Adversarial Attacks and Defenses in Multivariate Time-Series Forecasting for Smart and Connected Infrastructures

Pooja Krishan [1], Rohan Mohapatra [2], Sanchari Das [3]and Saptarshi Sengupta [4]

[1,2,4] *Department of Computer Science, San José State University, San José, CA, 95192, USA*

[3] *Department of Computer Science, University of Denver, Denver, CO, 80210, USA*
pooja.krishan@sjsu.edu
rohan.mohapatra@sjsu.edu
sanchari.das@du.edu
saptarshi.sengupta@sjsu.edu

ABSTRACT

The emergence of deep learning models has revolutionized various industries over the last decade, leading to a surge in connected devices and infrastructures. However, these models can be tricked into making incorrect predictions with high confidence, leading to disastrous failures and security concerns. To this end, we explore the impact of adversarial attacks on multivariate time-series forecasting and investigate methods to counter them. Specifically, we employ untargeted white-box attacks, namely the Fast Gradient Sign Method (FGSM) and the Basic Iterative Method (BIM), to poison the inputs to the training process, effectively misleading the model. We also illustrate the subtle modifications to the inputs after the attack, which makes detecting the attack using the naked eye quite difficult. Having demonstrated the feasibility of these attacks, we develop robust models through adversarial training and model hardening. We are among the first to showcase the transferability of these attacks and defenses by extrapolating our work from the benchmark electricity data to a larger, 10-year real-world data used for predicting the time-to-failure of hard disks. Our experimental results confirm that the attacks and defenses achieve the desired security thresholds, leading to a 72.41% and 94.81% decrease in RMSE for the electricity and hard disk datasets respectively after implementing the adversarial defenses.

1. INTRODUCTION

A time-series records a series of metrics over regular intervals of time as a sequence of values. Time-series forecasting refers to the task of estimating the output at a certain time step, given the previous values. It is used in a variety of domains such as finance (Sezer, Gudelek, & Ozbayoglu, 2020), power consumption prediction (Divina, García Torres, Goméz Vela, & Vázquez Noguera, 2019), health prediction of equipment (C.-Y. Lin, Hsieh, Cheng, Huang, & Adnan, 2019), healthcare (Kaushik et al., 2020), and weather forecasting (Karevan & Suykens, 2020). The widespread use of sensors and actuators has resulted in a proliferation of data, leading to the shift from traditional time-series forecasting methods to deep learning architectures (Siami-Namini, Tavakoli, & Siami Namin, 2018), which are more capable of gleaning insights and identifying long-term trends from the data. However, it is a double-edged sword as deep learning models can be easily compromised by attacks, causing the models to produce incorrect forecasts based on manipulated input data. This gullible nature of deep learning models to attacks paves the way for catastrophic failures in safety-critical applications and leads to the wastage of valuable resources, time, money, and productivity (Akhtar & Mian, 2018). This opens up a new area of research to develop models resistant to these types of attacks.

Adversarial attacks on deep learning models are classified into white-box or black-box attacks, and targeted or untargeted attacks depending on the ease of access, and the attacker's goal respectively. In white-box attacks, the attacker knows sensitive model-specific information such as inputs, targets, and gradients (Melis et al., 2021). Conversely, in black-box attacks, the model is viewed as an oracle that outputs values given input data and the attack is crafted based on observed model behavior (Oh, Schiele, & Fritz, 2019; Tsingenopoulos, Preuveneers, & Joosen, 2019). In targeted attacks, the adversary tries to not only delude the model but also prompts it to produce an output from a particular distribution (Fursov et al., 2021) whereas in untargeted attacks the attacker intends to trigger the model to generate incorrect out-

puts belonging to any distribution (Miller, Xiang, & Kesidis, 2020; J. Lin, Dang, Rahouti, & Xiong, 2021).

Adversarial defense involves training the deep learning network with augmented data that captures the noise distribution that the attacker plans on using, or modifying the model architecture to enhance the robustness of the model (Tariq et al., 2020; Akhtar, Mian, Kardan, & Shah, 2021).

In this study, we train a Long Short-Term Memory (LSTM) model (Hochreiter & Schmidhuber, 1997) on a toy dataset that is used to predict future power consumption, given the past values to avoid under-utilization or over-utilization of resources. After conducting numerous training experiments, we select the best-performing LSTM model. We then subject this model to two untargeted white box attacks – the Fast Gradient Sign Method (FGSM), and the Basic Iterative Method (BIM), causing it to learn the underlying distribution incorrectly and leading to an increased error rate. Once we demonstrate the impact and the ease of the attacks, we move on to implement adversarial defense using both a data augmentation-based approach, and a layer-wise hardening of the neural network weights enabling the model to learn over the poisoned noise distribution in addition to the actual training samples. We then repeat the above set of experiments on a large-scale hard disk drive dataset that predicts the Remaining Useful Life (RUL) to show the transferability of the attacks and defense schemes proposed.

Our key contributions are:

1. **Efficient training:** We run multiple experiments to identify the best deep learning model.
2. **Effective attacks:** We successfully demonstrate the impact of the adversarial attacks in all the datasets used.
3. **Risk mitigation:** We perform different adversarial defenses to develop models resilient to attacks.
4. **Imperceptibility of perturbation:** We visualize the indiscernible changes to the input after the attack.
5. **Widespread applicability:** We use two datasets to prove the efficacy and transferability of the attacks and defenses.

The paper is outlined as follows: In Section 2, we will go over previous literature and identify the opportunities in this domain. In Section 3 we will walk through the preprocessing of the datasets used in the experiments. In Section 4, we provide insights into the deep learning models and training parameters used. In Section 5 we summarize the overall attack and defense schemes used in this paper. We summarize the results in Section 6. In Section 7 we outline the future directions, and finally conclude our work in Section 8.

2. RELATED WORK

(Stergiou & Psannis, 2017) outline the interdependence between Internet of Things (IoT) and Big Data. With the proliferation of sensors that record and share information between devices on the Internet, there is no shortage of data, and developing deep learning models to predict future sensor values has become easy. This has led to a shift from traditional methods of time-series forecasting using model-driven methods to a data-driven method involving multiple deep learning models for time-series forecasting problems (Faloutsos, Gasthaus, Januschowski, & Wang, 2018). (Muzaffar & Afshari, 2019) found that LSTM models are better at predicting electricity consumed, given exogenous attributes such as temperature, humidity, wind speed, etc. (Wu, Liao, Miao, & Du, 2019) prove that using a Gated Recurrent Unit (GRU) (Cho et al., 2014) to predict power consumed in New South Wales in Australia led to much better forecasting results than using traditional models. (Mishra, Basu, & Maulik, 2019) use a dilated temporal CNN to capture load consumption using multiple synthetic and real-world datasets accurately. (Bohan & Yun, 2019) prove that a Bidirectional Recurrent Neural Network (BRNN) is better at forecasting traffic flows using the GPS data collected from the Hohhot Bus Corporation than LSTM or GRU models. (Orimoloye, Sung, Ma, & Johnson, 2020) showcase the advantages of using deep networks in comparison to shallow ones while predicting stock prices. (Yan & Ouyang, 2018) compare conventional machine learning models to deep learning ones which use a combination of wavelet decomposition and LSTMs in stock market predictions. (Chen, 2024) use an LSTM working through the attention mechanism to predict the RUL of aircraft engines. (Al-Dulaimi, Zabihi, Asif, & Mohammadi, 2019) have used a neural network model combining LSTM and Convolutional Neural Network (CNN) (LeCun, Bottou, Bengio, & Haffner, 1998) to predict the RUL of aircraft engines. (Deutsch & He, 2018) show the advantages of using a deep learning model to predict the RUL of spinning parts.

Despite these advances, deep learning models remain vulnerable to adversarial attacks. (Szegedy et al., 2014) reported vulnerabilities in deep learning models in image recognition, where Convolutional Neural Networks (CNNs) can be manipulated by injecting minute modifications into the data, thereby leading the network to miscalculate the input with high conviction. (Akhtar & Mian, 2018) have presented a survey of all types of antagonistic manipulations and their impacts on image recognition such as self-driving cars (Eykholt et al., 2018), robotic vision (Melis et al., 2017), cyberspace attacks (Papernot et al., 2017), etc. With the development of multiple threat models to perform adversarial attacks, it is no surprise that a lot of effort went into developing adversarial defense strategies for these attacks in image recognition. (Akhtar et al., 2021) summarize the adversarial attacks and defense mechanisms developed in computer vision in recent years.

In recent times, adversarial attacks and defense strategies used in time-series analysis have garnered the interest of re-

searchers. (Karim, Majumdar, & Darabi, 2020) have proposed an Adversarial Transformation Network (ATN) to attack univariate temporal sequential data models used in classification tasks. They have also successfully defended against these attacks using a naive method of data augmentation. (Harford, Karim, & Darabi, 2020) have extended the previous work to multivariate time-series classification problems. (Rathore, Basak, Nistala, & Runkana, 2020) have demonstrated the effect of FGSM, BIM, and Universal targeted and untargeted attacks on univariate time-series classification tasks, and defense based on the primitive data augmentation technique. (Mode & Hoque, 2020) have proposed traditional attacks on multivariate time-series regression datasets such as the Google Stock dataset from Nasdaq and the Electricity dataset (Hebrail & Berard, 2012). They show the vulnerability of CNNs, LSTMs, and GRUs. (Govindarajulu, Amballa, Kulkarni, & Parmar, 2023) have carried out targeted attacks based on amplitude, direction, and temporal components of the model and showed their effectiveness using statistical tests on the Google stock exchange and Electricity datasets. This is visually represented in Figure 1.

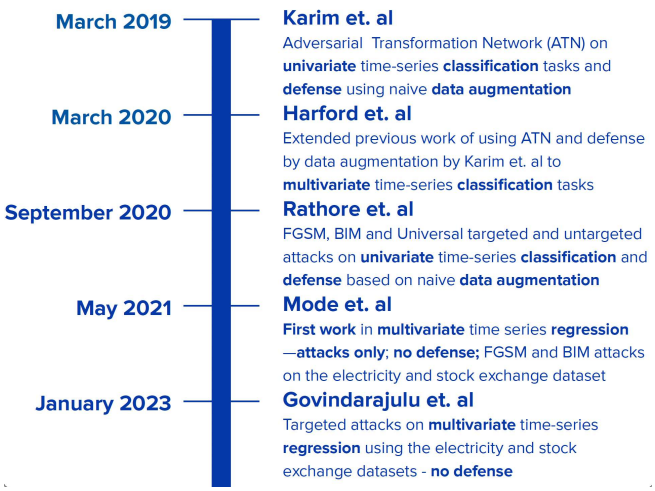


Figure 1. Overview of previous attacks and defenses work in the time-series forecasting domain.

