Poisoning Attacks on Deep Learning based Wireless Traffic Prediction

Tianhang Zheng, Baochun Li
University of Toronto, th.zheng@mail.utoronto.ca, bli@ece.toronto.edu

***Abstract*—Big client data and deep learning bring a new level of accuracy to wireless traffic prediction in non-adversarial environments. However, in a malicious client environment, the training-stage vulnerability of deep learning (DL) based wireless traffic prediction remains under-explored. In this paper, we conduct the first systematic study on training-stage poisoning attacks against DL-based wireless traffic prediction in both centralized and distributed training scenarios. In contrast to previous poisoning attacks on computer vision, we consider a more practical threat model, specific to wireless traffic prediction, to design these poisoning attacks. In particular, we assume that potential malicious clients do not collude or have any additional knowledge about the other clients' data. We propose a perturbation masking strategy and a tuning-and-scaling method to fit data and model poisoning attacks into the practical threat model. We also explore potential defenses against these poisoning attacks and propose two defense methods. Through extensive evaluations, we show the mean square error (MSE) can be increased by over 50% to 10^8 times with our proposed poisoning attacks. We also demonstrate the effectiveness of our data sanitization approach and anomaly detection method against our poisoning attacks in centralized and distributed scenarios.**

***Index Terms*—data and model poisoning attacks, deep learning, wireless traffic prediction**

I. Introduction

Wireless traffic prediction is one of the keys to enable more intelligent wireless communications. A high-accuracy wireless traffic prediction model can guide the servers and routers to wisely organize wireless traffic and mitigate network congestion caused by unexpected traffic overhead. To facilitate intelligent wireless communications, recent advances fuse deep learning techniques and big client data for training wireless traffic prediction models, leading to substantial performance gains [1]–[4], compared to the conventional methods.

For instance, in the centralized training scenario, Wang *et al.* [1] proposed a hybrid deep learning solution for wireless traffic prediction with an autoencoder for spatial modeling and long short term memory units (LSTM) for temporal modeling. In terms of the mean square error (MSE) of the predictions, it outperformed the conventional methods, such as AutoRegressive Integrated Moving Average (ARIMA) [5] and Support Vector Regression (SVR) [6], by more than 30%. As another example, Zhang *et al.* [4] proposed a dual attention based federated learning scheme (FedDA) for wireless traffic prediction, leading to performance gains of 10% to 30% over previous baselines in the distributed training scenario.

However, current achievements of using deep learning on wireless traffic prediction are accomplished in non-adversarial environments, and it can be a completely different story in a malicious client environment. To enable model training on big client data, the cloud server has to involve a large group of clients (*e.g.,* base stations) in the training stage, which might contain quite a few malicious clients. In a centralized training scenario, the cloud server collects the training data from the clients, which provides the chance for the malicious clients to launch data poisoning attacks [7]–[11] by injecting poisoned data into the training dataset. In a distributed training scenario (*e.g.,* federated learning [12]), the cloud server aggregates the model updates from a group of clients, providing a malicious client with the chance to launch model poisoning attacks by submitting poisoned model updates to the server.

Recent studies have investigated the severe negative impacts caused by the training-stage poisoning attacks in some other applications. In computer vision, [8], [10], [11], [13] demonstrate that data poisoning can mislead deep learning models to output targeted predictions on targeted data with a small set of poisoned data. [14] shows that poisoning attacks can teach neural code autocompleters to suggest the insecure ECB mode for AES encryption or a low iteration count for password-based encryption. The data poisoning attack proposed in [15] pushes deep learning based recommendation systems to recommend attacker-chosen items to many users by injecting a few fake users with well-crafted ratings into the training dataset. [16] shows that attackers can implant a backdoor into deep learning models by submitting poisoned model updates.

Despite the recent rich research on poisoning attacks, the training-stage vulnerability of deep learning (DL) based wireless traffic prediction is still under-explored. To fill this research gap, we conduct the first systematic study on poisoning attacks against DL based wireless traffic prediction. In our study, we consider a generally more practical threat model for wireless traffic prediction, compared to some related works in the other applications, to design the poisoning attacks. Specifically, in both centralized and distributed training scenarios, we assume that a malicious client only has the access to its own traffic data, without any additional knowledge about the other client data. Also, in the distributed training scenario, we assume no collision between malicious clients, which means a malicious client needs to manipulate its poisoned model update individually. A malicious client can not manipulate multiple model updates or optimize its update based on the other client's model updates. The main benefit of assuming the above threat model is that the poisoning attacks designed under such a practical threat model can pose a real-world threat

to DL based wireless traffic prediction.

Most of the recent poisoning attacks assume that the adversary has the knowledge about the other clean data or the other clients' model updates, and thus are not directly applicable to our practical threat model. To fit data and model poisoning attacks into our threat model, we propose a perturbation masking strategy for data poisoning and a tuning-and-scaling method for model poisoning. The attempt of the perturbation masking strategy is to mimic the centralized model optimization process with limited data. In each perturbation crafting step, a malicious client randomly samples a mask ξ from a Bernoulli distribution $Bern(p)$ for each data sample and multiplies the corresponding perturbation by ξ (see Section IV-A). Then the malicious client crafts the perturbations on the surrogate models. In this regard, the surrogate models seem to be optimized on $100(1-p)\%$ clean data ($\xi = 0$) and $100p\%$ poisoned data ($\xi = 1$) in the perturbation crafting process, which is similar to the centralized model optimization process and thus improves the generalizability of the crafted perturbations. The tuning-and-scaling method is proposed for non-collusive malicious clients in the distributed scenario. In this method, a malicious client first initializes the poisoned model update as the negative of the previous global model update and then fine-tunes the poisoned update on its traffic data by maximizing the MSE. Finally, the malicious client sets a scaling factor and multiplies the update by the factor to magnify the effects of the poisoned update or bypass anomaly detection (see Section IV-B).

Beyond designing poisoning attacks, we also investigate potential defenses against the poisoning attacks on wireless traffic prediction. For data poisoning, we implement and evaluate data sanitization and randomized smoothing based defenses [17]–[19]. Since in practice, those defenses are not very effective against our data poisoning attacks on wireless traffic prediction, we propose a more suitable data sanitization method for wireless traffic data. Our intuition is that the natural wireless traffic volume rarely changes too much between two adjacent time points in practice. Thus, we define a metric as the sum of absolute traffic volume differences between two adjacent time points, called adjacent distance, and remove the data samples with the largest adjacent distances (See Section V-A). To defend against model poisoning, we implement state-of-the-art robust aggregation methods including Multi Krum, Trimmed Mean, and Median [20], [21] and evaluate their performance on wireless traffic prediction. We also design an anomaly detection method for detecting the poisoned model updates with abnormal magnitude (ℓ_2-norm).

We conduct an array of experiments on the real-world wireless traffic data from Telecom Italia [22]. We evaluate the poisoning attacks against four baselines in the centralized or distributed training scenarios, including centralized training on LSTM, centralized training on ConvLSTM*, federated averaging (FedAvg) on LSTM, and FedDA on LSTM [4]. The empirical results show that our poisoning attacks can increase the MSE of the four baselines by over 50% to more than 10^8 times (almost an arbitrary level). We also evaluate data sanitization and randomized smoothing against our data poisoning attack and Multi Krum, Trimmed Mean, Median, and the anomaly detection method against our model poisoning attack. The evaluation results show that our data sanitization approach and anomaly detection method achieve the overall best performance against our poisoning attacks.

