STEWART: STacking Ensemble for White-Box AdversaRial Attacks Towards More Resilient Data-driven Predictive Maintenance

Onat Gungor[a,*], Tajana Rosing[a], Baris Aksanli[b]

[a]*Electrical and Computer Engineering Department, University of California, San Diego*
[b]*Electrical and Computer Engineering Department, San Diego State University*

Abstract

Industrial Internet of Things (I-IoT) is a network of devices that focus on monitoring industrial assets and continuously collecting data. This data can be utilized by Machine Learning (ML) methods to perform Predictive Maintenance (PDM) which identifies an optimal maintenance schedule for the industrial assets. The computational systems in the I-IoT are usually not designed with security in mind. Their limited computational power creates security vulnerabilities that attackers can exploit to prevent asset availability, sabotage communication, and corrupt system data. In this work, we first demonstrate that cyber-attacks can impact the performance of ML-based PDM methods significantly, leading up to 120× prediction performance loss. Next, we develop a stacking ensemble learning-based framework that stays resilient against various white-box adversarial attacks. The results show that our framework performs well in the presence of cyber-attacks and has up to 60% higher resiliency compared to the most resilient individual ML method.

Keywords: Cybersecurity in Industrial IoT, Predictive Maintenance, Adversarial Machine Learning, Ensemble Learning

1. Introduction

Industry 4.0 is the latest industrial revolution aiming to develop fully automated production systems. This idea brings the notion of Industrial Internet of Things (I-IoT) which paves the way for full automation, and higher reliability using computer networks to collect big data from the connected machines and convert this data into actionable information (Zhao et al. 2016). However, these systems are often designed without security in mind or use communication protocols that are not sufficiently secure and vulnerable off-the-shelf commercial products (Tuptuk and Hailes 2018; Wu et al. 2018; *Good Practices for Security of Internet of Things in the context of Smart Manufacturing* 2018). I-IoTs numerous small-scale devices, with their limited computation and communication capabilities, make them vulnerable to potential attacks. An attacker can discover these vulnerabilities and exploit them to steal information, sabotage communication, prevent asset availability, and corrupt monitoring data (Tuptuk and Hailes 2018). These cyber-attacks might result in serious negative financial outcomes, e.g., average estimated loss of $10.7 million per breach of data among manufacturing organizations in Asia Pacific in 2019 (*Understanding the Cybersecurity Threat Landscape in Asia Pacific* 2019). To minimize these costs, cyber-security measures should be taken such as cyber-security awareness training, keeping softwares up-to-date, installing a firewall, using strong passwords (Thames and Schaefer 2017; He et al. 2019).

Predictive maintenance (PDM) goal is to find an optimal maintenance schedule based on time-to-failure prediction of an asset (Khan et al. 2020). It is becoming a more common practice across the industry where its global market size is expected to grow from $4.0 billion in 2020 to $12.3 billion by 2025 (*Predictive Maintenance Market* 2020). Remaining useful life

[*]Corresponding author
Email addresses: ogungor@ucsd.edu (Onat Gungor), tajana@ucsd.edu (Tajana Rosing), baksanli@sdsu.edu (Baris Aksanli)

(RUL) estimation is a crucial PDM application (Kopuru, Rahimi, and Baghaei 2019). Recently, data-driven RUL prediction methods (ML approach) became popular since I-IoT-based instrumentation has led to abundance of system monitoring data. Some of these ML methods include long short-term memory (Zheng et al. 2017), auto-encoders (Bampoula et al. 2021), convolutional neural network (X. Li, Ding, and Sun 2018) etc. However, the performance of ML methods relies heavily on input data quality. Thus, these methods are quite vulnerable to adversarial attacks where an attacker can alter input data or model parameters worsening ML prediction performance significantly. Since ML is in the center of data-driven RUL prediction, these attacks may have serious consequences such as wrong maintenance decisions causing undetected failures in a system (Mode and Hoque 2020). We need novel ML solutions that can stay resilient against adversarial attacks.

The performance of an ML-based application depends also on the specific ML algorithm used. Selecting a single ML method is a difficult process since its performance may change drastically based on the underlying dataset (Güngör, Akşanlı, and Aydoğan 2019). For adversarial attacks, it is easier to decode ML model parameters for a single method, reducing the system resiliency against attacks (Mode, Calyam, and Hoque 2019). Alternatively, ensemble learning combines multiple individual algorithms (i.e., base learners) and it usually improves base learner prediction performance (Shi et al. 2020; Gungor, Rosing, and Aksanli 2021b). Against adversarial attacks, many ensemble learning studies are proposed that are more resilient than a single method (Pang et al. 2019; Mirzaeian et al. 2020; Löwe et al. 2021). To the best of our knowledge, ensemble learning has not been used previously to show its superior resiliency in RUL prediction in an I-IoT setting.

In this work, we propose a stacking ensemble learning framework which can stay resilient against four different adversarial attack scenarios: fast gradient sign, basic iterative, momentum iterative, and robust optimization. We first train 10 different deep learning (DL) methods from three different architectures: recurrent, convolutional, and hybrid. Our ensemble learner then combines the most resilient DL method predictions based on our iterative selection procedure. Using NASA C-MAPSS (Saxena et al. 2008), and UNIBO Powertools (Wong et al. 2021) dataset, we demonstrate that adversarial attacks can impact the performance of DL-method considerably, leading up to 120× prediction performance loss which can lead to premature replacements or completely missed maintenance decisions. We use this performance loss to quantify method resiliency, where more resilient methods would lead to smaller performance loss. Our experiments show that proposed stacking ensemble improves resiliency against adversarial attacks by up to 60% (48% on average) compared to the most resilient single method.