Our extensive review of the literature opened up a realm of opportunities for improvement of current state-of-the-art methods. We observed the following:

1. All previous work has been done only on toy datasets such as the Electricity and stock exchange datasets
2. Adversarial defense has not been performed on the multivariate time-series datasets
3. The literature also lacked in demonstrating the imperceptible nature of the adversarial attacks on visualization of training sample inputs

We were motivated to address the research gaps by:

1. Extending our experiments on the toy Electricity dataset to a real-world dataset predicting the Remaining Useful Life (RUL) of Hard Disk Drives (HDDs)
2. Performing adversarial defenses:
   (a) Using data augmentation-based adversarial training
   (b) Using model hardening techniques that perturb the gradients of the model during the training process to successfully defend against adversarial attacks
3. Demonstrating the indiscernible nature of the attacks to the naked-eye by visualizing the training sample inputs to the machine learning models, after performing the adversarial attacks

3. EXPERIMENTAL SETUP

We first carry out our experiments on a smaller dataset which is used to predict the power consumed sometime in the future, given the past readings. Once we have shown the success of the attacks and defense techniques in the electricity dataset, we repeat the experiments on a substantially larger dataset which is used to predict the RUL of HDDs proving the ease of transferability of these attacks and defenses. We describe the datasets used in this research and the preparation steps done to make the data more viable for ingestion by the deep learning models, in this section.

3.1. Individual Household Power Consumption Dataset

Power consumption prediction is a vital task to estimate the amount of power that has to be supplied to various locations at any given time. It has a direct bearing on the environment and helps to cut costs. Motivated by the applications of power consumption prediction and to demonstrate the effects of antagonistic manipulation and fortification on a multivariate time-series dataset, we chose the Electricity dataset from the UCI machine learning repository (Hebrail & Berard, 2012) in this research.

The Electricity dataset from the UCI machine learning repository consists of over 2 million rows and 9 columns sampled by the minute in a household for 4 years from the end of 2006 to the end of 2010. *global active power* refers to the total actual power consumed in kW and *global reactive power* corresponds to the unused power in the transmission wires in kW. *voltage* and *global intensity* represents the mean voltage in volts and mean current in amperes respectively. *sub metering 1* refers to the total energy consumed in the kitchen in Watt-Hour, *sub metering 2*, refers to the active energy readings in the laundry room in Watt-Hour, and *sub metering 3* represents the total energy consumed by the electric water heating and air conditioning equipments in Watt-Hour.

While preprocessing the dataset, we treated the missing values represented by *'?'* as null values represented by *'NaN'* for simplicity. Each column contains *'NaN'* values, and since

machine learning models do not handle missing data very well, we replace the null values in each column with the mean of the respective column values. To ensure that the values between columns are in a comparable scale, the values in the dataset are normalized using the smallest and largest values of the samples and they all lie in the range 0 and 1 to ensure consistency in predictions. We resampled the dataset daily, by extrapolating the average of the per-minute values as the value of the samples per day thereby generating a dataset with 1400 samples to predict *global active power*. From Figure 2, which shows the distribution of *global active power* per day, per week, per month, and per quarter, it is evident that the periodicity of the distribution decreases as the time interval increases. This suggests that using more samples from the past does not contribute in making accurate predictions into the future.

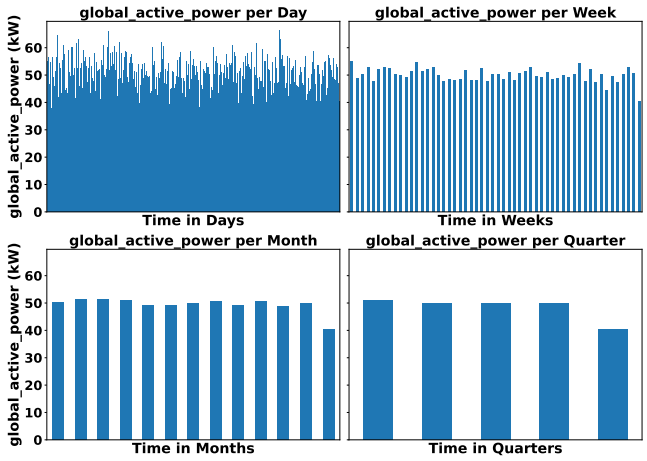


Figure 2. Periodicity decreases as time interval increases.

3.2. Backblaze Hard Disk Drive Dataset

Before the surge in big data, model-driven approaches to Prognostic Health Management (PHM) depended on reactive maintenance and preventative maintenance techniques. In reactive maintenance, the hard disks are replaced only after failure resulting in disruption of normal operations until the drive is fixed. In preventative maintenance, the hard disks are replaced well before failure leading to the replacement of a fully functioning disk and resulting in wastage of resources. In data-driven approaches, predictive maintenance is performed using the Remaining Useful Life (RUL) metric. RUL is an important metric used to indicate the time to failure of any equipment. To prove the transferability of the attacks and defenses on any real-world dataset, we employ the data store from Backblaze (2023) housing the hard drive sensor readings for a period of 9 years from 2014 to 2022. We concentrate on the Seagate family of hard disk drives since they have the most amount of reliable data (Mohapatra, Coursey, & Sengupta, 2023).

The dataset consists of attributes such as *date, serial_number, model, capacity, failure* and multiple *S.M.A.R.T features*. S.M.A.R.T records various attributes of the hard disk drive. The date represents the time the reading is recorded in YYYY-MM-DD format, serial_number, and model columns represent the serial number and model number assigned by the manufacturer. The model number of the dataset used is ST4000DM000 where ST stands for Seagate. The capacity column refers to the capacity of the HDD, and the final failure column consists of a binary 0/1 value which represents whether the hard disk drive has failed or not. Using the failure column, the RUL column is created to generate 5 different types of datasets each giving the time to failure of the disk starting from 5, 15, 25, 35, or 45 days. Remaining Useful Life can be calculated from the degradation curve shown in Figure 3.

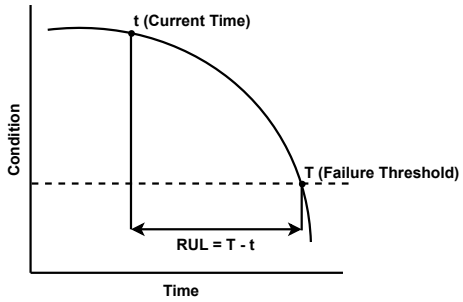


Figure 3. Remaining Useful Life curve.

If t is the current time at which the disk is healthy and functioning properly, and T is the time at which the disk fails (given by the failure column), then the RUL of the disk is given by T-t. In other words, the day before the failure is marked as 1, the penultimate day before the failure is marked as 2, and so on up to 5, 15, 25, 35, and 45 days generating 5 different kinds of datasets with an increasing number of samples with an increase in look back days for the experiments. RUL is given by:

$$\text{RUL} = \text{Date of failure of HDD} - \text{First log date of HDD} \quad (1)$$

The dataset can now be modeled as a multivariate time-series problem (Mohapatra & Sengupta, 2023). The dataset is preprocessed by dropping any null or missing values and by normalizing the data values between 0 and 255 which is the standard proposed by Backblaze to account for the wide fluctuation of data values between 0 and 10^{14}.

4. TRAINING PROCESSES

We use a vanilla LSTM model (Hochreiter & Schmidhuber, 1997) on the Electricity dataset and an Encoder-Decoder LSTM model (Sutskever, Vinyals, & Le, 2014) on the HDD dataset. We use the Python programming language in con-

junction with the tensorflow and keras libraries to perform our experiments on the Intel Xeon CPU with 13GB RAM. The next two subsections consist of details about the experiments performed.

4.1. Individual Household Power Consumption Dataset

We propose a vanilla LSTM model due to the fixed number of attributes and target values and also because it is the most prevalent model for multivariate time-series regression tasks on the Electricity dataset (Alden, Gong, Ababei, & Ionel, 2020) (Ibrahim, Megahed, & Abbasy, 2021) (da Silva, Geller, dos Santos Moura, & de Moura Meneses, 2022).

The dataset is divided into 80%, 10%, and 10% for the training, validation, and test sets respectively. We trained the vanilla LSTM model consisting of a sequential layer followed by 100 hidden nodes with the ReLu activation function. ReLU overcomes the problem of vanishing gradients by outputting a value equal to the input if the input is positive. We also added a 10% dropout to regularize the network and prevent overfitting. Finally, we added a dense fully connected layer to the LSTM. Adam (Kingma & Ba, 2017) is used as the optimizer and the metric defined is Root Mean Squared Error (RMSE).

We divided the experiment into 3 parts to validate our findings and the behavior of the model:

1) Without using cross-validation: We trained the vanilla LSTM model to predict the *global active power* target variable by using the same validation set every time to inform training. This training process only looked back 1 day before to predict the next day's power consumption.

2) Using walk-forward cross-validation: We repeated the experiment with 3, 5, and 10-fold cross-validation. Since classical k-fold and stratified cross-validation schemes shuffle the data and disrupt the order and seasonality of time-series input, we used walk-forward cross-validation. In walk-forward cross-validation, the first few data points in a finite window correspond to the training set and the next few data points correspond to the validation set. In the next iteration, more data points that were formerly in the validation set are included in the train set, and the window is expanded to include subsequent data points in the validation set as shown in Figure 4. Based on the amount of folds, this process is repeated and the average RMSE score is reported as the final training and validation RMSE scores. These k-fold cross validation schemes were also carried out by only looking back 1 day into the past to predict the next day's consumption.

3) Using look-back: The vanilla LSTM model trained so far looked back one day in the past to predict the future. We conducted experiments by increasing the look-back window size to allow the model to consider more samples from the past while making future predictions. The look-back window

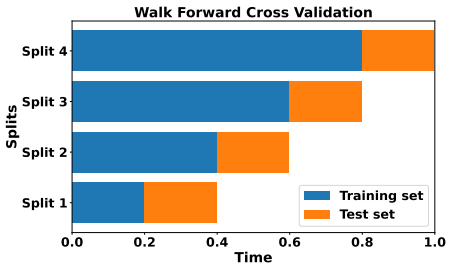


Figure 4. Walk-forward cross-validation.

sizes we used in our experiments are 3, 6, 9, 12 and 15 days. The graph of the RMSE values for the train and test sets for varying look-back window sizes are shown in Figure 5. It is evident that as the look-back interval increases, the test set error increases, showing that looking further into the past worsens predictions. This can be attributed partly to the fact that LSTMs do not work well with long sequences and partly to the property inherent in the dataset, where increasing the time interval decreases the periodicity as shown in Fig. 2. Since we have already down-sampled the dataset from minutes to days, samples further in the past do not contribute to the seasonal trends.