The contributions of this paper are summarized as follows:

1) We conduct the first systematic study on the training-stage vulnerability of deep learning based wireless traffic prediction in centralized and distributed scenarios.
2) We consider a practical threat model and fit data and model poisoning attacks into the threat model with our proposed masking and tuning-and-scaling methods.
3) To defend against the poisoning attacks, we implement several defenses and design a data sanitization method and an anomaly detection method.
4) We conduct extensive evaluations to verify the effectiveness of our poisoning attacks and examine the performance of several defenses on different types of real-world wireless traffic data from Telecom Italia.

The remainder of the paper is organized as follows: We begin with introducing the background and related work in Section II. In Section III, we formulate the problem and introduce our practical threat model. We introduce the poisoning attacks in Section IV and potential defenses in Section V. We conduct extensive evaluations and show the results in Section VI. Finally, we conclude the paper in Section VII.

II. Background and Related Work

A. Definitions and Notations

In general, we denote a sequence of wireless traffic data by $\mathbf{v} = \{v_1, v_2, ..., v_T\}$ with a total of T time points. v_t refers to the traffic volume at the time point t. The attempt of wireless traffic prediction is to predict the traffic volume at a time point based on the previous traffic volumes. The previous literature on deep learning based wireless traffic prediction utilizes part of the previous traffic volumes, *e.g.*, $\{v_{t-1}, v_{t-2}, ..., v_{t-\phi 1} ..., v_{t-\phi q}\}$, to predict the traffic volume v_t. p and q refer to two sliding window sizes for capturing the dependence of v_t on the closest historical data and the periodicity of the traffic data, and ϕ refers to the period. **For simplicity, we denote $\{v_{t-1}, v_{t-2}, ..., v_{t-\phi 1} ..., v_{t-\phi q}\}$ by $\mathbf{x}_t$ and the target v_t as y_t.** We denote the dataset owned by the k-th client (*e.g.*, base station) by $\mathcal{D}_k = \{\mathbf{x}_i^k, y_i^k\}_{i=1}^{N_k}$. We denote a wireless traffic prediction model as $f_{\boldsymbol{\theta}}(\cdot)$ and its prediction as $\hat{y}_t = f_{\boldsymbol{\theta}}(\mathbf{x}_t)$, where $\boldsymbol{\theta}$ denotes model parameters. The previous literature usually trains $f_{\boldsymbol{\theta}}(\cdot)$ to minimize the mean square error between $\hat{y}_t$ and y_t, denoted by $\|\hat{y}_t - y_t\|_2^2$. We denote the perturbation added on $\mathbf{x}_i^k$ by $\boldsymbol{\delta}_{\mathbf{x}_i^k}$ and the perturbation on y_i^k by $\delta_{y_i^k}$ (poisoned data refers to $\mathbf{x}_i^k + \boldsymbol{\delta}_{\mathbf{x}_i^k}$ and $y_i^k + \delta_{y_i^k}$).

B. Wireless Traffic Prediction

In general, wireless traffic prediction is a time-series forecasting problem, which means the approaches developed for

*ConvLSTM refers to 1d convolution layer followed by LSTM

time-series forecasting are applicable to wireless traffic prediction. The earlier works on traffic prediction utilize traditional time-series forecasting methods, such as autoregressive integrated moving average (ARIMA) and support vector regression (SVR), to predict wireless/urban traffic load [5], [6], [23]. Recent advances focus on the application of deep learning techniques to wireless traffic prediction. For instance, Wang *et al.* [1] proposed a hybrid deep learning model with an autoencoder-based deep model for spatial modeling and long short memory units (LSTM) for temporal modeling. Based on LSTM, Qiu *et al.* [24] developed a centralized multi-task learning architecture to explore the commonalities and differences between cells for performance improvement. In the distributed training scenario, Zhang *et al.* [4] recently proposed to cluster the base stations based on their geometrical information and augmented traffic data and aggregate the local models under a hierarchical architecture by a dual attention based model aggregation mechanism (FedDA).

C. Data Poisoning against Deep Learning

Data poisoning refers to training-stage attacks that can degrade model performance or mislead the model to output targeted predictions by injecting a subset of carefully crafted data, referred to as poisoned data, into the training dataset. Data poisoning assumes that the adversaries only participate in data preparation, without any control on model optimization and the inference stage. In terms of the adversary's goals, existing poisoning attacks (including model poisoning attacks) can be divided into targeted and untargeted attacks. The targeted data poisoning attacks mislead the model to output targeted predictions on the targeted data. The untargeted attacks focus on degrading the model performance on all the testing data. In terms of the adversary's capabilities, existing data poisoning attacks include clean-label data poisoning, label-flipping attacks, etc. Clean-label data poisoning assumes that the adversary does not have control on the labeling process [8], [10], while label-flipping attacks refer to the attacks conducted by flipping the labels of the poisoned data [25], [26]. To mitigate the effects of data poisoning, the community has developed defenses against data poisoning based on the theory of robust statistics, differential privacy, etc. [17], [19], [27].

D. Model Poisoning against Federated Learning

In contrast to the centralized training scenario, a malicious client can directly modify the uploaded model updates in federated learning. Compared with the indirect effects caused by data poisoning on model weights, the direct manipulation on model weights by poisoned model updates substantially improves the attack performance [16]. A commonly-used method is to craft the poisoned updates on a poisoned dataset and then multiply the updates by a scaling factor to magnify their effects [16]. But the poisoned updates generated by this method can be discarded by some advanced Byzantine-robust aggregation algorithms such as Multi-Krum and Trimmed Mean (detailed in Section V-C). To improve the attack performance against Byzantine-robust aggregation algorithms, the malicious clients can instead formulate the attack as an optimization problem for optimizing the poisoned updates, with the knowledge about other clients' data or collision between malicious clients [28], [29]. Note that [28] assumes that an adversary compromises multiple worker devices, and each worker device sends a (poisoned) model update to the master device, which is similar to collision between multiple malicious clients. [29] assumes that the adversary controls multiple clients and may have access to other client data (distribution). Since our practical threat model assumes no collision between malicious clients and no additional knowledge about the other clients' data, we could not directly utilize the attacks in [28], [29].

E. Related Work

The previous works have studied data and model poisoning attacks in a wide range of applications. For instance, previous works [7], [9]–[11], [30] proposed several basic data poisoning methods and applied them to computer vision. [14], [31]–[33] developed several data poisoning attacks on natural language processing. Bagdasaryan *et al.* [16] first introduced a model poisoning attack and demonstrated its effectiveness in image classification and word prediction. Following [16], [28], [29] proposed advanced optimization methods for generating the poisoned model updates and applied them to classification tasks. Compared to computer vision and natural language processing, fewer studies focus on the training-stage vulnerability of deep learning based models for wireless communications. Ali *et al.* [34] studied the effects of poisoning attacks on deep learning based intrusion detection systems for heterogeneous wireless communications. Zheng *et al.* [11] conducted a case study to evaluate the first-order data poisoning method against network traffic classification. To our best knowledge, we conduct the first systematic study on the training-stage vulnerability of deep learning based wireless traffic prediction to data and model poisoning attacks. The previous works also developed several defense methods against data poisoning attacks, including data sanitization [17], [35], differentially-private learners [27], randomized smoothing [19], etc. Also, multiple Byzantine-robust aggregation methods were proposed to defend against the manipulated model updates uploaded by malicious clients, including Multi-Krum [20], Trimmed Mean [21], Median [21], etc.

