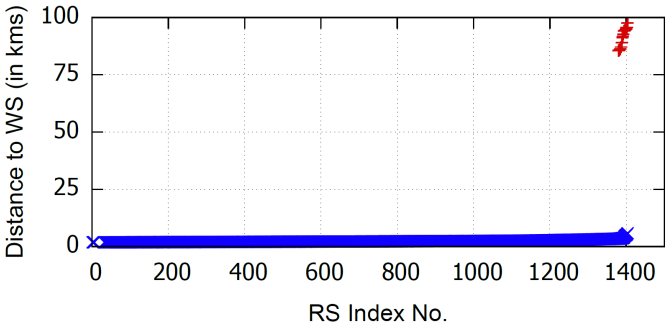

Fig. 7: Distances from radio sites (RS) to weather stations (WS).

we get a distance set: $\{d_1, ..., d_r\}$. Thus, we ensure that radio tower station is associated with at least one weather station. Next, we take the maximum from this distance set as the radio and weather station association threshold, which is 4.2 kms.

We merge corresponding samples across the same periods once we find the correct correlation between radio and weather stations. For instance, KPI samples are taken once every morning, so we merge these samples with the weather forecast at the same time. Next, we resolve the missing samples using the standard mean, median, mode, or elimination approach [33]. For example, we eliminate a sample with a missing radio link ID or frequency band as replacing it with a random numeric value will create a bias in the distribution. However, we use the mean value to replace the missing values of min-temperature, max-temperature, humidity-max, humidity-min, wind-speed, and link-length. We also utilized the median for replacing the missing cell for a scalability feature.

Data encoding. Our dataset has categorical features (e.g., rain, snow, etc.). As we mentioned earlier, we cannot use such categorical data while using deep learning models. The data has to be converted to numerical ones using binary or one-hot encoding methods. However, binary encoding can introduce artificial ordering to the encoded data, where the original categorical ones have no ordinality. In a neural network model training, such unregulated noise may create model performance degradation and overfitting [34]. Thus, we applied one-hot encoding to eliminate the possibility of such noise, where it converts categorical features into binary vectors [35]. Specifically, we created a new feature according to the categorical features and mapped it with a binary variable such that 0 and 1 represented the absence and presence of a category for a sample, respectively.

Dimension reduction. The categorical encoding increases the dimensionality of the data, which can cause features to appear equally spaced among them. Thus, the model may not learn the distribution accurately [36]. One way to solve this issue is by reducing the data dimensionality, which also reduces the learning complexity and cost. This work considers PCA, a feature compression approach, to reduce the dimension while keeping the maximum variance. Note that PCA removes low-ranked components without losing any information or impacting the feature distribution learning [36].

We first performed the parameter tuning while using PCA, i.e., tested different variances, number of components, perfor-

PC	Coeff	F1 Score
1377	100%	0.71
1136	96%	0.85
728	**90%**	**0.88**
665	85%	0.82
413	75%	0.77
169	65%	0.61

TABLE IV: Parameter tuning during the dimensionality reduction.

mance, and computation cost. Table IV presents the comparison among Principal Components (PC), Retained Coefficient Variance (Coeff), and F1 Score. The results show that the 90% variance retention with 728 components offers the best performance with the chosen one-hot encoding. The original number of features before applying PCA was 1377.

Data balancing and time-series split. Only a fraction of the total samples we have resulted in link failures; e.g., the minority to majority class ratio is 1:320. Such imbalanced data can prevent a model from successfully learning the decision boundary. There are several methods to re-balance the data like random under-sampling, over-sampling, class weight redistribution, or synthetic over-sampling [37]. However, undersampling can remove important information from the data. The most naive solution for oversampling is simply duplicating samples from the minority class, which does not add any diversity to the data. Thus, we chose the synthesized minority oversampling technique (SMOTE) to balance the data. The SMOTE adds synthetic samples systematically between minor class members and their k-closest neighbors [38]. Thus, the minority class sample is increased while data distribution remains intact. We chose $k = 4$ as the number of nearest neighbors in our implementation. After the re-balancing operation, the final minority to majority ratio improved from 1:320 to 1:10.

A random sampling of k-fold cross-validation is a standard approach in non-time-series data to split it into train, validation, and test sets, which is not the case in time-series data to retain the temporal correlation. Therefore, we do a 5-fold cross-validation for a time series split. We consider $k-1$ folds as a training set and the remaining fold for validation at every cross-validation iteration. For example, in the first iteration, we trained the proposed candidate model on the data from January to April 2019 and validated it over May 2019. Then, the trained data spanned from January to May while we validated over June 2019 in the next iteration. Thus, we utilized the expanding window approach of data splitting to keep the sequential property of the data intact [39]. We continue the process until September 2019 and use the remaining three-month data for model testing. We also test the model outcome with the data from January to June 2020. The entire process is depicted in Fig. 8, which is similar to a real-world scenario, i.e., we predict failure based on past observations.

B. Model Training and Validation

In this section, we present the model training procedure for the proposed LSTM-autoencoder as well as existing ensemble and LSTM+.