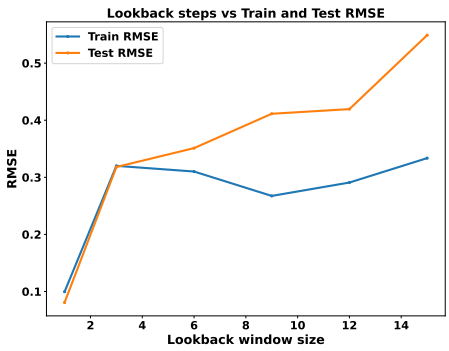


Figure 5. Error rate for varying look back window sizes.

The RMSE scores of the train, validation, and test sets produced while running the experiments are tabulated in Table 1. It is clear that the train RMSE is consistently higher than the test RMSE in all types of experiments performed. To eliminate sampling bias as a reason for this, we split the dataset into 60-20-20 to check if the test RMSE is still lower than the training RMSE. We found the test RMSE (0.0746) to still be lower than the train RMSE (0.1053) elucidating that the test set distribution mirrors the training set distribution due to the conspicuous seasonality inherent in the dataset making the process of predicting the *global active power* easier, once the model has learned using the training set.

It is seen that 3-fold walk-forward cross-validation with 1-day look-back gives the best test RMSE result. Figure 6 shows the actual and predicted values.

We conclude our experiments and select the 3-fold walk-forward cross-validation model using 1-day look-back and no

Table 1. Experimental Results of the Vanilla LSTM Network on the Unpoisoned Dataset

Experiments	Train	Validation	Test
Without using CV	0.0997	0.0703	0.0807
Using 3-fold CV	0.1120	0.0849	**0.0764**
Using 5-fold CV	0.1166	0.0858	0.0805
Using 10-fold CV	0.1156	0.0857	0.0805
Using 3-day look-back	0.3241	0.3073	0.3181
Using 6-day look-back	0.3102	0.2758	0.3512
Using 9-day look-back	0.2674	0.2744	0.4113
Using 12-day look-back	0.2909	0.1805	0.4195
Using 15-day look-back	0.3336	0.1830	0.5490

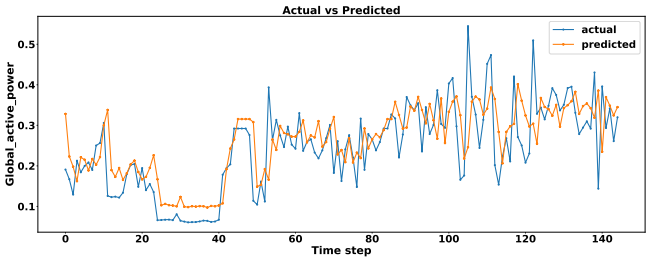


Figure 6. True vs. predicted values of the best LSTM model.

feature selection as the model to conduct further experiments on.

4.2. Backblaze Hard Disk Drive Dataset

We propose an Encoder-Decoder LSTM (Sutskever et al., 2014) as the model to learn the underlying distribution of the HDD dataset from Backblaze. We chose this model architecture because of the varying nature of the input sequence due to the addition of S.M.A.R.T features over the years, and also since we are aiming to predict a sequence of RUL values given the sequence of inputs. This Encoder-Decoder model generates the most likely sequence given a sequence of data. The encoder scans the input and outputs a constant-size array called the context vector, and the decoder reads from the context vector.

Similar to the electricity dataset, this dataset is split following the 80-10-10 rule for the train, validation, and test sets respectively. Our Encoder-Decoder LSTM model consists of a Sequential Layer of one hundred hidden units with ReLu activation. We look 5, 15, 25, 35, and 45 days into the past to check the effect of feeding more past data to the model. Since we are looking back 't' periods, the output of the encoder is repeated 't' times before passing it through another ReLu layer with 100 units We added a 10% dropout to prevent overfitting and help in generalization. A dense layer is added to every period of the decoder's output series using the *Time Distributed* wrapper thereby enabling the model to predict the RUL for each time step. Adam optimizer is utilized to fine-tune the weights during training, and RMSE is used as metrics. For higher lookback time steps, the gradients can become excessively large, a phenomenon often referred to as *'exploding gradients.'* To prevent NaN metrics, the gradients are clipped if they exceed 0.5.

We divide our experiments into two to choose the top-performing model to execute the attacks and defenses as follows:

1) Using look-back: We facilitate the learning of the Encoder-Decoder LSTM model by providing the five different kinds of datasets generated in Section 3.2. From Figure 7, it is clear that increasing the look-back window exacerbates predictions since the main disadvantage of an Encoder-Decoder LSTM is its inability to handle long sequences. We conduct cross-validation experiments on the 5-day look-back dataset.

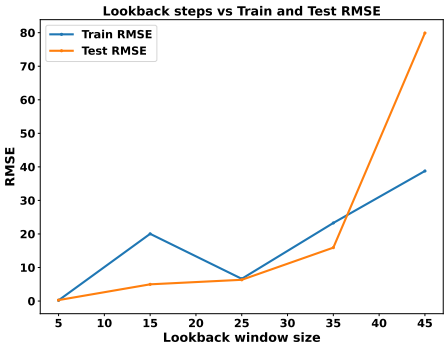


Figure 7. Error rate for varying look back window sizes.

2) Using walk-forward cross validation: Like the electricity dataset, we perform 3, 5, and 10-fold walk-forward cross-validation on the 5-day look-back dataset. The results are tabulated in Table 2.

Table 2. Results of the LSTM Network on the Unpoisoned Dataset

Experiments	Train	Validation	Test
Using 3-fold CV	0.0677	1.1446	0.3084
Using 5-fold CV	0.0808	0.4718	0.2085
Using 10-fold CV	0.1065	0.8493	**0.0715**
Using 5-day look-back	0.2156	1.0192	0.2886
Using 15-day look-back	20.0390	6.0372	4.9856
Using 25-day look-back	6.6315	7.5154	6.3345
Using 35-day look-back	23.2777	19.0839	15.9206
Using 45-day look-back	38.7680	52.4585	79.9001

It is seen that 10-fold walk-forward cross-validation on the dataset which looks back 5 days gives the best results. Figure 8 shows the true versus predicted values of this Encoder-Decoder LSTM model.

We choose the 10-fold cross validated 5-day look-back model as the final one to attack and defend against.

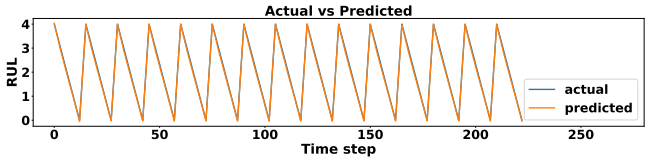


Figure 8. True vs. predicted values of the best Encoder-Decoder LSTM.

5. ATTACKS AND DEFENSE STRATEGIES

We illustrate the process of performing adversarial attacks and adversarial defenses in this section.

5.1. Overview of Process Flow

We demonstrate that the adversarial attacks are successful and exploit the sequential nature of deep learning networks and their susceptibility to adversarial perturbations using FGSM, and BIM attacks. Once the best LSTM models selected in Section 4.1 and Section 4.2 succumb to the two types of attacks mentioned above, we perform adversarial defenses using data augmentation at the data plane, and layer-wise perturbations at the gradient plane to inform the training process. This ensures model robustness to adversarial attacks. The entire process flow is summarized in Figure 9.

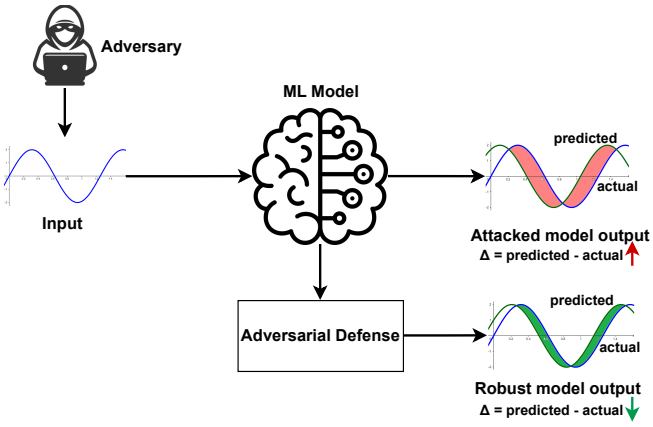


Figure 9. Overall System Architecture.

5.2. Adversarial Attacks

We perform two types of adversarial perturbations of the data.

5.2.1. Fast Gradient Sign Method (FGSM)

(Goodfellow, Shlens, & Szegedy, 2015) proposed FGSM to perform adversarial perturbations on CNNs for image data. The FGSM logic given by Algorithm 1, adds disturbances to the input in the direction of the gradients with regard to the loss function of the data. We have extended the algorithm to multivariate time-series datasets described earlier.

Algorithm 1 FGSM Algorithm

Require: Mapping ϕ from X to y, perturbation magnitude ϵ, features or attributes X, label or target y
Ensure: X_ϵ: adversarial examples for different values of ϵ
$X_\epsilon \leftarrow \emptyset$
for $\epsilon \leftarrow$ start to end **do**
  $X_{\text{adv}} \leftarrow \emptyset$
  for $i \leftarrow 1$ to m **do**
    $X_{\text{adv}}(i) \leftarrow X(i) + \epsilon \cdot \text{sign}(\nabla J(\phi, X(i), y(i)))$
  end for
  $X_\epsilon \leftarrow X_\epsilon \cup X_{\text{adv}}$
end for
return X_ϵ

The equation for FGSM attack is:

$$X_{\text{adv}} = \text{X} + \epsilon \cdot \text{sign}\left(\nabla J(f, \text{X}, y)\right) \tag{2}$$

where X_{adv} is the disturbed input, X is the actual input, ϵ is a constant (perturbation intensity), $\text{sign}(\cdot)$ computes the sign of the gradient, $J(f, X, y)$ is the gradient of J with regard to X, f is the neural network, and y is the actual output.

5.2.2. Basic Iterative Method (BIM)

(Kurakin, Goodfellow, & Bengio, 2017) proposed BIM which applies the FGSM attack multiple times. Since in each iteration, the attack forces the model to add noise or perturbation in the direction of the gradients with regard to the loss function, this attack mechanism is generally considered to be more powerful than FGSM. We have used the BIM method given by Algorithm 2 to attack the best-performing vanilla LSTM and Encoder-Decoder LSTM models selected earlier.