III. Problem Formulation

A. Deep Learning based Wireless Traffic Prediction (WTP)

In this section, we first formulate the attack problem in both centralized and distributed training scenarios. We then introduce our practical threat model for wireless traffic prediction.

a) Centralized Scenario: In the centralized scenario, the participating clients agree to upload their wireless traffic data to cloud server for training a global model. After uploading the data, the clients are blind to the model optimization process, and the cloud server will train a model on the collected data by a standard model optimization scheme. As shown in Fig. 1 (left), the malicious base stations can attack the global model

by uploading carefully crafted poisoned data. In this scenario, we could formulate the attack as an optimization problem, *i.e.,*

$$\max_{\boldsymbol{\delta}_x, \delta_y} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}, y \in \mathcal{D}} \|f_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) - y\|_2^2 \tag{1}$$
$$s.t. \quad \hat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}, y \in \mathcal{D}} \|f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\delta}_x) - (y + \delta_y)\|_2^2 \ ,$$

where $\boldsymbol{\delta}_x$ and δ_y are the perturbations of the poisoned data. Since we assume that the malicious client does not have the access to the other client data, $\mathcal{D}$ refers to the training dataset owned by the malicious client in practice. We follow the previous works on data poisoning to bound the infinity norm of $\boldsymbol{\delta}_x$ and δ_y by ϵ. By default, we set ϵ as 20% of the difference between the maximum and minimum traffic volume in the malicious client's dataset. The above problem is basically a bi-level optimization problem, which can be approximately solved by the meta learning based method in [10].

b) Distributed Scenario: Since the raw traffic data might contain private information, the cloud server might switch to a distributed training scheme, *i.e., federated learning* [12], for protecting the raw traffic data. Note that the wireless traffic data from different clients usually distributes in a non-i.i.d. (independent and identically distributed) way. Thus, the previous literature employs federated learning, a widely-used distributed training technique for non-i.i.d. data, to train the prediction model. In t-th round of federated learning, the cloud server selects a subset of clients S_t and broadcasts the current global model weights $\boldsymbol{\theta}_t$ to the clients. The clients update the model weights on their traffic data and then send the model updates $\{\Delta\boldsymbol{\theta}_k^t : k \in S_t\}$ back to the cloud server. Then the cloud server aggregates the model updates into a single update to optimize the global model, *e.g.,* $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{|S_t|}\sum_{k \in S_t} \Delta\boldsymbol{\theta}_k^t$ (in **FedAvg with aggregation weights** $\frac{1}{|S_t|}$). As shown in Fig. 1, the malicious clients ($k \in \tilde{S}_t$) can attack the global model by uploading poisoned model updates $\{\Delta\tilde{\boldsymbol{\theta}}_k^t : k \in \tilde{S}_t\}$. Then the malicious client's objective can be formulated as

$$\max_{\Delta\tilde{\boldsymbol{\theta}}_k^t} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}, y \in \mathcal{D}} \|f_{\boldsymbol{\theta}^{t+1}}(\mathbf{x}) - y\|_2^2 \tag{2}$$
$$s.t. \quad \boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \frac{1}{|S_t|}\Big(\sum_{k \in S_t / \tilde{S}_t} \Delta\boldsymbol{\theta}_k^t + \sum_{k \in \tilde{S}_t} \Delta\tilde{\boldsymbol{\theta}}_k^t\Big) \ .$$

(2) adopts the aggregation rule of federated averaging (FedAvg). [4] proposes a dual attention based aggregation scheme (FedDA), and we refer the interested readers to [4] for details. Note that under our threat model, a malicious client has to maximize the MSE $\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}, y \in \mathcal{D}} \|f_{\boldsymbol{\theta}_{t+1}}(\mathbf{x}) - y\|_2^2$ without access to $\Delta\boldsymbol{\theta}_k^t$ or $\Delta\tilde{\boldsymbol{\theta}}_k^t$ from the other clients.

B. Threat Model

a) Adversary's goal: In terms of adversary's goals, poisoning attacks can be divided into two categories, *i.e.,* targeted and untargeted attacks. Generally, untargeted attacks aim at degrading the model's overall performance, while targeted attacks aim to change the predictions of certain targeted data.

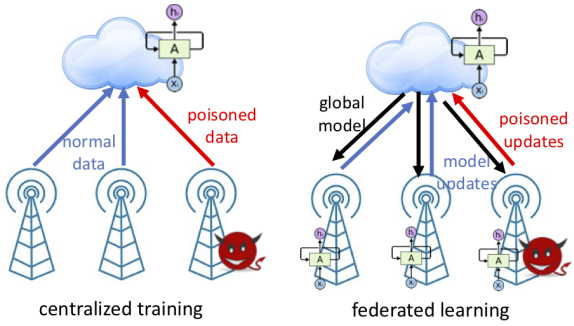

Fig. 1. Poisoning attacks against deep learning based wireless traffic prediction—data poisoning against centralized model training (left) and model poisoning against federated learning (right).

Specific to wireless traffic prediction, untargeted attacks attempt to increase the mean square error (MSE) of the model predictions on all the wireless traffic data. In this paper, we mainly focus on untargeted attacks against deep learning based wireless traffic prediction.

b) Adversary's Knowledge: We assume that the malicious client only has its own training wireless traffic data, without any additional knowledge about the other clients' data. In the centralized scenario, we assume that the malicious clients do not have any control on the model optimization process after uploading the poisoned data. Also, we assume that the malicious clients might or might not know the neural network architecture of the model used by the cloud server. If the malicious clients know the neural network architecture, they will initialize surrogate models with the same architecture to craft poisoned data. Otherwise, they will initialize surrogate models with other architectures to craft the poisoned data.

In the distributed scenario, we assume that the malicious clients only knows the global model weights without any additional knowledge about the other clients' model updates. We assume that the malicious clients can receive or evaesdrop on the global model weights broadcast by the cloud server in each training round. In practice, the server is very likely to broadcast unencrypted global model weights or encrypt the global weights with a common key since encrypting the global model weights with different secret keys for different clients will bring huge computational overhead to the server due to the large number of clients. In such cases, the malicious clients can easily obtain the global model weights in each round. Even if the malicious clients cannot obtain the global model weights in each round, they can initialize their local models with the most recent historical global model weights in their hands.

c) Adversary's Capabilities: In the centralized scenario, the malicious clients are capable of perturbing *all* of their own training data. We bound the infinity norm of the perturbations by a relatively small value, compared to the difference between the maximum and minimum traffic volume. In the distributed scenario, the adversaries can directly manipulate the model updates before sending them to the cloud server. We assume that the malicious clients do not collide in the distributed scenario. This is because collision leads to much additional

communication cost for the malicious clients. Also, collision between the malicious clients delays their response to the cloud server, which might raise suspicion from the server.