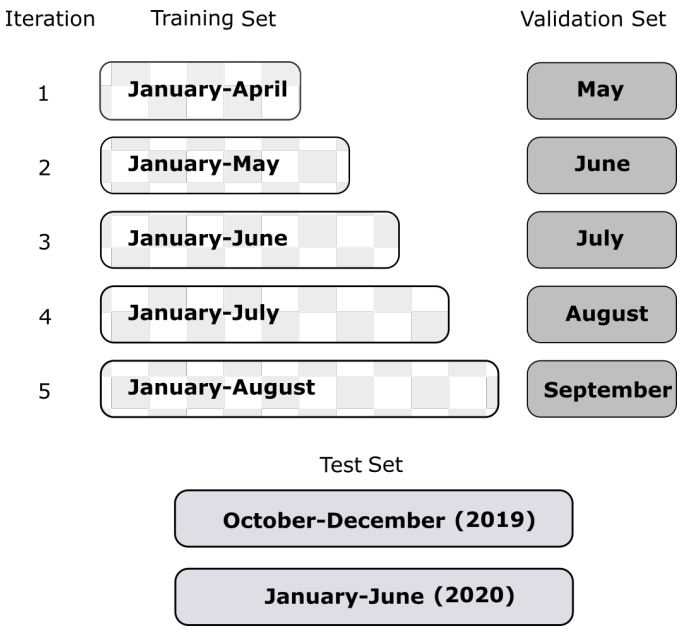

Fig. 8: Time series split representation.

LSTM-Autoencoder. The model utilizes an autoencoder's reconstruction property (learns the best parameters) and the sequence preserving property of an LSTM. An autoencoder is a great candidate to learn the structure of a given dataset and can detect any anomaly if that structure changes. Thus, we first trained our model with regular links without having any failed ones. Then, we provide that trained model with a mix of regular and failed links to check whether it generates high reconstruction errors for the failed ones. Precisely, each of the four layers of encoder and decoder consists of multiple LSTM units (see Fig. 9). In the following, we present details of each component of the proposed model.

The input data $X(n) \in \mathbb{R}$ was encoded over the encoder block to generate the feature vector $Z(n) \in \mathbb{R}$, where the dimensions of $X(n)$ and $Z(n)$ are 728 and 22, respectively. The $RepeatVector$ acts as a bridge between the encoder and the decoder. The encoder and the decoder each contain four hidden layers. A number of hidden layers were tested to determine the most optimal one. Similar to the encoder block, the layers in the decoder block are arranged in reverse order. The encoded features $Z(n)$ are sent over a series of LSTM blocks to generate the output feature vector $X^{'}(n)$. Each subsequent layer of the decoder increases the dimensions in the reverse order. The $TimeDistributed$ layer used the decoder's final output as input and created a 728×16 vector matched with the input one. Next, we describe how the proposed LSTM-AE is trained and validated before testing its performance.

We started our training with time-series data from January to April 2019, which does not contain any failed link samples. Then, we validated the model with a mix of regular and failed links from May 2019, where we considered training and validation sample sizes of 427080×728 and 106770×728, respectively. We selected *Mean Absolute Error (MAE)* for calculating the training loss as we are training the model for a binary prediction. The training loss converged to a solution after several iterations, indicating that the model could learn

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3229658

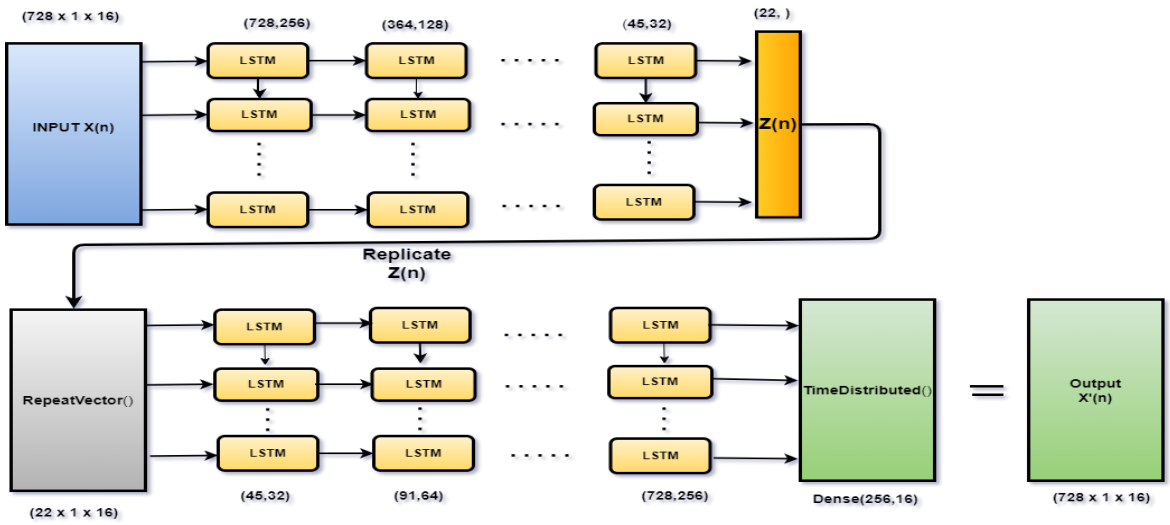

Fig. 9: The LSTM-autoencoder architecture.

the underlying structure. We chose the *Adam* optimizer to optimize the gradient descent for a better loss calculation. We used *grid-search* method to choose the appropriate learning rate, where the optimal values are: learning rate (0.001) and dropout (0.3).

Subsequently, the reconstruction loss during the validation was higher than the train one, i.e., the model struggled to reconstruct the failed events. We repeated the above procedure following time-series splitting described in Fig. 8. Specifically, we calculated a threshold based on the validation set reconstruction error and applied it to the test data for the failure prediction. The batch size and epochs are 16 and 50, respectively, chosen based on the optimal performance. The Hyper-parameters are listed in Table V.