Algorithm 2 BIM Algorithm

Require: Mapping ϕ from X to y, perturbation magnitude ϵ, features or attributes X, label or target y, step size α
Ensure: X_ϵ: adversarial examples for different values of ϵ
$i \leftarrow 1$
$X_\epsilon \leftarrow \emptyset$
for $\epsilon \leftarrow$ start to end **do**
  while $i \leq \min(4 + \epsilon/\alpha, 1.25 \times \epsilon/\alpha)$ **do**
    for $j \leftarrow 1$ to m **do**
      $X_{\text{adv}}(j) \leftarrow X(j) + \alpha \cdot \text{sign}\left(\nabla J(\phi, X(j), y(j))\right)$
      $X_{\text{adv}}(j) \leftarrow \min\left(X(j) + \epsilon, \max\left(X(j) - \epsilon, X_{\text{adv}}(j)\right)\right)$
    end for
    $X \leftarrow X_{\text{adv}}$
    $i \leftarrow i + 1$
  end while
  $X_\epsilon \leftarrow X_\epsilon \cup X_{\text{adv}}$
end for
return X_ϵ

Its equation is given as follows:

$$X_{\text{adv}} = X + \alpha \cdot \text{sign}(\nabla J(f, X, y)) \tag{3}$$

$$X_{\text{adv}} = \min(X + \epsilon, \max(X - \epsilon, X_{\text{adv}})) \tag{4}$$

where X_{adv} is the disturbed input, X is the actual input, α is the step size, ϵ is a constant (perturbation intensity), $\text{sign}(\cdot)$ computes the sign of the gradient, $J(f, X, y)$ is the gradient of J with regard to X, f is the neural network, and y is the actual output.

5.3. Adversarial Defenses

We enumerate the two types of adversarial defenses performed, in this section.

5.3.1. Data Augmentation-based Adversarial Training (DAAT)

DAAT given by Algorithm 3 is a naïve process of using adversarial attacks to create adversarial examples and augmenting the dataset to incorporate these examples during the training of the deep learning models (Goodfellow et al., 2015).

Algorithm 3 DAAT Algorithm

Require: Mapping ϕ_{best} from X to y, features or attributes X, label or target y, perturbation magnitude ϵ, step size α (Present/Absent based on the attack algorithm chosen)
Ensure: Mapping ϕ from X_{aug} to y_{aug}
1: $X_\epsilon \leftarrow \text{attack}(\phi_{\text{best}}, \epsilon, X, y, \alpha)$
2: $X_{\text{aug}} \leftarrow X + X_\epsilon$
3: $y_{\text{aug}} \leftarrow \text{concat}(y, \text{len}(\epsilon) + 1 \text{ times})$
4: Find an optimal mapping ϕ_{daat} such that $\phi(X_{\text{aug}}) \approx y_{\text{aug}}$
5: Calculate $\text{rmse}(y_{\text{aug}}, \phi(X_{\text{aug}}))$

This mechanism is used after anticipating the types of perturbations the adversary can use during the attack. DAAT provides the deep learning models with prior information necessary to stay resilient to attacks during the training process. We used the adversarial attacks introduced earlier to augment the dataset with adversarial examples for different values of perturbation magnitude epsilon. Then we trained a robust classifier on the augmented dataset. This robust classifier is more resistant to adversarial attacks since it has been trained to predict the right values, given the perturbed values in its training set. The system architecture of DAAT is shown in Figure 10.

5.3.2. Layer-wise Perturbation-Based Adversarial Training (LPAT)

This defense technique is inspired by the LPAT algorithm proposed by (Zhang, Wang, He, Li, & Yu, 2018) to handle class imbalances in hard drive health prediction. This robust LSTM network is trained by going through two rounds of feed-forward and backpropagation in each iteration. The first round is similar to any neural network architecture where the outputs are computed in the forward propagation step, and the gradients are updated in the backpropagation step. In the second round, however, the gradients in each layer are perturbed using FGSM and BIM attacks before the forward and backward propagation steps are performed again. We have extended this algorithm to include a deterministic gradient update, where a predetermined value is always used as the magnitude of gradient perturbation, and a stochastic gradient update, where any random value between a specified range is used to perturb the gradients. The entire algorithm is given in Algorithm 4.

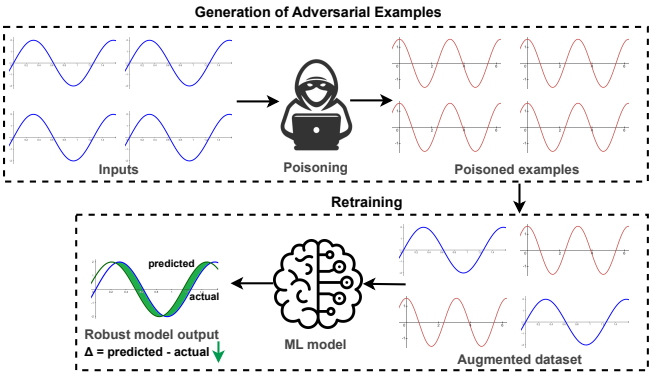


Figure 10. System Architecture of Data Augmentation-based Adversarial Training (DAAT).

Algorithm 4 LPAT Algorithm

Require: Features or attributes X, label or target y, perturbation magnitude ϵ
Ensure: Mapping ϕ from X to y
1: **for** each epoch in epochs **do**
2: Learn the mapping ϕ in the forward pass
3: Compute the gradients and update the parameters in the backward pass
4: **Perturb the gradients in each of the network layers:**
5: **if** dlpat **then**
6: $\text{grads} \leftarrow \text{grads} + \text{avg}(\epsilon) \cdot \text{sign}(\text{grads})$
7: **else if** slpat **then**
$\text{grads} \leftarrow \text{grads} + \text{random}(\epsilon, \text{start}, \text{end}) \cdot \text{sign}(\text{grads})$
8: **end if**
9: Perform feedforward process to learn the new mapping ϕ
10: Perform backpropagation again to update parameters
11: **end for**
12: Calculate $\text{rmse}(y, \phi(X))$

The system architecture of LPAT is shown in Figure 11.

6. RESULTS

In this section, we disclose the findings from carrying out the attacks and defenses on the final deep learning models selected in Section 3. The accompanying code has been made available on GitHub.[1].

[1]Accompanying code can be found at `https://github.com/micosyslab/mts-adv-attack-defense`.

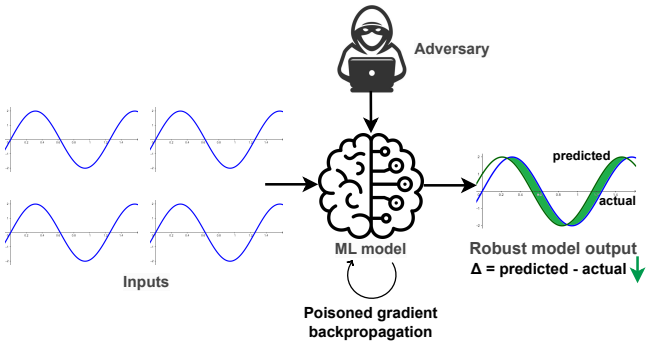


Figure 11. System Architecture of Layer-wise Perturbation-based Adversarial Training (LPAT).

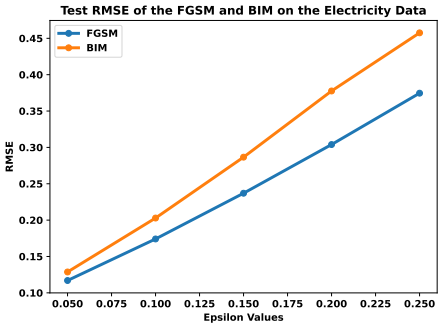


Figure 12. Comparison of test RMSE for FGSM and BIM with varying ϵ on the Electricity dataset.

6.1. Individual Household Power Consumption Dataset

6.1.1. Results of Adversarial Attacks

We contaminate the inputs to the vanilla LSTM model during the evaluation stage using the FGSM and BIM attacks mentioned in Section 5.2. For this experiment with the electricity dataset, we find that smaller orders of perturbation magnitude between 0.05 and 0.25, incremented in steps of 0.05 give test set RMSE values in the range of 0.117 and 0.3746 for FGSM, and between 0.1287 and 0.4575 for BIM.

α parameter in Equation 4 is always set to 0.01 for all the experiments conducted. The total iterations I is given as follows:

$$I = \min\left(4 + \frac{\epsilon}{\alpha}, 1.25 \cdot \frac{\epsilon}{\alpha}\right) \tag{5}$$

where ϵ is the degree of fluctuation and α is the step size.

The results of the attacks on the electricity dataset are tabulated in Table 3. It is seen that RMSE increases with increasing ϵ.

Table 3. RMSE Values on the Test Electricity Dataset after Attack

Attack Type	ϵ	Attack RMSE	% increase in error
FGSM	0.05	0.117	54.14
	0.1	0.1741	127.88
	0.15	0.237	210.21
	0.2	0.3039	297.79
	0.25	0.3746	390.31
BIM	0.05	0.1287	68.46
	0.1	0.2029	165.58
	0.15	0.2866	275.13
	0.2	0.3776	394.24
	0.25	0.4575	498.82

We see that BIM is a stronger attack than FGSM from Figure 12.

Figure 13 and Figure 14 show the estimations on the altered evaluation dataset after the FGSM and BIM attacks respectively. Figure 15 and Figure 16 show the changes in a subset of input data after the FGSM and BIM attacks respectively.

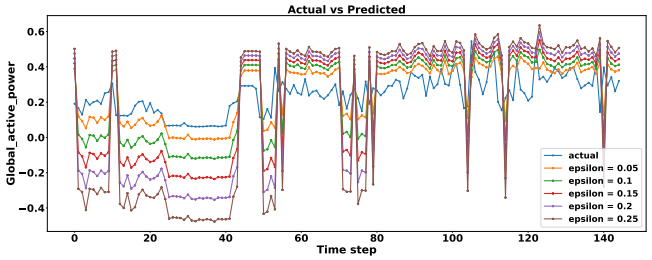


Figure 13. Predictions on the FGSM perturbed electricity test dataset for varying ϵ.

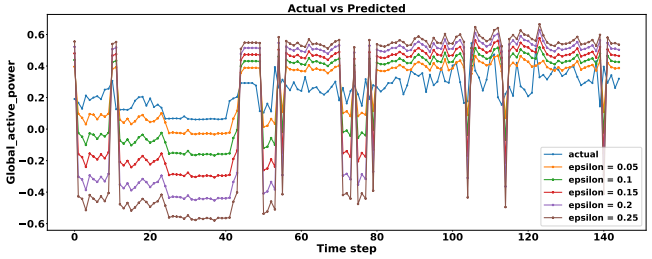


Figure 14. Predictions on the BIM perturbed electricity test dataset for varying ϵ.