IV. Practical Poisoning against Wireless Traffic Prediction (WTP)

A. Data Poisoning against Centralized WTP

The community has developed several advanced data poisoning methods on computer vision such as [10], [11]. However, an obstacle to the application of those methods to our threat model is that a malicious client only has its own (training) traffic data, which is a small subset of the whole dataset. For a malicious client, directly executing the algorithms in [10], [11] and crafting perturbations on its own small subset might lead to sub-optimal attack generalizability in some cases. In this regard, we propose a perturbation masking strategy to mimic the centralized model optimization process with limited data, to craft more generalizable perturbations. The basic idea is to mask (conceal) the perturbations ($\boldsymbol{\delta}_{\mathbf{x}_i^k}$ and $\delta_{y_i^k}$) of some data and optimize the remaining perturbations in each perturbation crafting step. More specifically, masking

Algorithm 1 Data Poisoning against Centralized WTP

Require: Training datasets for the K clients $\{\mathcal{D}_k\}_{k=1}^K$; total number of epochs T; learning rate lr; ℓ_∞-norm bound of the perturbations ϵ; unmasking probability $p \in (0,1)$.

Malicious Client k:
Initialize M surrogate models $\{f_{\boldsymbol{\theta}_m}\}_{m=1}^M$ and pretrain them for $0 \sim M-1$ epochs on the clean $\mathcal{D}_k$.
Initialize the perturbations $\boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}$ with zero to form an enlarged training dataset $\mathcal{D}_k = \{\mathbf{x}_i^k, y_i^k, \boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}\}$.
for t = 0 to $T-1$ **do**
  Sample a minibatch from $\mathcal{D}_k$, *i.e.,* $\{\mathbf{x}_i^k, y_i^k, \boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}\}_{i=1}^{n_k}$
  For each sample, generate a mask $\xi_i^k \sim Bern(p)$.
  for m = 1 to M models **do**
    Copy $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_m$
    Optimize $\tilde{\boldsymbol{\theta}}$: $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}} - lr\nabla_{\tilde{\boldsymbol{\theta}}} \frac{1}{n_k}\sum_{i=1}^{n_k} \|f_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i^k + \xi_i^k \cdot \boldsymbol{\delta}_{\mathbf{x}_i^k}) - (y_i^k + \xi_i^k \cdot \delta_{y_i^k})\|_2^2$
    Compute the gradients of $\boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}$: $\mathbf{g}_{i,m}^k(\boldsymbol{\delta}_{\mathbf{x}_i^k}) = -\nabla_{\boldsymbol{\delta}_{\mathbf{x}_i^k}} \frac{1}{|\mathcal{D}_k|}\sum_{\mathbf{x}_i^k, y_i^k \in \mathcal{D}_k} \|f_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i^k) - y_i^k\|_2^2$ (similar for $g_{i,m}^k(\delta_{y_i^k})$). The minus sign is due to maximizing (1).
  end for
  Aggregate the gradients of $\boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}$. $\mathbf{g}_i^k(\boldsymbol{\delta}_{\mathbf{x}_i^k}) = \frac{1}{M}\sum_{m=1}^M \mathbf{g}_{i,m}^k(\boldsymbol{\delta}_{\mathbf{x}_i^k})$ (similar for $g_i^k(\delta_{y_i^k})$).
  Update $\boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}$ with Adam optimizer using $\mathbf{g}_i^k(\boldsymbol{\delta}_{\mathbf{x}_i^k})$ and $g_i^k(\delta_{y_i^k})$ as the gradients.
  Clip the elements of $\boldsymbol{\delta}_{\mathbf{x}_i^k}, \delta_{y_i^k}$ into the range of $[-\epsilon, \epsilon]$.
end for

means that we multiply the perturbations by masks $\xi = 0$ or 1. We can sample the mask ξ for each data sample from a Bernoulli distribution $Bern(p)$ (p is usually set as 0.2, *i.e.,* $P(\xi = 1) = 0.2$). Then in each perturbation crafting step, the surrogate model parameters $\tilde{\boldsymbol{\theta}}$ in Alg. 1 seem to be optimized on $100(1-p)\%$ clean data ($\xi = 0$) and $100p\%$ poisoned data ($\xi = 1$), which is similar to the centralized model optimization process. To optimize the perturbations, we solve the bi-level problem (1) by the meta-learning based method [10], [36]. Our proposed attack algorithm is detailed in Alg. 1.

B. Model Poisoning against FL-based WTP

Since we restrict the malicious clients' knowledge and capabilities to a very low and practical level, the existing advanced model poisoning attacks, such as those proposed in [28], [29], are not applicable here. Thus, we propose a practical model poisoning method. In our proposed method, a malicious client first initializes the poisoned update as the negative of the previous global model update[†], in that the malicious clients can together push the model in a similar wrong direction to increase the objective in (2) even without collision. Note that as stated in Section III-B, the malicious clients can obtain the global model weights in each round or instead use the most recent historical global model weights in their hands. After initialization, the local model is initialized with the previous global model weights. The malicious client then fine-tunes its local model on its own training dataset by *maximizing* the mean square error between the predictions and the true traffic volumes with a small learning rate (large learning rate can cause the model weights to explode in practice). This tuning operation not only degrades the performance of the local model, but also avoids the poisoned update to be easily detected by the cloud server using the previous global model update as a reference. Finally, the malicious client sets a

Algorithm 2 Model Poisoning against FL-based WTP

Require: Training datasets for the K clients $\{\mathcal{D}_k\}_{k=1}^K$; global model $f_{\boldsymbol{\theta}}(\cdot)$; local models $f_{\boldsymbol{\theta}_k}(\cdot)$; total number of rounds T; learning rate lr.

Initialize the model weights for $f_{\boldsymbol{\theta}}(\cdot)$, denoted by $\boldsymbol{\theta}^0$.
for t = 0 to $T-1$ **do**
  Cloud Server: Randomly select a subset of clients S_t and broadcast $\boldsymbol{\theta}^t$ to all the clients.
  Malicious Client $k \in \tilde{S}_t$:
  1. Initialize $f_{\boldsymbol{\theta}_k^t}(\cdot)$ with $\boldsymbol{\theta}^\tau$, where $\tau = max(t-1, 0)$.
  2. Fine-tune $f_{\boldsymbol{\theta}_k^t}(\cdot)$ on $\mathcal{D}_k = \{\mathbf{x}_i^k, y_i^k\}_{i=1}^{N_k}$ by maximizing $\frac{1}{N_k}\sum_{i=1}^{N_k} \|f_{\boldsymbol{\theta}_k^t}(\mathbf{x}_i^k) - y_i^k\|_2^2$ with learning rate $lr/10$.
  3. Set a scaling factor γ and send the update $\Delta\tilde{\boldsymbol{\theta}}_k^t = \gamma(\boldsymbol{\theta}_k^t - \boldsymbol{\theta}^t)$ to the cloud server.
  Normal Client $k \in S_t/\tilde{S}_t$:
  1. Initialize $f_{\boldsymbol{\theta}_k^t}(\cdot)$ with $\boldsymbol{\theta}^t$
  2. Update $f_{\boldsymbol{\theta}_k}(\cdot)$ on $\mathcal{D}_k = \{\mathbf{x}_i^k, y_i^k\}_{i=1}^{N_k}$ by minimizing $\frac{1}{N_k}\sum_{i=1}^{N_k} \|f_{\boldsymbol{\theta}_k^t}(\mathbf{x}_i^k) - y_i^k\|_2^2$ with learning rate lr.
  3. Send the update $\Delta\boldsymbol{\theta}_k^t = \boldsymbol{\theta}_k^t - \boldsymbol{\theta}^t$ to the cloud server.
  Cloud Server: Update the global model by $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \frac{1}{|S_t|}(\sum_{k \in S_t/\tilde{S}_t} \Delta\boldsymbol{\theta}_k^t + \sum_{k \in \tilde{S}_t} \Delta\tilde{\boldsymbol{\theta}}_k^t)$.
end for