Hyper-Parameter	Selected Value
Epochs	50
Learning rate	0.001
Hidden layers	4
Batch size	16
Optimizer	Adam
Dropout	0.3
Activation function	ReLU
Loss function	Mean Absolute Error (MAE)

TABLE V: Hyper-parameters of LSTM-AE.

LSTM+. Fig. 10 presents the LSTM+ architecture and operation. LSTM+ separates temporal features from spatial ones and feeds the former to LSTM units and the latter to dense fully connected layers [40]. Then, the model feeds the output vectors to a dense fully connected layer that concatenates the outputs and computes prediction scores for input samples. Specifically, the configuration variables (time-independent) are directly linked with a fully connected layer with 256 neurons, followed by another with 512 neurons. Both branches of the model generate multiple local compressed vectors, where a concatenation layer concatenates the local vectors. These vectors are then fed to two dense layers, each with 512 neurons (layer 3 and layer 4 in Fig. 10) and the final output layer with one neuron. The output layer applied the matrix dot product among parameters and optimal weights to calculate the probability distribution of the input samples.

We trained the model for both the forecast and real weather data, then tested it with the corresponding test data, where we split the data into train and test sets in the default setting. Also, we separated configuration features to support the dual branch input layer of LSTM+. The configuration features are frequency bands, card types, tips, adaptive modulations, and modulations and processed with the second input branch as shown in Fig. 10. We applied the same preprocessing steps as in LSTM-autoencoder: the one-hot encoding, PCA, and data balancing using SMOTE. Then, we applied Adam optimizer along with the binary cross-entropy error function during the model training, where the learning rate and dropout are 0.001 and 0.3, respectively. Following the LSTM+ design, we implemented *tanh* activation for all layers except the final dense output layer, where the activation function is *Sigmoid* to get the prediction score of the binary classification [41]. The hyper-parameters selected for this model are listed in Table VI.

Hyper-Parameter	Selected Value
Epochs	50
Learning rate	0.001
Dense layer dropout	0.03
Batch size	16
Optimizer	Adam
LSTM Layers	1
Dense Layer	4
Loss function	Binary cross-entropy.
Activation function	Tanh, Sigmoid.

TABLE VI: Hyper-parameters of LSTM+.

Ensemble learning. This supervised learning method combines multiple models to offer better accuracy and robustness [42]. Existing work [6] combined Random Forest (RF), Light Gradient Boosting Machine (LGBM), and Extreme Gradient Boosting Machine (XGBoost) in their failure prediction model (see Fig. 11). Here, a random forest predicts classes based on a concept called *wisdom of crowds*. It utilizes votes from individual trees and determines the output class based on the

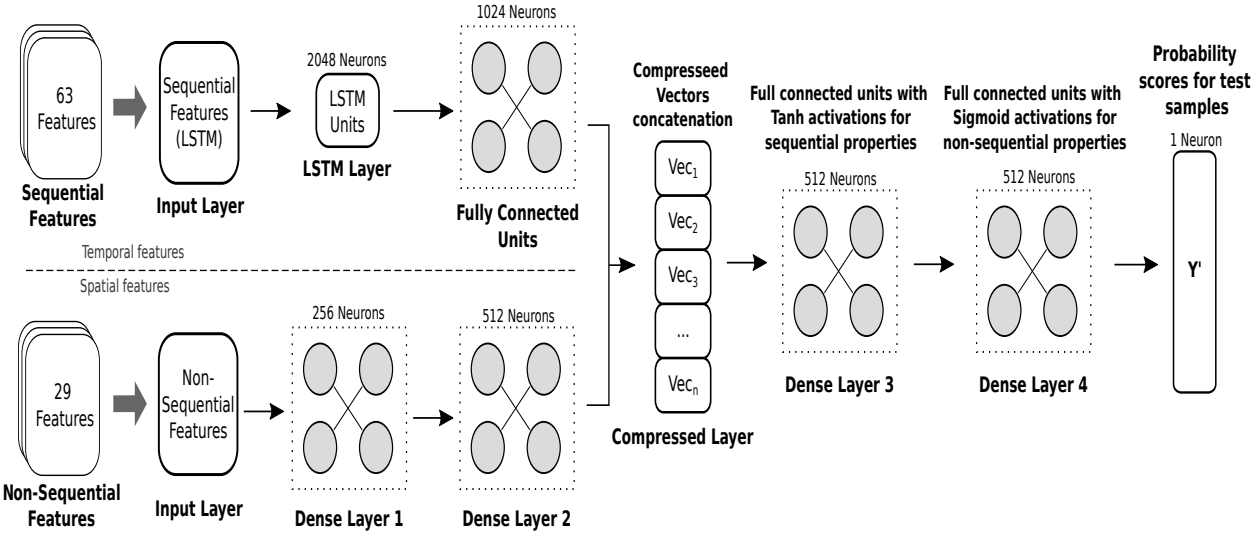

Fig. 10: The architecture of LSTM+.

highest number of votes [43], i.e., it avoids depending on the performance of a single tree.