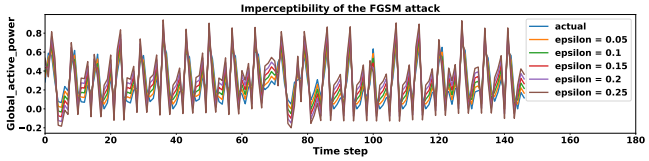


Figure 15. Imperceptibility of the FGSM attack on the electricity dataset.

It is obvious that the changes in the train set distribution for all ϵ values are almost imperceptible to the naked eye, but the predictions on the test set vary widely demonstrating the danger of these attacks.

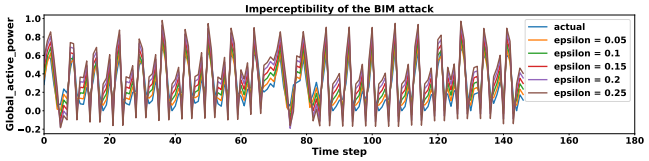


Figure 16. Imperceptibility of the BIM attack on the electricity dataset.

6.1.2. Results of Adversarial Defenses

We carry out two different types of adversarial defense strategies to harden the best-performing vanilla LSTM model on the electricity dataset. In DAAT, we augment the dataset with perturbed input attributes created by performing adversarial perturbations on the original feature samples. We train the vanilla LSTM network to learn the fluctuations in the features and predict the correct target. The different RMSE values obtained after performing DAAT on the electricity dataset are tabulated in Table 4. Although BIM is a stronger attack than FGSM, DAAT on the electricity dataset is slightly more resilient to BIM attack. The observed phenomenon can be ascribed to the broader range of variations in the test features perturbed by BIM enhancing the model's capability to identify noisy outliers within the underlying distribution more effectively.

Table 4. Results after Performing DAAT on the Electricity Dataset

Type of Attack	RMSE Metrics	Values
FGSM	Train	0.0994
	Validation	0.1271
	Test (clean data)	0.0761
	Test (poisoned data)	0.0847
BIM	Train	0.0986
	Validation	0.1279
	Test (clean data)	0.0802
	Test (poisoned data)	0.0949

In the second method, we harden the vanilla LSTM model by perturbing the gradients of the model during backpropagation, allowing it to account for the outliers better. We introduced two types of gradient update - deterministic and stochastic. In the deterministic update, the mean of ϵ (in this case 0.15) is used, whereas in the stochastic process, we choose from a range of ϵ (in this case between 0.05 and 0.25). The various RMSE metrics obtained after using LPAT on the electricity dataset are tabulated in Table 5.

The RMSE values after the FGSM and BIM attacks and defenses are shown in Figure 17 and Figure 18 respectively. Figure 19 a line chart representing the percentage decrease in error on the electricity dataset for different ϵ after performing the defenses showing that BIM DAAT followed by FGSM DAAT gives the maximum reduction in error during training. This is followed by BIM-based DLPAT and SLPAT techniques and finally FGSM-based DLPAT and SLPAT.

Table 5. Results after Performing LPAT on the Electricity Dataset

Attack Type	Training Type	RMSE Metrics	Value
FGSM	Deterministic	Train	0.1413
		Test	0.0974
	Stochastic	Train	0.1424
		Test	0.0979
BIM	Deterministic	Train	0.1304
		Test	0.0985
	Stochastic	Train	0.1389
		Test	0.101

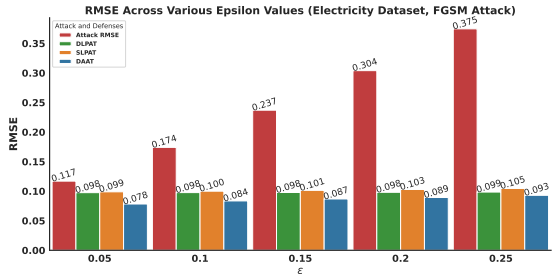


Figure 17. RMSE of FGSM attack and defenses for electricity test set perturbed by varying ϵ.

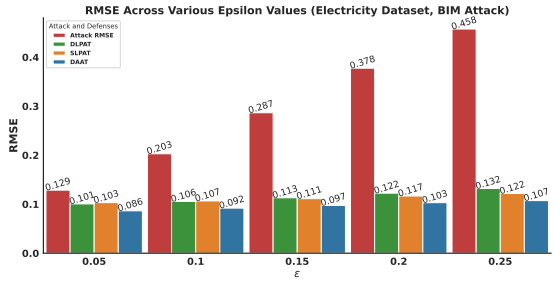


Figure 18. RMSE of BIM attack and defenses for electricity test set perturbed by varying ϵ.

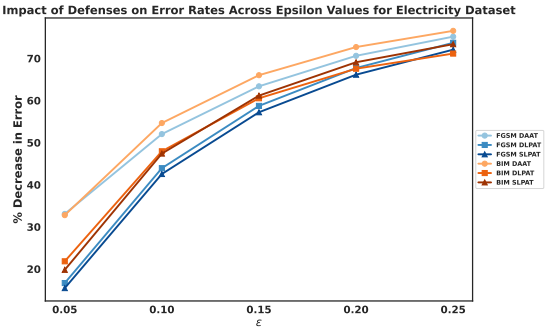


Figure 19. Percentage Reduction in Error Values after Adversarial Defense on the Electricity Dataset

6.1.3. Comparative Analysis of the Results

Mode et. al (Mode & Hoque, 2020) have used an LSTM network to perform the adversarial attacks on the Electricity dataset down-sampled to the per-hour values. They have

reported a 12.3% and 22.9% increase in RMSE values after performing the FGSM and BIM attacks respectively for an ϵ value of 0.2. For the same ϵ value of 0.2, but by down-sampling the dataset to include the per-day values, we have proved an increase in RMSE values of 297.79%, and 394.24% for the FGSM and BIM attacks respectively, proving that our attack algorithms severely compromise the model for the same input distribution. While the authors explore ideas for a defense algorithm, this work is limited by the lack of implementation of actual defense techniques. Our research bridges this gap by exploring two different defense techniques on the Electricity dataset.

Govindarajulu et. al (Govindarajulu et al., 2023) have explored different attack strategies based on the direction, amplitude and temporal components of the attack on the Electricity dataset. The only direct comparison we can make is that their RMSE value on the LSTM model at a 60% data split for the training dataset down-sampled to per-hour values is 0.085 whereas our model at the 60-20-20 split, yields a better RMSE value of 0.0746 for the per-day dataset. This work is also limited by the lack of implementation of defense techniques.

6.2. Backblaze Hard Disk Drive Dataset

6.2.1. Results of Adversarial Attacks

We poison the test inputs to the best-performing model identified in Section 4.2 using the FGSM and BIM attacks. We used epsilon ranging from 3 to 11 incremented in steps of 2 such as 3, 5, 7, 9, and 11 since the Encoder-Decoder LSTM model is inherently robust to lower ϵ resulting in only a 0.92% to 3.87% increase in error rate for FGSM and a 1.12% to 4.7% increase in error rate for BIM for ϵ between 0.05 and 0.2.

α in Equation 4 is set to 0.01 for all the experiments. The number of iterations I is given by Equation 5.

The results for the perturbations performed on the HDD dataset tabulated in Table 6 show that RMSE increases with increasing ϵ.

Table 6. RMSE Values on the Test Hard Disk Drive (HDD) Dataset after Attack

Attack Type	ϵ	Attack RMSE	% increase in error
FGSM	3	0.124	73.43
	5	0.1683	135.39
	7	0.2154	201.26
	9	0.2605	264.34
	11	0.298	316.78
BIM	3	0.1344	87.97
	5	0.1904	166.29
	7	0.2505	250.35
	9	0.3075	330.07
	11	0.3609	404.76

We see that BIM is a stronger attack than FGSM in Figure 20.

Figure 21 and Figure 22 show the forecasts on the disrupted

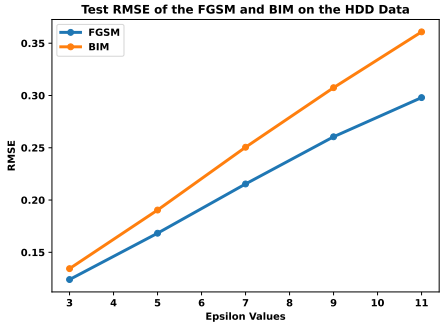


Figure 20. Comparison of test RMSE for FGSM and BIM with varying ϵ on the HDD dataset.

test set corresponding to the FGSM and BIM attacks respectively. Figure 23 and Figure 24 show the changes in a subset of input data after performing the FGSM and BIM attacks respectively thereby substantiating the fact that predictions on the attacked test set are worse while the inputs look the same to the naked eye.

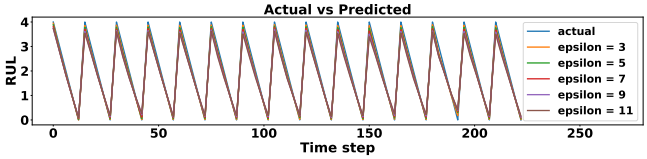


Figure 21. Predictions on the FGSM perturbed HDD test dataset for varying ϵ.

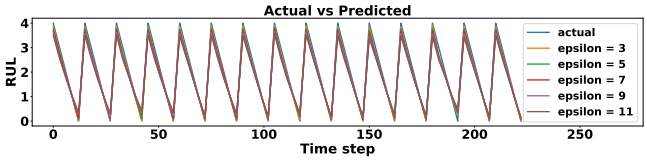


Figure 22. Predictions on the BIM perturbed HDD test dataset for varying ϵ.

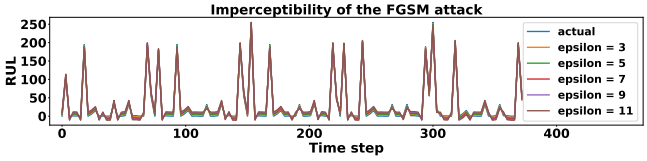


Figure 23. Imperceptibility of the FGSM attack on the HDD dataset.

6.2.2. Results of Adversarial Defenses

We performed two types of adversarial defenses to make the best-performing Encoder-Decoder LSTM more resilient to

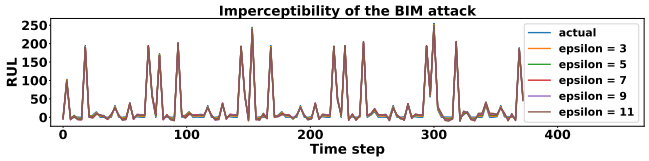


Figure 24. Imperceptibility of the BIM attack on the HDD dataset.

adversarial attacks seen in Section 6.2.1. In the first method, we augmented the input feature samples with perturbed data in the course of model training to permit the model to learn the true values given perturbed features. The results of the different RMSE values obtained while performing DAAT on the HDD dataset are tabulated in Table 7. Although BIM is stronger than FGSM, DAAT on the HDD dataset is more resilient to the BIM attack due to the wide variations in feature values enabling the model to detect the noisy outliers in the underlying distribution better, as seen earlier.