[†]The previous global update can be computed by subtracting the (most recent) historical global model weights from the current global model weights.

scaling factor and multiplies the poisoned update by the factor. We detail how to set this scaling factor in Section VI-A.

V. Potential Defenses against Poisoning Attacks

In this section, we discuss several existing defense strategies that might be used for defending against the poisoning attacks. Also, we propose a data sanitization metric and an anomaly detection method for wireless traffic prediction.

A. Data Sanitization

The attempt of data sanitization is to examine the full training dataset and remove the outliers, which might be the poisoned data samples. A commonly-used data sanitization method is to first estimate the data centroid and then remove the data points far from the data centroid. To estimate the centroid, the sphere defense [17] simply computes the mean of the data as the estimation. Since the problem we study is not a classification problem, the slab defense [17], which projects the data points onto the line between two class centroids, is not applicable here. Note that in practice, the sphere defense is not effective against our data poisoning attacks. Thus, we propose a metric, *i.e., adjacent distance*, to examine and exclude the outliers. We define adjacent distance as

$$|y_i^k - \mathbf{x}_i^k[0]| + \sum_{j=0}^{J-1} |\mathbf{x}_i^k[j] - \mathbf{x}_i^k[j+1]|, \tag{3}$$

where $\mathbf{x}_i^k[0] \sim \mathbf{x}_i^k[J]$ refers to $v_{t-1}, v_{t-2}...$, and y_i^k is the v_t (see Section II-A). Our intuition is that the adjacent distance should be relatively small for the natural traffic data since the natural wireless traffic volume rarely changes too much between two adjacent time points in practice. We compute the adjacent distances for all the training samples and remove the samples with top $100p\%$ largest adjacent distances (p refers to the proportion of the potential malicious clients).

B. Randomized Smoothing

Randomized smoothing is an intuitive approach to corrupt the poisoned data. As mentioned in Section II-E, [19] provides certified protection against label-flipping attacks by randomizing the data labels. [27] shows differentially-private learners (*i.e., applying noise to SGD*), can provide certified protection against data poisoning. However, this protection suffers from an exponential degrade with increasing number of poisoned data samples. Thus, it is difficult for differentially-private learners to provide a certified protection against the real-world poisoning attacks. In this paper, we apply Gaussian noise to the training data $\mathbf{x}_i^k$ and y_i^k in the centralized training scenario.

C. Byzantine-Robust Aggregation

To defend against the model poisoning attacks, we consider and implement three state-of-the-art Byzantine robust aggregation methods, *i.e.,* Multi-Krum [20], Trimmed Mean [21], and Median [21]. Suppose in each round, the cloud server receive n model updates. For each model update, Multi-Krum computes a score as the sum over the distances between the model update and its $n-q-2$ nearest model updates, where q denotes the number of (potential) adversaries. The cloud server then selects the m model updates with relatively small scores, compared to the other $n-m$ updates, and use the average over the m model updates to update the global model. For each i-th dimension of the model weights, Trimmed Mean sorts the values of the i-th dimension of all the n model updates and removes the β largest values and the β smallest values. Then the average of the remaining $n-2\beta$ values is used as the update for i-th dimension of the model weights. β is usually set as the number of (potential) adversaries. Median computes the median of the values of the i-th dimension over all the n model updates, as the update for i-th dimension of the global model weights. In this paper, we implement the above three methods and evaluate them against our model poisoning attacks. We show that applying any one of the above robust aggregation methods (without anomaly detection) cannot defend against our poisoning attacks in the experiments (see Section VI-C).

D. Anomaly Detection

We observe that, to compromise the above robust aggregation methods and increase the MSE to a large value, the ℓ_2-norm of poisoned model updates has to be large. Thus, we propose an anomaly detection method based on the ℓ_2-norm of the model updates to detect or mitigate the effects of our model poisoning attacks. In each training round, we first compute the ℓ_2-norm of all the model updates. We then compute the median of those ℓ_2-norm values, denoted by μ_t. For a robust estimation on the deviation, we also take the median value of the deviations between all the ℓ_2-norm values and μ_t instead of the standard deviation as the estimation, denoted by σ_t. Our detection method involves two criteria in each round: (1) The maximum ℓ_2-norm of all the model updates should not be larger than $c_1\mu_t$ (2) The maximum ℓ_2-norm of all the model updates should not be larger than $\mu_t + c_2\sigma_t$. Note that Sun *et al.* [37] proposed a norm thresholding defense with a static threshold M. However, it is difficult to set a common static M in the whole training process for different types of traffic data. In contrast, the thresholds in our criteria are dynamic (*i.e.,* $c_1\mu_t$ and $\mu_t + c_2\sigma_t$), which depend on the current model updates. In our testbed, $c_1 = 40, c_2 = 400$ is a suitable setting for the constants to ensure that FedAvg without any malicious clients can pass the anomaly detection for all the experiments. To bypass the anomaly detection, the malicious clients have to clip the ℓ_2-norm of their poisoned model updates, which limits the effects of the poisoned model updates.

VI. Experiments

A. Experimental Setup

a) Dataset and Network Architectures: The experiments are conducted on the real-world data from Telecom Italia [22], which includes the wireless traffic records from two areas in Italy, *i.e.,* Milan and Trentino. Since the data is collected from various companies with different standards, to ease the data management, the spatial distribution irregularity is aggregated in a grid with square cells [22]. According to [22], the area of Milan is divided into a grid of 1000 cells, and the area of

Trentino is divided into 6575 cells. Since the data in each cell is logged by the corresponding base station, we could refer to the cells as base stations or clients [4]. The data contains three types of wireless traffic, including SMS, voice calls, and internet services. We randomly select 100 cells (same as [4]) from each area and conduct experiments on these 100 cells' wireless traffic data. The last 7 days of the data in each cell is retained as the testing data, and the remaining data is used as training data. **Note that we execute the poisoning algorithms on the training data, and report the MSE/MAE on the testing data.** We evaluate our attacks and defenses on two neural network architectures for wireless traffic prediction: (1) long short term memory network (LSTM) (same as [4]) (2) 1d-convolutional layer followed by LSTM (ConvLSTM).