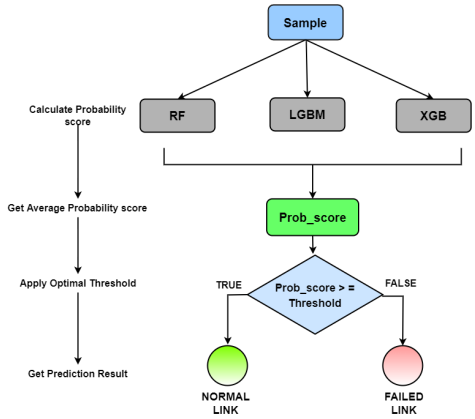

Fig. 11: The ensemble model architecture.

LGBM is a gradient boosting framework where a tree model expands horizontally, i.e., leaf-wise, instead of the traditional depth-wise expansion [44]. It chooses a leaf with max delta loss to grow. The leaf-wise expansion helps LGBM demand less resources while offering high accuracy [45]. Finally, XGBoost is a variant of LGBM for better speed and performance [46]. This is done through concurrent learning, i.e., splitting the dataset into smaller subsets and using weighted quantiles [47] to determine node splits instead of evaluating all possible splits. XGBoost also incorporates methods for determining tree splitting even with missing values. The model outcome at any instant, t, is weighed according to the model outcome at the previous instance, $t-1$.

When we implemented the above ensemble model, we considered 70:30 training and test data split as used in [6]. We deployed out-of-bag sampling as model validation during the training using 20% of the training data. The RF classifier was trained with thousand trees, each having a maximum depth of four, while the minimum sample splits were set to two to reflect the binary classification. We used two maximum terminal nodes from each tree to prevent the tree from growing further; thus, avoid model overfitting. As trees were added to the forest iteratively, the model calculated the unbiased estimate of the classification error based on the out-of-bag samples [48]. We calculated the corresponding proximity at the terminal node for each out-of-bag sample. The proximity value was increased by one when two nodes led to the same class. We repeated this process for the entire training dataset. Then, we divided the final proximity value by the number of trees in the forest for normalization.

We used *Gini impurity criterion* for the splitting decision at each node while also ensuring that this value is less for each descendant node compared to its parent node. Summing up the decrease in Gini impurity for each variable, across all trees, provides a quick measure of the variable importance in the dataset, which can further reduce the variable counts and decrease training times [48]. Finally, the model calculated the probability scores of each test sample using the out-of-bag error estimation.

LGBM and XGB were utilized to boost the performance of the RF classifier by performing optimal gradient descent along with the individual model prediction score. The number of horizontal leaves was 2^{max_depth}, where max_depth of the tree was two for the boosting algorithms. We considered low tree depth to avoid the model overfitting [48]. The average prediction score calculated from each of the three models was the final prediction score of the ensemble model. The model hyper-parameters are listed in Table VII.

Hyper-Parameter	Selected Value
Number of trees (RF)	1000
Maximum tree depth (RF)	4
Leaves per node (LGBM, XGB)	2
Maximum tree depth (LGBM, XGB)	2

TABLE VII: Hyperparameters of ensemble learning.

V. Model Evaluation Results

This section presents the evaluation results of the proposed LSTM-autoencoder model along with the existing ensemble

and LSTM+ models. We start the section by describing the used performance metrics and the evaluation testbed. Then, we show different results from the parameter tuning process followed by comparison and contrast of the proposed model with the existing ones.

A. Performance metrics

We consider the following performance metrics, where *True Positive* and *True Negative* represent the number of instances correctly classified as failure and non-failure, respectively. However, *False Positive* and *False Negative* represent the number of instances incorrectly classified as failure and non-failure, respectively. Note that we do not consider the model accuracy as an evaluation metric as it is a useful one for a balanced dataset (symmetric class ratio), which is not the case in our dataset [49].

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)} \quad (1)$$

$$Recall = \frac{TruePositive}{(TruePositive + FalseNegative)} \quad (2)$$

$$F1\_Score = 2 \times \frac{(Precision * Recall)}{(Precision + Recal}) \quad (3)$$

We also used the receiver operating characteristic curve (ROC) to represent the classification results. A ROC is a graphical representation of the model performance of classifying *True Positive Rate (TPR)* and *False Positive Rate (FPR)* for an optimal threshold. Specifically, the curve plots TPR vs. FPR at different thresholds and indicates the optimal one [50].

Testbed. We conducted all experiments on a Core i7-10700K processor running at 4.2 GHz, 32 GB DDR4 RAM, and Nvidia RTX 2080 TI GPU SLI. The OS and GPU versions were Ubuntu 18.04 LTS and CUDA 10.2.9.55, respectively, where we used CUDA to accelerate the training process.

B. Parameter Tuning

In this section, we present the results of different configuration of models, including different encoding, number of layers, and duration of past history (in number of days).

Impact of encoding methods. The impact of encoding is presented in Table VIII, where one-hot encoding is the winner. The binary encoding assigns a numerical value to each category before converting it to binary. This artificial ranking of categorical features impacts the model training. On the other hand, one-hot encoding generates binary vectors for categorical features. Each vector only has a single true value corresponding to the presence of the considered feature; thus, it avoids any bias that binary encoding faces. Specifically, one-hot encoded data offers smoother training and validation error convergence, i.e., better feature representation learning. We iteratively selected the most optimal threshold and categorized the reconstruction errors for every sample in the last training epoch. Then, we concluded that the sample was a link failure if the generated error was higher than the optimal threshold.