Table 7. Results after Performing DAAT on the HDD Dataset

Type of Attack	RMSE Metrics	Values
FGSM	Train	0.0334
	Validation	0.0858
	Test (clean data)	0.0206
	Test (poisoned data)	0.0313
BIM	Train	0.0285
	Validation	0.0425
	Test (clean data)	0.0129
	Test (poisoned data)	0.0130

In the next method, we perturb the gradients of the model during training hoping that the model learns a more robust distribution. We modified LPAT to include two different types of gradient perturbation during training - deterministic where ϵ is 7 i.e. the average of all expected ϵ values, and stochastic where ϵ takes a value between 3 and 11 i.e. the expected range. The RMSE metrics after applying LPAT to the HDD dataset are tabulated in Table 8.

Table 8. Results after Performing LPAT on the HDD Dataset

Attack Type	Training Type	RMSE Metrics	Value
FGSM	Deterministic	Train	0.23
		Test	0.1422
	Stochastic	Train	0.5457
		Test	0.1457
BIM	Deterministic	Train	0.3538
		Test	0.0926
	Stochastic	Train	0.2848
		Test	0.0455

Figure 25 and Figure 26 compare the RMSE values after the FGSM and BIM attacks and defenses respectively. Figure 27 shows the reduction in error after defense for different ϵ and different attacks.

It is seen that DLPAT and SLPAT for FGSM when $\epsilon = 3$ have RMSE values higher than the attack RMSE. Although LPAT leads to more robust models resistant to attacks in most cases, in this case, we see that an average ϵ value of 7 did not help in learning a robust model when poisoned with a smaller ϵ value of 3. The same can be said for the deterministic gradient update where the random values picked during SLPAT do not effectively contribute to learning the underlying distribution with allowance for noise.

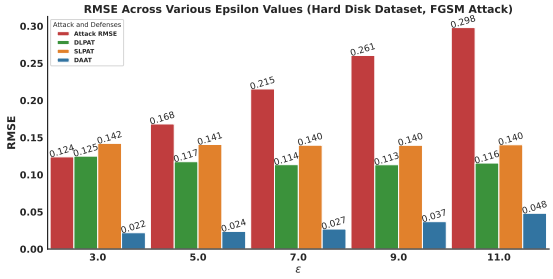


Figure 25. RMSE values of FGSM attack and defenses for HDD test set perturbed by varying ϵ.

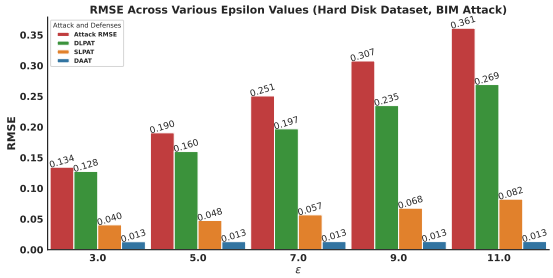


Figure 26. RMSE values of BIM attack and defenses for HDD test set perturbed by varying ϵ.

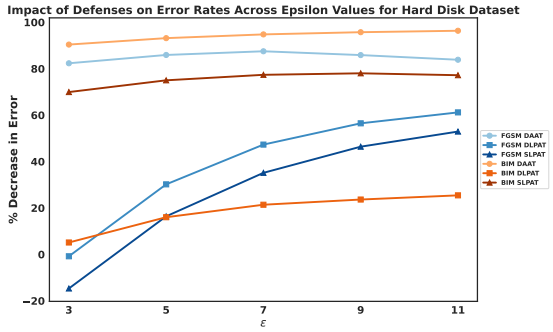


Figure 27. Percentage Reduction in Error Values after Adversarial Defense on the HDD Dataset

6.2.3. Comparative Analysis of the Results

The use of adversarial attacks and defense techniques on the hard disk dataset represents the first exploration of these techniques on this dataset. While we modified the LPAT algorithm initially proposed to handle class imbalances by Zhang

et. al (Zhang et al., 2018) on the Backblaze dataset, there is no other comparable literature that we can compare these results to, to the best of our knowledge.

7. LIMITATIONS AND FUTURE WORK

Research in adversarial attacks and defenses for multivariate time-series forecasting is nascent, with many areas unexplored. This study focuses on white-box attacks, which necessitate knowledge of model parameters and inputs. In the future, we will explore black-box attacks to construct an attack model using the estimates of the target model alone. We aim to develop a model that can discern the adversary's ϵ range based on prediction deviations and identify the maximum perturbation beyond which defense fails. Adversarial training with BIM as the perturbation method yields superior results, suggesting further exploration of the impact of ϵ on training, as BIM is a stronger attack than FGSM. We find that stochastic gradient update is highly dependent on the random values of ϵ picked during each iteration of backpropagation, warranting research into the selection of ϵ during the training process Investigating black-box transferability and integrating DAAT and LPAT into a hybrid approach could enhance model robustness against adversarial attacks.

8. CONCLUSION

Our study investigates the susceptibility of deep learning models in multivariate time-series forecasting to adversarial attacks and evaluates defense mechanisms. We show that models like vanilla LSTM and Encoder-Decoder LSTM when tested on the Individual Household Power Consumption and Backblaze Hard Disk Drive datasets, undergo significant performance degradation under adversarial perturbations like FGSM and BIM. The average error rate increases by 248.17% and 223% on the electricity and HDD datasets respectively, highlighting the impact of these attacks.

We also visualize the subtle changes to the input distribution post-attack, which are not easily detectable. To counter these vulnerabilities, we implement two robust defenses: Data Augmentation-based Adversarial Training (DAAT) and Layer-wise Perturbation-based Adversarial Training (LPAT). DAAT, particularly with BIM for augmentation, greatly enhances model resilience, reducing errors by up to 72.41% and 94.81% for the electricity and HDD datasets respectively.

LPAT also shows effectiveness, with performance varying based on perturbation magnitude. These results emphasize the need for adversarial defense strategies in deep learning models for critical applications in smart and connected infrastructures, enhancing model reliability and ensuring secure, dependable predictions.

ACKNOWLEDGMENT

The research reported in this publication was supported by the Division of Research and Innovation at San José State University under Award Number 23-UGA-08-044. The content is solely the responsibility of the author(s) and does not necessarily represent the official views of San José State University.

REFERENCES

Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, *6*, 14410–14430. doi: 10.1109/ACCESS.2018.2807385

Akhtar, N., Mian, A., Kardan, N., & Shah, M. (2021). Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, *9*, 155161–155196. doi: 10.1109/ACCESS.2021.3127960

Alden, R. E., Gong, H., Ababei, C., & Ionel, D. M. (2020). Lstm forecasts for smart home electricity usage. In *2020 9th international conference on renewable energy research and application (icrera)* (pp. 434–438).

Al-Dulaimi, A., Zabihi, S., Asif, A., & Mohammadi, A. (2019). A multimodal and hybrid deep neural network model for remaining useful life estimation. *Computers in Industry*, *108*, 186–196. doi: https://doi.org/10.1016/j.compind.2019.02.004

Backblaze. (2023). *Backblaze hard drive test data.* Retrieved from `https://www.backblaze.com/b2/hard-drive-test-data.html`

Bohan, H., & Yun, B. (2019). Traffic flow prediction based on brnn. In *2019 IEEE 9th international conference on electronics information and emergency communication (iceiec)* (pp. 320–323). doi: 10.1109/ICEIEC.2019.8784513

Chen, X. (2024). A novel transformer-based dl model enhanced by position-sensitive attention and gated hierarchical lstm for aero-engine rul prediction. *Scientific Reports*, *14*(1), 10061.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using rnn encoder-decoder for statistical machine translation.*

Croce, F., & Hein, M. (2019). Sparse and imperceivable adversarial attacks. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 4724–4732).

da Silva, D. G., Geller, M. T. B., dos Santos Moura, M. S., & de Moura Meneses, A. A. (2022). Performance evaluation of lstm neural networks for consumption prediction. *e-Prime-Advances in Electrical*

Engineering, Electronics, and Energy, *2*, 100030.

Deutsch, J., & He, D. (2018). Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(1), 11–20. doi: 10.1109/TSMC.2017.2697842

Divina, F., García Torres, M., Goméz Vela, F. A., & Vázquez Noguera, J. L. (2019). A comparative study of time series forecasting methods for short term electric energy consumption prediction in smart buildings. *Energies*, *12*(10). doi: 10.3390/en12101934

Dong, Y., Deng, Z., Pang, T., Zhu, J., & Su, H. (2020). Adversarial distributional training for robust deep learning. *Advances in Neural Information Processing Systems*, *33*, 8270–8283.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2018). Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9185–9193).

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., & Madry, A. (2018). A rotation and a translation suffice: Fooling cnns with simple transformations.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., ... Song, D. (2018). *Robust physical-world attacks on deep learning models.*

Faloutsos, C., Gasthaus, J., Januschowski, T., & Wang, Y. (2018). Forecasting big time series: old and new. *Proceedings of the VLDB Endowment*, *11*(12), 2102–2105.

Finlay, C., Pooladian, A.-A., & Oberman, A. (2019). The logbarrier adversarial attack: making effective use of decision boundary information. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4862–4870).

Fursov, I., Morozov, M., Kaploukhaya, N., Kovtun, E., Rivera-Castro, R., Gusev, G., ... Burnaev, E. (2021). Adversarial attacks on deep models for financial transaction records. In *Proceedings of the 27th acm sigkdd conference on knowledge discovery & data mining* (pp. 2868–2878).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples.*

Govindarajulu, Y., Amballa, A., Kulkarni, P., & Parmar, M. (2023). Targeted attacks on timeseries forecasting. *arXiv:2301.11544*.

Harford, S., Karim, F., & Darabi, H. (2020). Adversarial attacks on multivariate time series. *arXiv:2004.00410*.

Hebrail, G., & Berard, A. (2012). *Individual household electric power consumption.* UCI Machine Learning Repository. doi: doi.org/10.24432/C58K54

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Huang, Q., Katsman, I., He, H., Gu, Z., Belongie, S., & Lim, S.-N. (2019). Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4733–4742).

Ibrahim, N. M., Megahed, A. I., & Abbasy, N. H. (2021). Short-term individual household load forecasting framework using lstm deep learning approach. In *2021 5th international symposium on multidisciplinary studies and innovative technologies (ismsit)* (pp. 257–262).

Jang, Y., Zhao, T., Hong, S., & Lee, H. (2019). Adversarial defense via learning to generate diverse attacks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2740–2749).