b) Baseline Methods and Evaluation Metrics: We evaluate our poisoning attacks against four baselines for wireless traffic prediction: (1) centralized training on LSTM (2) centralized training on ConvLSTM (3) Fedavg on LSTM (4) FedDA on LSTM. In (1) and (2), the cloud server collects data from the clients and then train the model on LSTM or ConvLSTM. In this centralized scenario, a malicious client executes the data poisoning algorithm (Alg. 1) to craft the poisoned training data. In (3) and (4), the cloud server aggregates model updates from the clients, and thus a malicious client executes the model poisoning algorithm (Alg. 2) for crafting the poisoned model updates. We measure the performance of all the models by mean square error (MSE) and mean absolute error (MAE) on the testing data from all the randomly-selected 100 cells.

c) Hyperparameter Settings: Following the existing works on model poisoning attacks [28], [29], we assume that there are 20 malicious clients out of the 100 clients (cells) by default. We do not compare our attacks with [28], [29] since these attacks assume stronger threat models than ours. In the centralized scenario, we train all the models for 10 epochs on the training data (including poisoned training data) from those 100 cells, with batch size being set as 50. The malicious clients execute Alg. 1, where we set $M = 2$ and $lr = 0.01$. By default, we set $\epsilon = 0.2$, referring to 20% of the difference between the maximum and minimum traffic volume in the malicious client's training dataset. We set the initial learning rate of the Adam optimizer for crafting the perturbations as 10. The settings of the other hyperparameters of the Adam optimizer follow the default settings of Pytorch.

In the distributed scenario, we follow [4] to randomly sample 10 cells (S_t) from the 100 cells and aggregate their model updates in each training round. In each training round, the local models are trained for one epoch by the momentum SGD optimizer with $lr = 0.01$ learning rate and a momentum of 0.9 (or fine-tuned for one epoch for the malicious clients with $lr = 0.001$). We update the global model for totally 100 training rounds following [4]. For FedDA, we set $\phi = 10$ (10% augmented data); $\rho = 0.0$; $C = 16$. We refer the readers to [4] for the meanings of ϕ, ρ, and C under FedDA. If the server does not apply anomaly detection (AD), we can set a very large scaling factor $\gamma = 1000$ or 2000, which can compromise all the robust aggregation methods under the above experimental settings. With the setting $\gamma \sim \Theta(10^3)$, the poisoned model updates might increase exponentially in some cases. To avoid exploding model updates, we can bound γ by a random number sampled from $\mathcal{U}(10^4, 10^5)$ or $\mathcal{U}(10^5, 10^6)$ divided by the ℓ_2 norm of the unscaled poisoned model update. For the robust aggregation methods, we set $q = 2$, $m = 6$, and $\beta = 2$ by default (given $n = 10$). If the server applies anomaly detection, the malicious clients can set γ as $\min(a_1\|\Delta\theta^{t-1}\|_2/\|\Delta\theta_k^t\|_2, (a_2 + \|\Delta\theta^{t-1}\|_2)/\|\Delta\theta_k^t\|_2)$ to bypass the anomaly detection. Then the ℓ_2 norm of the poisoned update is scaled into $\min(a_1\|\Delta\theta^{t-1}\|_2, a_2 + \|\Delta\theta^{t-1}\|_2)$, where θ^{t-1} refers to the previous global model update. In the experiments, we sample a_1 and a_2 from $\mathcal{U}(10, 11)$ and $\mathcal{U}(1.0, 1.1)$ to bypass the anomaly detection for all the experiments. Since other settings of a_1 and a_2 might also work, the server cannot simply detect the poisoned model updates by setting certain a_1 and a_2 and comparing the ℓ_2 norm of model updates with $\min(a_1\|\Delta\theta^{t-1}\|_2, a_2 + \|\Delta\theta^{t-1}\|_2)$.

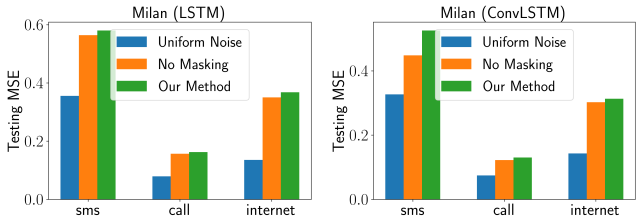

Fig. 2. Compare our poisoning method with two baselines. (1) Uniform Noise: sample the perturbation from a Uniform distribution (2) No Masking: our poisoning algorithm without perturbation masking.

B. Attack Performance

In the centralized scenario, we show the results of our data poisoning method (Alg. 1) in Table I. Our *data* poisoning method increases the MSE by over 50% in all the cases, except for centralized training on ConvLSTM on the voice calls and Internet data from Trentino. Besides, we compare our data poisoning attack with two baselines, *i.e.,* uniform noise and no masking. Uniform noise means that the malicious clients directly sample the perturbations from a uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$ and add the perturbations to the training data. No masking means applying Alg. 1 without perturbation masking to craft the perturbations. The comparison results in Fig. 2 show that our poisoning method is superior to the two baselines. Besides, we also consider the scenario that the malicious clients do not know the model architecture used by the cloud server. In such case, the malicious clients can choose another neural network as the surrogate model for crafting the perturbations. The results in Table II show that our data poisoning attack still achieves good attack performance, with another neural network as the surrogate model. In the distributed scenario, we show the results of our model poisoning method (Alg. 2). Without any defense, we show that our model poisoning attack can increase MSE to an arbitrary level (more than 10^8 times), if we set the scaling factor as $\gamma = 1000$ (One exception is that we set $\gamma = 2000$ for FedAvg