Once the model is trained with nine months of data from Jan to Sept 2019, we tested its performance with the remaining three months' data and the first six months' data from 2020 (Table VIII), where we calculated the Precision, Recall, and F1 score using the optimal threshold on the reconstruction error. We also applied PCA and recursive feature elimination (RFE) after each type of encoding. We found that PCA offers better prediction results compared to RFE. Thus, in the subsequent evaluations, we chose one-hot encoding and PCA.

Encoding technique	Precision	Recall	F1 Score
Binary (2019)	0.80	0.81	0.81
Binary (2020)	0.80	0.79	0.78
One-hot (2019)	0.87	0.85	0.86
One-hot (2020)	0.89	0.83	0.85

TABLE VIII: Prediction results for encoding techniques.

Impact of the number of hidden layers. Table IX shows failure prediction results for a different number of hidden layers, where four layers offer the best result. In particular, we used 2, 3, 4, and 5 layers in this test with 384, 438, 480, and 556 LSTM neurons, respectively, along with one-hot encoding and PCA. The model trained with four hidden layers performed better than the other configurations. The smaller number of layers could result in model underfitting due to the high dimensional data and complex feature correlations. Similarly, five hidden layers with 556 units could lead to the model overfitting. This added complexity forces the model to capture noise from the data to the extent that it negatively impacts the overall model performance.

Hidden layers	Precision	Recall	F1 Score
2	0.64	0.40	0.49
3	0.73	0.67	0.70
4	**0.90**	**0.87**	**0.88**
5	0.52	0.66	0.58

TABLE IX: Prediction results for different hidden layers.

Impact of the size of past samples. When we use a model like LSTM that preserves the temporal context in the data, we need to measure the impact of such context window size on the prediction. Therefore, we varied the number of days (1, 5, 7, and 10) in the past that the model received to predict failures. We observed that increasing the number of days from one to five leads to improved performance in prediction, which is presented in Table X. However, any further increase does not improve the results. We suspect these extra couple of days are further away from the past and do not significantly impact the prediction. Thus, we consider five days as the optimal past observation length. In addition to tuning parameters of the proposed LSTM-autoencoder, we do the same for LSTM+ and ensemble methods, which are presented below.

Test Data	Past Days	Precision	Recall	F1 Score
2019	1	0.87	0.85	0.86
	5	**0.90**	**0.87**	**0.88**
	7	0.90	0.86	0.88
	10	0.89	0.85	0.87
2020	1	0.87	0.85	0.86
	5	0.90	0.86	0.88
	7	0.90	0.86	0.88
	10	0.90	0.84	0.87

TABLE X: Prediction results for different number of days in the past.

LSTM+. It uses real weather data instead of the forecast one for failure prediction, which is not the case for the other two models. Thus, we first tested LSTM+ using both real and forecast data with varying past observation windows (see Table XI). The results revealed that the real-time data performed better compared to the forecast one while using the past ten days' observation. LSTM+ separates spatial and temporal data and considers real-time weather to train the model. That weather only has numerical values, i.e., fewer dimensions, without categorical ones. Thus, the architecture of LSTM+ with fewer LSTM units is more suitable for generalizing the real weather. This behavior is reflected in the results, i.e., the real weather yields better performance than the forecast. Furthermore, these fewer LSTM nodes require deeper observations in the past to learn the distribution, which yields ten days as the best observation window.

Data Type	Past Days	Precision	Recall	F1 Score
Forecast	1	0.48	0.68	0.55
	5	0.48	0.7	0.57
	7	0.42	0.67	0.51
	10	0.42	0.67	0.51
Real	1	0.57	0.59	0.58
	5	0.57	0.60	0.59
	7	0.60	0.60	0.60
	10	**0.60**	**0.62**	**0.58**

TABLE XI: Prediction results (2019) with varying past observation windows.

Ensemble learning. The authors of [6] designed an ensemble learning model that they evaluated on binary encoded balanced data. As we observed one-hot encoding and PCA offered the best prediction results, we applied the same while training the ensemble model. Furthermore, we observed that their fixed threshold of 0.50 could further be improved to 0.5802. Specifically, we computed the predicted class probabilities of an input sample as the softmax of the weighted terminal leaves from the decision tree ensemble corresponding to the provided sample. Then, we calculated the average predicted probability of the test data from individual model predictions. We chose the optimal threshold iteratively based on the maximum failure prediction performance. Table XII presents the results with a fixed vs. optimal threshold.

Data Setting	Precision	Recall	F1 Score
Fixed threshold (2019)	0.70	0.84	0.76
Optimal threshold (2019)	0.79	0.77	0.77
Optimal threshold (2020)	0.76	0.84	0.79
Fixed threshold (2020)	0.66	0.82	0.73

TABLE XII: Evaluation results of the ensemble model.

C. Performance Comparison of Different Models

Once each model is successfully trained, and associated parameters are tuned to have the model's best performance, we evaluate them over the same preprocessed data. Specifically, we compare and contrast the proposed LSTM-autoencoder with LSTM+ [7] and ensemble models [6]. The results are shown in Table XIII. The results show that the proposed LSTM-AE outperforms the other two schemes in predicting link failure in the upcoming day. More specifically, from the samples which were classified as failed links by the LSTM-AE, **87%** were actually failed links. On the other hand, **90%** of the actually failed links were correctly classified. Furthermore, LSTM-AE achieved 33% and 14% higher precision than LSTM+ and ensemble, respectively. It also has superior performance in recall and F1 scores compared to the existing solutions. Finally, the inference time of the three models is similar, with LSTM-AE slightly longer.