Jeddi, A., Shafiee, M. J., Karg, M., Scharfenberger, C., & Wong, A. (2020). Learn2perturb: an end-to-end feature perturbation learning to improve adversarial robustness. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1241–1250).

Karevan, Z., & Suykens, J. A. (2020). Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, *125*, 1–9. doi: 10.1016/j.neunet.2019.12.030

Karim, F., Majumdar, S., & Darabi, H. (2020). Adversarial attacks on time series. *IEEE transactions on pattern analysis and machine intelligence*, *43*(10), 3309–3320.

Kaushik, S., Choudhury, A., Sheron, P. K., Dasgupta, N., Natarajan, S., Pickett, L. A., & Dutt, V. (2020). Ai in healthcare: Time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in Big Data*, *3*. doi: 10.3389/fdata.2020.00004

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization.* arXiv. doi: 10.48550/arXiv.1412.6980

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). *Adversarial examples in the physical world.*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lin, C.-Y., Hsieh, Y.-M., Cheng, F.-T., Huang, H.-C., & Adnan, M. (2019). Time series prediction algorithm for intelligent predictive maintenance. *IEEE Robotics and Automation Letters*, *4*(3), 2807–2814. doi: 10.1109/LRA.2019.2918684

Lin, J., Dang, L., Rahouti, M., & Xiong, K. (2021). Ml attack models: Adversarial attacks and data poisoning attacks. *arXiv preprint arXiv:2112.02797*.

Liu, L., Park, Y., Hoang, T. N., Hasson, H., & Huan, J. (2022). Towards robust multivariate time-series forecasting: adversarial attacks and defense

mechanisms. *arXiv preprint arXiv:2207.09572*.

Madaan, D., Shin, J., & Hwang, S. J. (2020). Adversarial neural pruning with latent vulnerability suppression. In *International conference on machine learning* (pp. 6575–6585).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*.

Melis, M., Demontis, A., Biggio, B., Brown, G., Fumera, G., & Roli, F. (2017). *Is deep learning safe for robot vision? adversarial examples against the icub humanoid.*

Melis, M., et al. (2021). Explaining vulnerability of machine learning to adversarial attacks.

Miller, D. J., Xiang, Z., & Kesidis, G. (2020). Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, *108*(3), 402–433.

Mishra, K., Basu, S., & Maulik, U. (2019). Danse: a dilated causal convolutional network based model for load forecasting. In *Pattern recognition and machine intelligence: 8th international conference, premi 2019, tezpur, india, december 17-20, 2019, proceedings, part i* (pp. 234–241).

Mode, G. R., & Hoque, K. A. (2020). Adversarial examples in deep learning for multivariate time series regression. In *2020 ieee applied imagery pattern recognition workshop (aipr)* (pp. 1–10).

Mohapatra, R., Coursey, A., & Sengupta, S. (2023). Large-scale end-of-life prediction of hard disks in distributed datacenters. In *2023 ieee international conference on smart computing (smartcomp)* (pp. 261–266). doi: 10.1109/SMARTCOMP58114.2023.00069

Mohapatra, R., & Sengupta, S. (2023). Tfbest: Dual-aspect transformer with learnable positional encoding for failure prediction. *arXiv preprint arXiv:2309.02641*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1765–1773).

Mustafa, A., Khan, S., Hayat, M., Goecke, R., Shen, J., & Shao, L. (2019). Adversarial defense by restricting the hidden space of deep neural networks. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 3385–3394).

Muzaffar, S., & Afshari, A. (2019). Short-term load forecasts using lstm networks. *Energy Procedia*, *158*, 2922–2927.

Oh, S. J., Schiele, B., & Fritz, M. (2019). Towards reverse-engineering black-box neural networks. *Explainable AI: interpreting, explaining and visualizing deep learning*, 121–144.

Orimoloye, L. O., Sung, M.-C., Ma, T., & Johnson, J. E. (2020). Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices. *Expert Systems with Applications*, *139*, 112828.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). *Practical black-box attacks against machine learning.*

Rathore, P., Basak, A., Nistala, S. H., & Runkana, V. (2020). Untargeted, targeted and universal adversarial attacks and defenses on time series. In *2020 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Sankaranarayanan, S., Jain, A., Chellappa, R., & Lim, S. N. (2018). Regularizing deep networks using efficient layerwise adversarial training. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181. doi: 10.1016/j.asoc.2020.106181

Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2018). A comparison of arima and lstm in forecasting time series. In *2018 17th ieee international conference on machine learning and applications (icmla)* (pp. 1394–1401). doi: 10.1109/ICMLA.2018.00227

Stergiou, C., & Psannis, K. E. (2017). Recent advances delivered by mobile cloud computing and internet of things for big data applications: a survey. *International Journal of Network Management*, *27*(3), e1930.

Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, *23*(5), 828–841.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). *Intriguing properties of neural networks.*

Tariq, M. I., Memon, N. A., Ahmed, S., Tayyaba, S., Mushtaq, M. T., Mian, N. A., … Ashraf, M. W. (2020). A review of deep learning security and privacy defensive techniques. *Mobile Information Systems*, *2020*, 1–18.

Tsingenopoulos, I., Preuveneers, D., & Joosen, W. (2019). Autoattacker: A reinforcement learning approach for black-box adversarial attacks. In *2019 ieee european symposium on security and privacy workshops (euros&pw)* (pp. 229–237).

Wang, J., & Zhang, H. (2019). Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 6629–6638).

Wu, W., Liao, W., Miao, J., & Du, G. (2019). Using gated recurrent unit network to forecast short-term load considering impact of electricity price. *Energy Procedia*, *158*, 3369–3374.

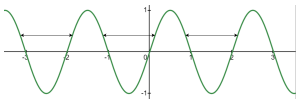
Yan, H., & Ouyang, H. (2018). Financial time series prediction based on deep learning. *Wireless Personal Communications*, *102*, 683–700.

Zhang, J., Wang, J., He, L., Li, Z., & Yu, P. S. (2018). Layerwise perturbation-based adversarial training for hard drive health degree prediction. In *2018 ieee international conference on data mining (icdm)* (pp. 1428–1433). doi: 10.1109/ICDM.2018.00197

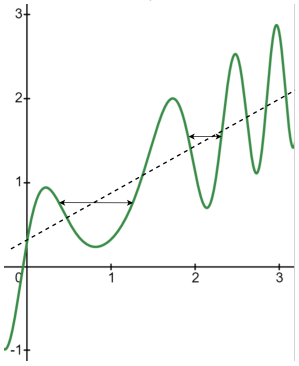
APPENDIX

A. TIME-SERIES DATA CHARACTERISTICS

Time-series data are characterized into two types - stationary time-series where the mean or variance is a constant given by $\mu = c$ or $\sigma = c$ or non-stationary time-series in which the mean or variance varies with time given by $\mu = f(t)$ or $\sigma = f(t)$. Figure 28 shows a stationary time-series with constant mean and variance, and a non-stationary time-series where the mean increases with time as shown by the dotted line, and the variance representing the distance between consecutive peaks decreases with time.



(a) Stationary time-series



(b) Non-stationary time-series

Figure 28. Characteristics of time-series data

Time-series data can be represented as a 2D array as:

$$\begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{t1} & v_{t2} & \dots & v_{tn} \end{bmatrix}$$

where *n* is the total readings recorded at any particular period, and *t* is the total readings recorded over *t* time steps. There are two types of time-series. The first is univariate time-series in which the future values of the series are dependent only on its past values. For example, in univariate time-series, the value of v_{tn} depends only on its past values such as $\left[v_{1n}, v_{2n}, \dots, v_{(t-1)n}\right]$. The second is a multivariate time-series in which the future values depend on a combination of the past values and predictors such as exogenous variables other than the series. For example, in multivariate time-series, the value of v_{tn} depends on both its past values and other parameters captured by sensing devices such as $[v_{11}, v_{12}, \dots, v_{1n}, v_{21}, v_{22}, \dots, v_{2n}, \dots, v_{t1}, v_{t2}, \dots, v_{(t-1)n}]$.

B. SUPPLEMENTAL BACKGROUND

(Croce & Hein, 2019) propose a new black-box attack to apply sparse perturbations to image pixels leading to unnoticeable changes in the resultant image. (Dong et al., 2018) iteratively attacked the samples by introducing a momentum term that prevents the gradients from being stuck in a local maxima resulting in a much more powerful attack. A paper published by (Engstrom, Tran, Tsipras, Schmidt, & Madry, 2018), outlines the vulnerability of CNNs (LeCun et al., 1998) to benign rotations and transformations. (Finlay, Pooladian, & Oberman, 2019) use a log barrier-based optimization technique to solve the constrained optimization problem that aims to minimize the perturbation magnitude in adversarial attacks. (Huang et al., 2019) show that an intermediate-level attack ensures high transferability of adversarial attacks between architectures. (Moosavi-Dezfooli, Fawzi, Fawzi, & Frossard, 2017) identify the presence of global perturbations which are independent of the images and depend on the geometric modeling of the decision edges of deep learning algorithms. (Su, Vargas, & Sakurai, 2019) proved that CNNs are susceptible to attacks of lower dimensions by modifying only one pixel based on differential evolution to perturb images.

(Sankaranarayanan, Jain, Chellappa, & Lim, 2018) suggest efficient layer-wise training to prevent overfitting in deep networks. (Mustafa et al., 2019) propose a convex polytope-based separation of features during learning, such that independent variables of various targets are maximally separated from one another. (Wang & Zhang, 2019) introduce a bilateral adversarial training framework by perturbing the labels and the features during the training process. (Madry, Makelov, Schmidt, Tsipras, & Vladu, 2017) perform adversarial training through robust optimization, exploring the universal transferability during training. (Jeddi, Shafiee, Karg, Scharfenberger, & Wong, 2020) introduce a framework that introduces perturbations during the training and inference and efficiently learns to detect noise in the input. (Dong, Deng, Pang, Zhu, & Su, 2020) outline a process called adversarial distribution training in which the internal maximization function seeks to learn the worst distribution possible, and the external minimization function seeks to minimize the loss over the distribution. (Madaan, Shin, & Hwang, 2020) put forth a novel loss function called vulnerability suppression loss that aims to minimize the latent space feature distortion. (Jang, Zhao, Hong, & Lee, 2019) develop an iterative stochastic generator to generate diverse adversarial examples capable of exposing the vulnerabilities in the target model. (Liu, Park, Hoang, Hasson, & Huan, 2022) propose an attack scheme that introduces imperceptible perturbations to create poisoned examples and training mechanisms based on randomized smoothing to enhance model robustness.