No Attack	Milan						Trentino					
	SMS		Call		Internet		SMS		Call		Internet	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
LSTM (Centralized)	0.3171	0.3223	0.0653	0.1633	0.1083	0.2290	2.0080	0.7138	0.6717	0.3703	3.8454	0.8666
ConvLSTM (Centralized)	0.3081	0.3278	0.0639	0.1659	0.1051	0.2279	2.0318	0.7077	0.7281	0.3733	2.7188	0.7795
FedAvg (Distributed)	0.3744	0.3386	0.0776	0.1838	0.1096	0.2319	2.2287	0.7416	1.6048	0.5319	4.7988	1.0668
FedDA (Distributed)	0.3411	0.3278	0.0742	0.1803	0.1061	0.2274	2.0112	0.7241	1.1704	0.4610	4.2386	0.9615
After Poisoning	**Milan**						**Trentino**					
	SMS		Call		Internet		SMS		Call		Internet	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
LSTM (Centralized)	0.5802	0.4813	0.1624	0.2728	0.3681	0.4588	3.5960	1.0841	1.1395	0.5304	6.4016	1.5196
ConvLSTM (Centralized)	0.5256	0.4468	0.1302	0.2437	0.3133	0.4022	5.1135	1.2715	0.9347	0.5069	3.9219	1.0076
FedAvg (Distributed)	1.6×10^{9}	39545	2.7×10^{9}	51671	3.5×10^{7}	5941.0	3.0×10^{9}	54845	5.4×10^{9}	73640	3.0×10^{9}	54414
FedDA (Distributed)	1.6×10^{11}	3.4×10^{5}	9.2×10^{9}	95783	6.7×10^{10}	2.6×10^{5}	6.7×10^{10}	2.6×10^{5}	1.6×10^{11}	4.0×10^{5}	1.7×10^{10}	1.3×10^{5}
After Defense	**Milan**						**Trentino**					
	SMS		Call		Internet		SMS		Call		Internet	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
LSTM + SDS	0.6428	0.5082	0.1681	0.2764	0.2391	0.3655	6.1865	1.2934	2.6180	0.7478	8.8121	1.6551
LSTM + RAND	0.5463	0.4373	0.1235	0.2338	0.3487	0.4536	3.9502	1.1164	1.1584	0.5224	6.3162	1.5049
LSTM + ADS	0.3392	0.3547	0.0902	0.2201	0.1373	0.2798	2.4320	0.8104	1.5373	0.5373	3.1065	0.9210
FedAvg + MKrum	8.9×10^{5}	945.56	14247	119.36	1.4×10^{8}	8508.8	4.5×10^{8}	9530.8	3.7×10^{8}	5818.2	5.5×10^{6}	1459.2
FedAvg + TMean	3.5×10^{5}	444.04	2.7×10^{6}	161.98	8.0×10^{8}	28328	1.2×10^{5}	275.51	2.8×10^{5}	427.78	30694	161.28
FedAvg + Median	2.4×10^{6}	342.83	3.2×10^{6}	861.09	3.8×10^{6}	151.86	3.9×10^{5}	111.89	86972	83.053	2.7×10^{5}	84.426
FedAvg + AD	0.4933	0.4244	0.3408	0.5003	0.1562	0.2832	3.1436	0.8520	1.8120	0.5951	4.5572	1.0729
FedAvg + AD + MKrum	0.4342	0.3583	0.0816	0.1867	0.1158	0.2359	4.2458	0.9830	1.7817	0.5600	5.1978	1.1011
FedAvg + AD + TMean	0.4422	0.3606	0.0897	0.1964	0.1248	0.2502	3.8271	0.9216	1.6054	0.5344	5.2416	1.1125
FedAvg + AD + Median	0.4348	0.3574	0.0881	0.1966	0.1230	0.2460	4.2736	0.9853	1.6902	0.5475	5.4982	1.1234

TABLE I

EMPIRICAL RESULTS ON THE DATA FROM TELECOM ITALIA. **CENTRALIZED SCENARIO**: SDS REFERS TO SPHERE DEFENSE (DATA SANITIZATION); RAND REFERS TO ADDING RANDOM NOISE; ADS REFERS TO ADJACENT DISTANCE BASED DATA SANITIZATION; **DISTRIBUTED SCENARIO**: AD REFERS TO ANOMALY DETECTION; MKRUM, TMEAN, MEDIAN REFER TO THE THREE ROBUST AGGREGATION METHOD (SEE SECTION V-C).

Surrogate → Target Network	SMS	Call	Internet
LSTM → LSTM	0.5802	0.1624	0.3681
LSTM → ConvLSTM	0.5220	0.1506	0.2879
ConvLSTM → ConvLSTM	0.5256	0.1302	0.3133
ConvLSTM → LSTM	0.6130	0.1454	0.2788

TABLE II

THE TESTING MSE ON A TARGET NEURAL NETWORK BY CRAFTING THE POISONED DATA ON ANOTHER SURROGATE NETWORK ($M = 2$).

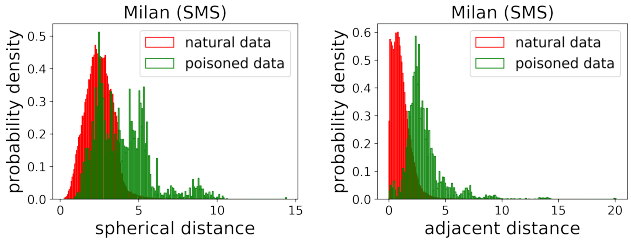

Fig. 3. Distance metrics: (1) spherical distance: the ℓ_2 distance between the mean and the data point (for sphere defense) (2) adjacent distance: our metric for data sanitization. We refer to unpoisoned data as natural data.

+ MKrum on the data from Trentino). In Section VI-C, we show that with anomaly detection and robust aggregation, our model poisoning attack can still increase MSE by about 20%.

C. Defense Performance

In the centralized scenario, we evaluate the defensive performance of sphere defense, adding random noise, and our proposed data sanitization method. The results shown in Table I indicate that our proposed data sanitization method achieves the overall best performance. In Fig. 3, we visualize the distributions of the spherical distances and adjacent distances of the natural and poisoned data. Fig. 3 shows that there is a large overlap between the spherical distance distributions of the natural and poisoned data. Thus, the outliers removed by the sphere defense [17] will include quite a number of natural data points. In this regard, sphere data sanitization (SDS) might even end up in a poorer performance compared with no defense in some cases, as shown in Table I. In contrast, the adjacent distance distributions of the natural and poisoned data are much more separable, *i.e.,* most of the poisoned data usually have larger adjacent distances. Thus, after applying adjacent distance based data sanitization (ADS), we can train a model with good performance. Interestingly, on the Internet data from Trentino, LSTM + ADS even achieves a smaller MSE compared to standard training on LSTM. We conjecture that this is because the original Internet data from Trentino already includes some outliers (thus with larger MSE compared to the other data), and removing those outliers with ADS improves the model performance on the testing data.

In the distributed scenario, we evaluate the defensive performance of Multi-Krum, Trimmed Mean, Median, and the anomaly detection method against the model poisoning attacks. As shown in Table I, without anomaly detection, our model poisoning attack can compromise Multi-Krum, Trimmed Mean, and Median with a large scaling factor $\gamma \sim \Theta(10^3)$ under the experimental settings in Section VI-A. This is because if there are 20 malicious clients out of totally 100 clients, the probability that the 10 clients sampled by the server include at least 5 malicious clients in *at least* one round of the 100 training rounds is approximately over 0.9. If this high-probability event happens, with the common setting $m = n - q - 2$ (*i.e.,* $m = 6$), the server will select 6 model updates to compute the global model update, which includes at least one poisoned update from the 5 malicious clients. The selected poisoned update can mislead the model

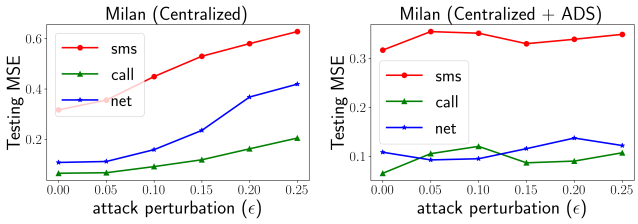

Fig. 4. The effects of ϵ on the performance of the data poisoning attack. Note that it may be easier for ADS to remove some poisoned data with larger ϵ.