Model	Precision	Recall	F1 Score	Time (min)
Ensemble	0.76	0.84	0.79	5
LSTM+	0.57	0.60	0.58	5
LSTM-AE	**0.90**	**0.87**	**0.88**	6

TABLE XIII: Comparison of prediction results.

We observed that LSTM+ has the worst performance, which may have occurred due to multiple reasons. As mentioned before, fewer LSTM units in LSTM+ may not be suitable for learning high-dimensional data distribution. Moreover, the features were separated into different input layers: one trained with time-dependent features and the other with time-independent ones. We suspect that processing these separated features with architecturally different layers (LSTM and dense Units) leads to inconsistent distribution learning and data compression, which eventually translated to the worst performance.

In the case of ensemble learning, we suspect that not considering the temporal context impact its performance. However, using forecast data along with three models compensates for that price. Overall, we conclude that the proposed model has the best performance. We attribute this improvement to the strategic use of the spatial and temporal correlations of the features together in the proposed model.

D. Discussion

Throughout this work, we have demonstrated how network signal failures due to weather factors can be predicted using machine learning. We can extend this system to identify the areas most susceptible to failures and deploy a solution for circumventing them partially or entirely. One such deployment choice can be using RISs in conjunction with an SDN-based controller for supervision. Preliminary work in this space [51] describes a smart radio environment consisting of users, SDN controller, and metasurfaces (RISs) to construct an optimal route for energy efficiency. We envision that incorporating a learning-based failure prediction along with a RIS-based route construction can dynamically be managed and configured in an SDN platform, which we plan to explore as future work.

VI. Related Work

This section first presents two sets of works: investigations of the impact of weather conditions on mmWave and learning-based channel condition prediction. Then, we include RIS-based schemes in the context of 5G RAN to offer a holistic view of existing schemes.

Weather impact on mmWave. There are a number of works that demonstrate that weather fluctuation has an impact

on wireless communications like mmWave [32], [52]. For instance, the author in [32] presents that the precipitation has the most severe impact on the mmWave link, where a set of spatial links may suffer from signal attenuation or failure due to the surrounding weather like heavy rain. The author proposed periodically updating the routing topology following such disturbance using antenna alignment and disjoint route constructions. Such disjoint routing topology construction is a standard practice both in wireless [53], [54] and wired [55], [56] networks reliability. Yaghoubi *et al.* [57] also consider the impact of heavy rain in 5G networks and propose a reliable topology construction heuristic, where the failures are assumed to be spatially correlated. However, we envision the above solutions as fixing strategies after failures detection.

Learning-based failure prediction. There exist a couple of learning-based failure prediction schemes in the context of changes in weather conditions. Du *et al.* [58] proposed a Bayesian Neural Network (BNN) model to predict link failures due to wind, rain, and lightning in distributed systems. The authors claim that BNN is a good choice for an imbalanced and small dataset, but our dataset is not small. The authors in [6] focused on failure prediction in 5G RAN using the weather forecast and radio KPI data that we used in this work. They considered a supervised ensemble model consisting of Random Forest, Light Gradient Boosting Machine (LGBM), and Extreme Gradient Boosting Machine (XGB). The solution did not provide adequate focus on the data preprocessing. For instance, the authors used binary encoding, which cannot offer good performance with categorical data. Furthermore, the solution did not consider the temporal correlation in the data.

Aktaş *et al.* filled that gap and constructed an LSTM-based model that captures the temporal correlations. Specifically, they separated the time-varying weather and KPI data from the network configuration ones. The former time-varying ones are fed into an LSTM while the configuration data is passed through dense neural networks. The output from both networks is then concatenated and sent through dense neural networks again for the final outcome. Our investigation revealed that separating configuration data from the temporal one is not useful as it may lose precise context. In addition, the authors' choice of using current weather data instead of the forecast one does not seem a good fit for the future failure prediction. We propose to use an LSTM-autoencoder to jointly capture both the spatial and temporal properties using weather forecast. Zhang *et al.* combined Convolutional Neural Networks (CNNs) with LSTM to predict anomalies in cellular networks [59]. ConvLSTM is a model usually applied to image classification with a matrix as an input instead of vectors, which is the case in our failure prediction problem.

RIS-based designs. RISs consist of an array of passive optical elements, which can manipulate the properties of waves incident on it, thereby reconfiguring them according to requirements of the channel in which they propagate [60]. Works such as [61], [62] focus on RIS placement optimization, element configuration, channel estimation, beamforming, and feasible use cases. Diamanti *et al.* focus on the energy optimization of RIS in UAV-assisted backhaul networks [63]. The authors use a game-theory based approach to optimize the energy usage in an end-to-end backhaul network. They use a combination of distributed RIS-assisted signal strength optimization and bandwidth splitting. A similar optimization problem was presented in [64], where a model is developed to place RIS elements optimally as an alternative route to compensate for possible link failures. Also, machine learning has been deployed to design automated RIS-assisted systems. A number of these approaches are outlined in [65]. For instance, Liu *et al.* explored channel estimation using neural networks [66]. Deep reinforcement learning-based RIS energy optimization schemes are presented in [67], [68]. We envision that the RIS-based work can be a post-prediction solution to offer better resiliency and performance in the target area.