C. INDIVIDUAL HOUSEHOLD POWER CONSUMPTION DATASET

C.1. Data Preprocessing

We conducted experiments using an LSTM which was designed to predict the *global_active_power*. To facilitate this, we resampled the dataset daily, incorporating the mean of the per-minute values. Figure 29 illustrates the distribution of the mean and sum of the per-minute values when the dataset is resampled daily. It is evident that whether we aggregate over the mean or sum while resampling over the day, the distribution remains consistent. Therefore, the pick of the clustering technique does not significantly impact the model estimates.

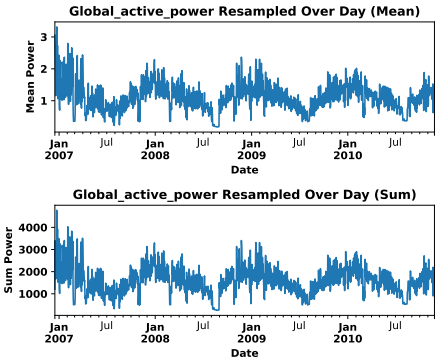


Figure 29. Global active power resampled per day for mean and sum of minutes.

C.2. Architecture of vanilla LSTM

The vanilla LSTM unit shown in Figure 30 consists of:

1) Forget Gate: The forget gate determines what piece of information to persist and what to delete. The input at a particular time step, i(t), and information from the preceding hidden state, s(t-1) act as inputs for the sigmoid activation, producing an output of 0 (forget), or 1 (remember). Its equation is:

$$f_g(t) = \sigma(i(t)X_f + s(t-1)Z_f) \tag{6}$$

2) Input Gate: This gate decides the relevant information to be passed further from the present. The input vector and the previous hidden layer's output are multiplied element by element, after passing through a sigmoid and a tanh function:

$$j(t) = \sigma(i(t)X_j + s(t-1)Z_{ig}) \tag{7}$$

$$k(t) = \tanh(i(t)X_k + s(t-1)Z_k) \tag{8}$$

$$i_g(t) = j(t) \cdot k(t) \tag{9}$$

*3) Cell State:*It reserves relevant long-term memory and performs element-by-element multiplication of the ouput of the forget gate and previous cell state to preserve only the relevant state of the network, and then performs element-by-element addition with the output of the input gate, given by:

$$C_s(t) = \sigma(f_g(t) \cdot C_s(t-1) + i_g(t)) \tag{10}$$

4) Output Gate: This gate decides the resultant value at any period. The input vector, i(t), and the byproduct from the preceding hidden layer, s(t-1) are put into a sigmoid activation to calculate $o_g(t)$, which is then multiplied with the tanh of the new cell state $C_s(t)$, to pass on as intake of the next time step along with the cell state, $C_s(t)$.

$$o_g(t) = \sigma(i(t)X_{og} + s(t-1)Z_{og}) \tag{11}$$

$$s(t) = \tanh(C_s(t)) \cdot o_g(t) \tag{12}$$

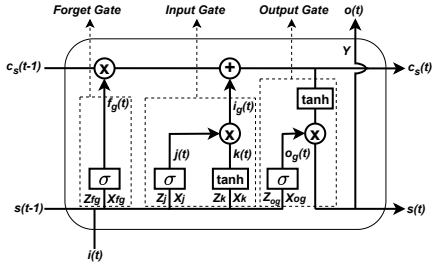


Figure 30. Vanilla LSTM unit

C.3. Training Process

Loss curve of the train and validation sets for the LSTM without using cross-validation or look back is shown in Figure 31.

Figure 32 shows the correlation between features for the data resampled over days. Since each feature is highly correlated with itself, the diagonal values of the correlation matrix are 1.

We perform feature selection based on the correlation matrix to check if it improves results. We identified the top features as *global active power, global intensity, sub-metering 1, and sub-metering 3*. Training the LSTM using these features and without using cross-validation or look-back gives us a train RMSE of 0.1024 and a test RMSE of 0.0783 compared to the test RMSE of 0.0807 without using cross-validation thereby improving model efficacy by 2.97%.

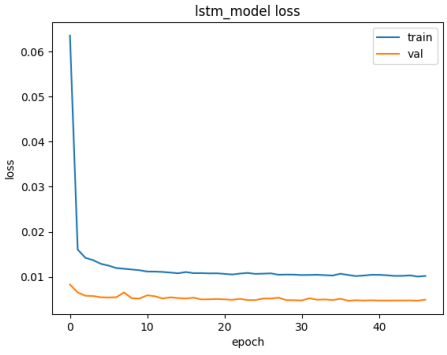


Figure 31. Loss function of the vanilla LSTM network.

C.4. Adversarial Defense Results

The overall percentage decrease in error on the electricity dataset after performing the different fortifications is tabulated in Table 9 and is illustrated in Figure 19.

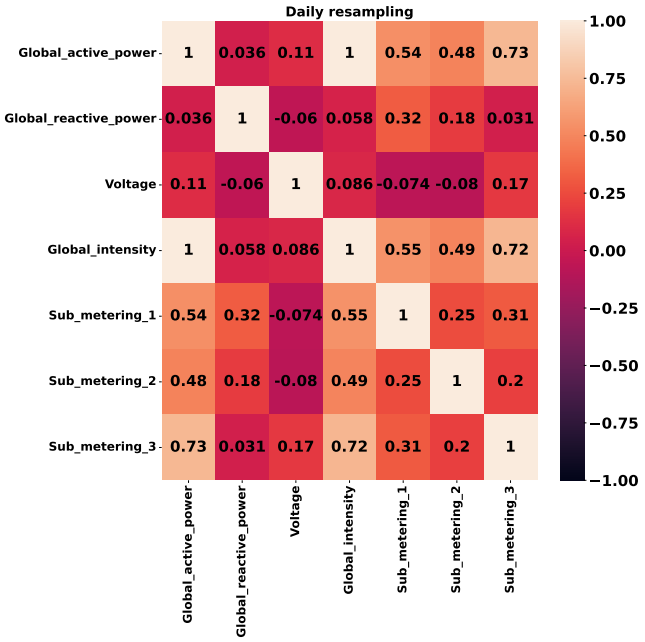


Figure 32. Correlation matrix of values resampled per day.

Table 9. Percentage Reduction in Error Values after Adversarial Training on the Electricity Dataset

Attack Type	Training Type	ϵ	% decrease in error
FGSM	DAAT	0.05	33.08
		0.1	52.04
		0.15	63.38
		0.2	70.62
		0.25	75.17
	DLPAT	0.05	16.67
		0.1	43.94
		0.15	58.73
		0.2	67.72
		0.25	73.71
	SLPAT	0.05	15.47
		0.1	42.56
		0.15	57.22
		0.2	66.14
		0.25	72.08
BIM	DAAT	0.05	32.79
		0.1	54.66
		0.15	66.02
		0.2	72.7
		0.25	76.55
	DLPAT	0.05	21.83
		0.1	47.95
		0.15	60.54
		0.2	67.56
		0.25	71.15
	SLPAT	0.05	19.81
		0.1	47.41
		0.15	61.17
		0.2	69.09
		0.25	73.38

D. BACKBLAZE HARD DISK DRIVE DATASET

D.1. Architecture of Encoder-Decoder LSTM

The components of the Encoder-Decoder model used on this dataset are explained below and illustrated in Figure 33.

1) Encoder: The encoder is composed of several RNNs, LSTMs, or GRUs stacked together accepting an input and propagating that information forward to the next units. The hidden state a(t) is computed from the input array, i(t), and the byproduct of the previous layer, a(t-1) based on the mapping of the chosen unit, if it is RNN, LSTM, or GRU. The final a(t) is composed of all the encoded information from the previous states and hidden layers. Its equation is given by:

$$a(t) = f(Za(t-1) + Xi(t)) \tag{13}$$

2) Context Vector: The context vector represents the last hidden state output of the encoder and the first hidden state input to the decoder. It is an encoded latent space representation of the inputs and allows the decoder to forecast accurately.

3) Decoder: The decoder consists of similar stacked recurrent RNN, LSTM, or GRU units, receiving a hidden state from the preceding unit, and calculates its hidden state and the output.

$$b(t) = f(Za(t-1)) \tag{14}$$

Softmax activation is applied on the hidden state, b(t), and the corresponding weight to output a probability vector as:

$$\hat{o}(t) = \text{softmax}(Yb(t)) \tag{15}$$

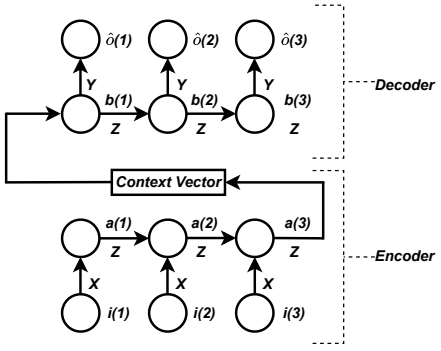


Figure 33. Architecture of Encoder Decoder LSTM.

D.2. Training Process

The loss function of the 5-day look back Encoder-Decoder model without cross-validation is shown in Figure 34.

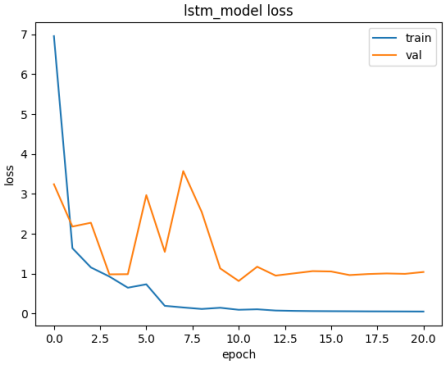


Figure 34. Loss function of Encoder Decoder LSTM.

We perform feature selection by training the 25-day look-back Encoder-Decoder LSTM with the top 10 features identified by an eXtreme Gradient Boosting (XGB) classifier as reported by (Mohapatra et al., 2023). We found the train RMSE to be 12.1174 and the test RMSE to be 16.5570 which is much higher than those reported in Table 2.

D.3. Adversarial Defense Results

Table 10 shows the reduction in error after performing adversarial defense for different perturbation values and different adversarial attacks. and is visualized in Figure 27.

Table 10. Percentage Reduction in Error Values after Adversarial Training on the HDD Dataset

Attack Type	Training Type	ϵ	% decrease in error
FGSM	DAAT	3	82.34
		5	85.92
		7	87.51
		9	85.87
		11	83.86
	DLPAT	3	-0.81
		5	30.18
		7	47.31
		9	56.47
		11	61.18
	SLPAT	3	-14.68
		5	16.4
		7	35.14
		9	46.41
		11	52.92
BIM	DAAT	3	90.4
		5	93.17
		7	94.78
		9	95.73
		11	96.35
	DLPAT	3	5.13
		5	16.02
		7	21.4
		9	23.64
		11	25.44
	SLPAT	3	69.94
		5	75
		7	77.37
		9	78.02
		11	77.2