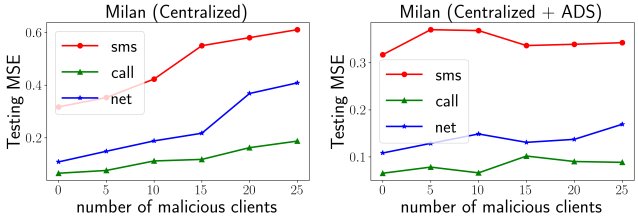

Fig. 5. The effects of the number of malicious clients on the performance of the data poisoning attack. For ADS, the proportion of removed outliers is same as the proportion of the malicious clients. Note that removing more outliers might improve the generalization performance in some cases.

due to its large ℓ_2-norm. Trimmed Mean and Median also suffer from a similar vulnerability. For Median, even if the server employs a different n, the probability that the n clients includes at least $\lceil n/2 \rceil + 1$ malicious clients in *at least* one round of the 100 training rounds is always very high. Thus, Median suffers from this vulnerability with high probability for every n. For Multi-Krum, even if the server increases q to 3 ($m = 5$), since the poisoned updates are relatively close to each other due to their similar initialization, it is still possible that the Multi-Krum method selects one poisoned model update for aggregation. However, if the server applies the anomaly detection method, the malicious clients have to set a relatively small scaling factor to bypass the anomaly detection, which limits the impacts of the poisoned updates. As shown in Table I, combination of anomaly detection and the robust aggregation methods can further improve the defensive performance in some cases. On the SMS and Internet data from Trentino, anomaly detection already achieves a good defensive performance. In such cases, robust aggregation methods might degrade the model performance since they might exclude some useful information from the model updates.

D. Ablation Studies

In this subsection, we evaluate the effects of different hyperparameters, including the attack perturbation size ϵ, the number of malicious clients, on the attack and defense performance on Milan's traffic data. In Fig. 4, we show the effects of the perturbation size ϵ in the centralized training scenario. As shown in the left of Fig. 4, as ϵ increases, the MSE of the trained model also increases. But if the server applies our adjacent distance based data sanitization method (ADS), the trained model can achieve a small MSE regardless of the perturbation size. In Fig. 5, we demonstrate the effects of the number of malicious clients. If the server does not apply any

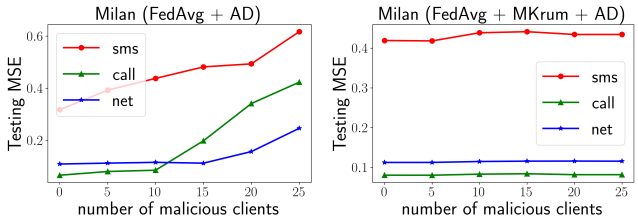

Fig. 6. The effects of the number of malicious clients on the performance of the model poisoning attack in the distributed scenario.

defense, the MSE shows a clear upward trend as the number of malicious clients increases. If the server applies the adjacent distance based data sanitization (ADS), the MSE maintains at a low level as long as the number of malicious clients keeps within a reasonable range. In the distributed scenario, if the server does not apply anomaly detection (AD), even one malicious client might compromise the model with a very large scaling factor γ. Thus, we mainly consider the scenario that the server applies anomaly detection (AD), and evaluate the effects of the number of malicious clients. In Fig. 6, we show that if the server only applies anomaly detection, as the number of malicious clients increases, the model's MSE will still increase. This is because although anomaly detection can limit the effects of the poisoned updates, it cannot eliminate the effects of the poisoned updates in each training round. Thus, a more robust defensive strategy on the data from Milan is to combine anomaly detection with a robust aggregation method, *e.g.,* Multi-Krum. With Multi-Krum, the server can remove the poisoned updates in most of the training rounds. In the remaining few rounds, even if the poisoned updates are included for updating the global model, their effects are still limited by the anomaly detection method. Thus, as shown in Fig. 6, this strategy demonstrates a more robust defensive performance, compared to anomaly detection.

VII. Conclusion

In this paper, we conduct the first systematic study on training-stage poisoning attacks against deep learning based wireless traffic prediction in both centralized and distributed training scenarios. Compared to the previous poisoning attacks on computer vision, we consider a more practical threat model, which assumes limited adversary knowledge and no collision between malicious clients. We propose a perturbation masking strategy and a tuning-and-scaling method to fit data and model poisoning attacks into our threat model. To defend against the poisoning attacks, we implement several potential defenses and propose a data sanitization approach and an anomaly detection method. Extensive evaluations on real-world data from Telecom Italia verify the effectiveness of our poisoning attacks, and also demonstrate that our data sanitization method and anomaly detection method are respectively effective defenses in centralized and distributed scenarios.

VIII. Acknowledgments

This work is supported by the NSERC Discovery Research Program.

REFERENCES

[1] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[2] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1389–1401, 2019.

[3] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, and P. Tengkvist, "A meta-learning scheme for adaptive short-term network traffic prediction," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2271–2283, 2020.

[4] C. Zhang, S. Dang, B. Shihada, and M.-S. Alouini, "Dual attention-based federated learning for wireless traffic prediction," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021.

[5] Y. Shu, M. Yu, O. Yang, J. Liu, and H. Feng, "Wireless traffic modeling and prediction using seasonal arima models," *IEICE transactions on communications*, vol. 88, no. 10, pp. 3992–3999, 2005.

[6] W.-C. Hong, "Application of seasonal svr with chaotic immune algorithm in traffic flow forecasting," *Neural Computing and Applications*, vol. 21, no. 3, pp. 583–593, 2012.

[7] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 1885–1894.

[8] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 6103–6113.

[9] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[10] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," *arXiv preprint arXiv:2004.00225*, 2020.

[11] T. Zheng and B. Li, "First-order efficient general-purpose clean-label data poisoning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[13] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *International Conference on Machine Learning*, 2019, pp. 7614–7623.

[14] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[15] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *28th Annual Network and Distributed System Security Symposium, NDSS 2021*, 2021.

[16] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[17] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 3520–3532.

[18] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[19] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8230–8241.

[20] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[21] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[22] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Scientific data*, vol. 2, no. 1, pp. 1–15, 2015.

[23] B. Zhou, D. He, and Z. Sun, "Traffic modeling and prediction using arima/garch model," in *Modeling and simulation tools for emerging telecommunication networks*. Springer, 2006, pp. 101–121.

[24] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 554–557, 2018.

[25] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian conference on machine learning*. PMLR, 2011, pp. 97–112.

[26] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines." in *ECAI*, 2012, pp. 870–875.

[27] Y. Ma, X. Z. Zhu, and J. Hsu, "Data poisoning against differentially-private learners: Attacks and defenses," in *International Joint Conference on Artificial Intelligence*, 2019.

[28] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[29] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *28th Annual Network and Distributed System Security Symposium, NDSS 2021*, 2021.

[30] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," *arXiv preprint arXiv:2009.02276*, 2020.

[31] M. Jagielski, G. Severi, N. P. Harger, and A. Oprea, "Subpopulation data poisoning attacks," *arXiv preprint arXiv:2006.14026*, 2020.

[32] W. Yang, L. Li, Z. Zhang, X. Ren, X. Sun, and B. He, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 2048–2058.

[33] E. Wallace, T. Zhao, S. Feng, and S. Singh, "Concealed data poisoning attacks on nlp models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 139–150.

[34] M. Ali, Y.-F. Hu, D. K. Luong, G. Oguntala, J.-P. Li, and K. Abdo, "Adversarial attacks on ai based intrusion detection system for heterogeneous wireless communications networks," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–6.

[35] J. Steinhardt, M. Charikar, and G. Valiant, "Resilience: A criterion for learning in the presence of arbitrary outliers," in *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[36] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[37] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.