VII. Conclusions

mmWave radio links from radio access networks can fail because of the weather impacts. Predicting and preventing failures in advance can save time, resources, and quality of user experience in critical IoT applications. In this work, we investigated the limitations in existing link failure prediction solutions and proposed approaches and models to address them. We proposed a deep learning model of LSTM-autoencoder that can efficiently model the normal link data while generating high errors for the failed ones. Experiments on real data revealed that the proposed model could efficiently predict the failures one day ahead of the current date and classify them from the normal links. As part of our future work, we plan to explore alternative models to avoid labeling data and add explainability to the constructed models.

References

[1] J. Navarro-Ortiz *et al.*, "A survey on 5g usage scenarios and traffic models," *IEEE Comm. Surveys and Tutorials*, vol. 22, no. 2, 2020.

[2] I. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Comm. Surveys and Tutorials*, vol. 18, no. 4, pp. 2713–2737, May 2016.

[3] "Millimeter wave opens new opportunities for 5g networks," https://www.globenewswire.com/en/news-release/2020/12/02/2138610/0/en/Millimeter-Wave-Opens-New-Opportunities-for-5G-Networks.html, December 2020.

[4] JOUR, Nadeem, F.Chessa, S. Leitgeb, E. Zaman, and Safdar, "The effects of weather on the life time of wireless sensor networks using fso/rf communication," in *Radioengineering*, June 2010.

[5] J. Abdul, "Survivable millimeter-wave mesh networks," *Computer Communications*, vol. 34, no. 16, pp. 1942–1955, 2011.

[6] D. Kotagiri, A. Sawabe, and T. Iwai, "Radio link failure prediction," 2020. [Online]. Available: https://www.itu.int/en/ITU-T/AI/challenge/2020/Documents/ITUChallenge_LinkBusters.pdf

[7] S. Aktaş, H. Alemdar, and S. Ergüt, "Towards 5g and beyond radio link diagnosis: Radio link failure prediction by using historical weather, link parameters," *Computers & Electrical Engineering*, vol. 99, p. 107742, 2022.

[8] M. Y. Selim, A. E. Kamal, and F. Nait-Abdesselam, "X-haul outage compensation in 5g/6g using reconfigurable intelligent surfaces." [Online]. Available: https://arxiv.org/pdf/2207.03582.pdf

[9] P. Fiore *et al.*, "Boosting 5g mm-wave iab reliability with reconfigurable intelligent surfaces," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, August 2022, pp. 1–6.

[10] M. Diamanti, P. Charatsaris, E. Tsiropoulou, and S. Papavassiliou, "The prospect of reconfigurable intelligent surfaces in integrated access and backhaul networks," *IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING*, vol. 6, no. 2, pp. 859–872, June 2022.

[11] D. Le and N. Zincir-Heywood, "Anomaly detection for insider threats using unsupervised ensembles," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1152–1164, June 2021.

[12] F. Musumeci *et al.*, "Supervised and semi-supervised learning for failure identification in microwave networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1934–1945, Nov 2020.

[13] ITU, "Itu-challange p.036," 2020. [Online]. Available: https://aiforgood.itu.int/event/itu-ai-ml-in-5g-challenge-radio-link-failure-prediction-challenge/

[14] "Radio site KPI and weather station data," 2020. [Online]. Available: https://docs.google.com/spreadsheets/d/1KvFV5WXtRo9dU3Yak40b2G7JG8KCIqrg/edit?usp=sharing&ouid=105740223973240568967&rtpof=true&sd=true

[15] M. A. Islam, "The data and code repository," 2022. [Online]. Available: https://github.com/ArifulIslamPreence/RLFPredictionMainThesis

[16] X. Ge *et al.*, "5g ultra-dense cellular networks," *IEEE Wireless Comms.*, vol. 23, no. 1, pp. 72–79, 2016.

[17] M. K. Dahouda and I. Joe, "A deep-learned embedding technique for categorical features encoding," *IEEE Access*, vol. 9, pp. 114 381–114 391, 2021.

[18] J. Liang *et al.*, "One-hot encoding and convolutional neural network based anomaly detection." *Journal of Tsinghua University (Science and Technology)*, vol. 8, no. 2, 2011.

[19] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," 2018.

[20] Wold, Svante, K. Esbensen, and P. Geladi, "Principal component analysis," in *Chemometrics and intelligent laboratory systems*, 2020.

[21] P. Geladi and J. Linderholm, "Principal component analysis," 2020.

[22] N. V. Chawla, "Data mining for imbalanced datasets: An overview," *Data mining and knowledge discovery handbook*, pp. 875–886, 2009.

[23] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th Int. Conf on Information and Comm. Systems (ICICS)*, 2020, pp. 243–248.

[24] Bhagat, "Enhanced smote algorithm for classification of imbalanced big-data using random forest," in *IEEE International Advance Computing Conference*, 2015.

[25] N. Chawla *et al.*, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[26] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[27] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, 2014, pp. 4–11.

[28] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *ACM international conference on knowledge discovery and data mining*, 2017, pp. 665–674.

[29] Z. Chen *et al.*, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*. IEEE, 2018, pp. 1–5.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] S. Elsayed, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020.

[32] R. J, "A new approach to design of weather disruption-tolerant wireless mesh networks," *Telecommun Syst*, vol. 61, 2016.

[33] Baweja, "how to deal with missing data in python," https://towardsdatascience.com/how-to-deal-with-missing-data-in-python-1f74a9112d93.

[34] E. van der Steen, "Effects of noise in train data for neural classifiers," Ph.D. dissertation, Faculty of Science and Engineering, 2003.

[35] Li and Jia, "Deep convolutional neural network based ecg classification system using information fusion and one-hot encoding techniques." *Mathematical Problems in Engineering)*, vol. 6, no. 3, 2018.

[36] XIE and XIAOHUI, "Principal component analysis," in *Principal component analysis*, 2019.

[37] G. Buetow *et al.*, "The benefits of rebalancing," *The Journal of Portfolio Management*, vol. 28, no. 2, pp. 23–32, 2002.

[38] S. Satpathy, "Overcoming class imbalance using smote techniques," https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[39] Herman, "Time based cross validations," https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8.

[40] S. Aktaş, H. Alemdar, and S. Ergüt, "Towards 5g and beyond radio link diagnosis: Radio link failure prediction by using historical weather, link parameters," *Computers & Electrical Engineering*, vol. 99, p. 107742, 2022.

[41] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[42] Dietterich, "Ensemble learning." *The handbook of brain theory and neural networks*, vol. 2, no. 1, 2002.

[43] Ditter, "Random forest classifier for remote sensing classification." *International journal of remote sensing.*, vol. 26, no. 1, 2005.

[44] Documentation, "Light gradient boosting," https://lightgbm.readthedocs.io/en/latest/.

[45] "Light gradient boosting machine," https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/, September 2021.

[46] Saul, "Light gradient boosting machine," https://towardsdatascience.com/xgboost-extreme-gradient-boosting-how-to-improve-on-regular-gradient-boosting-5c6acf66c70a, March 2021.

[47] "weighted.quantile: Quantiles of a weighted cdf," https://arxiv.org/pdf/2203.03032.pdf.

[48] "Random forest." [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[49] Jake, "Classification evaluation: It is important to understand both what a classification metric expresses and what it hides," in *Nature Methods*, Aug. 2016.

[50] Qian, "Time-dependent roc curve analysis in medical research: current methods and applications." *BMC medical research methodology*, vol. 17, no. 1, 2017.

[51] C. Liaskos *et al.*, "Software-defined reconfigurable intelligent surfaces: From theory to end-to-end implementation," *Proceedings of the IEEE*, pp. 1–28, 2022.

[52] M. Tornatore, "A survey on network resiliency methodologies against weather-based disruptions," *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.

[53] I. Haque, S. Islam, and J. Harms, "On selecting a reliable topology in wireless sensor networks," in *Proceedings of the 2015 IEEE International Conference on Communications*, ser. ICC '15, 2015.

[54] I. Haque and D. Saha, "SoftIoT: A resource-aware SDN/NFV-based IoT network," *The Elsevier Journal of Network and Computer Applications*, vol. 193, Nov 2021.

[55] M. A. Moyeen, F. Tang, D. Saha, and I. Haque, "SD-FAST: A packet rerouting architecture in SDN," in *15th International Conference on Network and Service Management, CNSM 2019, Halifax, Canada, October 21-25, 2019*. IEEE, 2019, pp. 1–7.

[56] M. Shojaee, M. C. Neves, and I. Haque, "Safeguard: Congestion and memory-aware failure recovery in SD-WAN," in *16th International Conference on Network and Service Management, CNSM 2020, Izmir, Turkey, November 2-6, 2020*. IEEE, 2020, pp. 1–7.

[57] F. Yaghoubi *et al.*, "Design and reliability performance of wireless backhaul networks under weather-induced correlated failures," *IEEE Transactions on Reliability*, 2021.

[58] Y. Du *et al.*, "Predicting weather-related failure risk in distribution systems using bayesian neural network," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 350–360, 2020.

[59] W. Zhang *et al.*, "Self-organizing cellular radio access network with deep learning," in *IEEE INFOCOM*, 2019, pp. 429–434.

[60] Y. Liu *et al.*, "Reconfigurable intelligent surfaces: Principles and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1546–1577, 2021.

[61] J. Herbst *et al.*, "Reconfigurable intelligent surfaces: About applications and their implementation."

[62] E. Björnson, Ö. Özdogan, and E. G. Larsson, "Reconfigurable intelligent surfaces: Three myths and two critical questions," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 90–96, 2020.

[63] M. Diamanti *et al.*, "The prospect of reconfigurable intelligent surfaces in integrated access and backhaul networks," *IEEE Trans. on Green Comms. and Networking*, vol. 6, no. 2, pp. 859–872, 2021.

[64] M. Y. Selim, A. E. Kamal, and F. Nait-Abdesselam, "X-haul outage compensation in 5g/6g using reconfigurable intelligent surfaces," *arXiv preprint arXiv:2207.03582*, 2022.

[65] K. Faisal and W. Choi, "Machine learning approaches for reconfigurable intelligent surfaces: a survey," *IEEE Access*, vol. 10, 2022.

[66] S. Liu *et al.*, "Deep denoising neural network assisted compressive channel estimation for mmwave intelligent reflecting surfaces," *IEEE Trans. on Vehicular Technology*, vol. 69, no. 8, pp. 9223–9228, 2020.

[67] G. Lee *et al.*, "Deep reinforcement learning for energy-efficient networking with reconfigurable intelligent surfaces," in *IEEE Int. Conf. on Comms.*, 2020, pp. 1–6.

[68] J. Lin *et al.*, "Deep reinforcement learning for robust beamforming in irs-assisted wireless communications," in *IEEE Globecom Conference*, 2020, pp. 1–6.