The rest of the paper is organized as follows: Section 2 lists the relevant studies. Section 3 demonstrates our proposed stacking ensemble learner framework. Section 4 presents the main results of our study. Section 5 concludes the paper.

2. Related Work

To understand where possible cyber-attacks can come from an adversary in an I-IoT environment, we present Figure 1 which illustrates DL enabled I-IoT architecture and its threat model. It consists of 4 main layers: physical, edge, cloud, and visualization (Mode and Hoque 2020). Physical layer contains industrial equipments such as machinery, actuators, and sensors (Qiu et al. 2020). The collected data from multiple sensors are sent to edge layer where gateway first collects this data (K. Wang et al. 2016). Then, it is pre-processed to be sent to cloud layer. Note that edge layer also keeps pre-trained DL models for data analytics. Cloud layer first collects the data (sent by edge layer) and trains DL models. These trained models are sent back to the edge layer (thus we have pre-trained models in edge layer). DL models in the cloud layer may require retraining when new data arrives and retrained models are sent back to the edge layer to keep prediction performance at a certain level. According to our architecture, DL model training can only happen at cloud layer, yet there are some works where DL training can occur both at edge and cloud layers (Thomas et al. 2019). The visualization

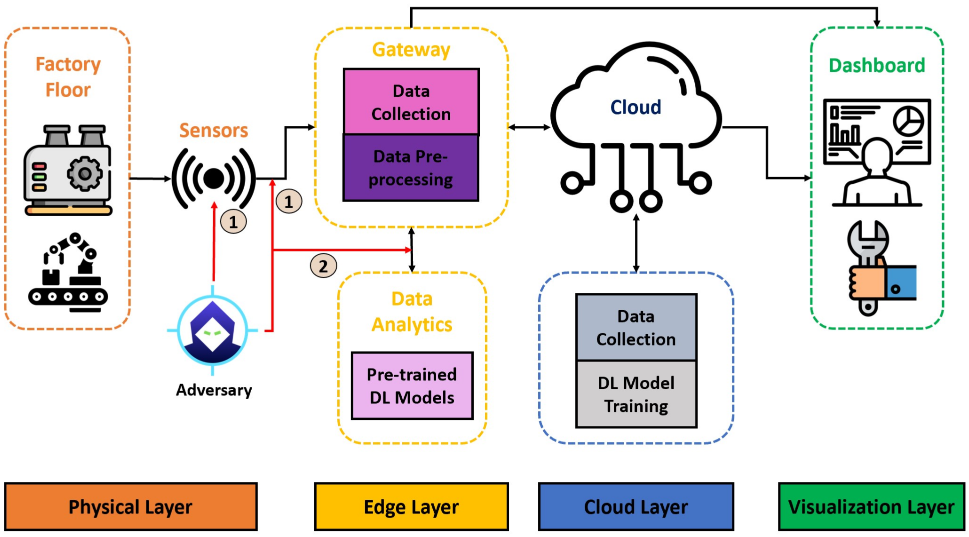


Figure 1: I-IoT Architecture and Threat Model

layer utilizes data from both edge and cloud layers and provides a visual representation of actionable insights to an engineer. In Figure 1, we also provide practical threat model where an adversary can attack to an I-IoT system. Here, we illustrate two different realistic attack scenarios: 1) **attacks on sensors and sensor networks** where an adversary can exploit the sensors and the network between sensors and gateway for different purposes such as transferring malicious code, capturing sensitive information shared between devices (Subramanian et al. 2013). 2) **attacks on DL models** where an attacker exploits pre-trained DL model knowledge to create perturbed examples leading to worse prediction performance (Anthi et al. 2021). The focus of this paper is on the latter since these attacks may have catastrophic consequences (e.g., undetected failures) and they are much harder to detect by defense mechanisms.

2.1. Cyber-security in I-IoT

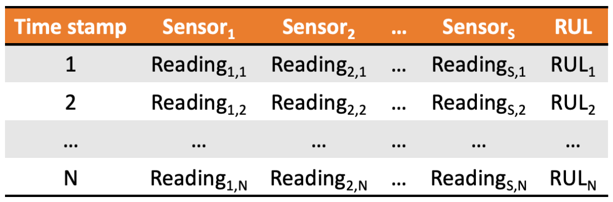
For I-IoT systems, cyber-security is a great challenge because of inadequate standardization, and the lack of required skills to implement them (Lezzi et al. 2018). Wu et al. (2018) summarize the manufacturing assets that are vulnerable to cyber-attacks under 4 categories: operating systems or firmware, application software, industrial communication protocols, and smart devices. Similarly, recent work by Corallo et al. (2021) identify the critical assets to be protected against cyber-attacks and make an assessment on the business impacts of these cyber-attacks using CNC machines and 3-D printers. Corallo et al. (2020) emphasize that these vulnerabilities can be exploited by acting on data where data may be improperly modified, or their flow may be interrupted. Increasing cyber-security awareness is one of the crucial activities towards more secure I-IoT systems. Corallo et al. (2022) analyze how the existing works deal with cyber-security awareness in the context of I-IoT. They categorize the main elements of cybersecurity awareness under three groups: 1) ensuring greater protection of data, information, and networks, 2) raising knowledge level about security threats, risks, and system vulnerabilities, 3) providing knowledge to employees to be responsible for information security and to be aware of cyber-attacks. We are witnessing an increasing trend in cyber-attacks, e.g., cyber-threats against factories increased by more than 200% in 2019 (*Smart Futures* 2019). To ensure security in a production environment, it is also important to understand possible cyber-attack types. Tuptuk and Hailes (2018) summarize the common attacks in a manufacturing environment under 13 different categories: denial of service, eavesdropping, man-in-the-middle, false data injection, time delay, data tampering, replay, spoofing, side channel, covert-channel, zero day, physical, and attacks against machine learning. In this paper, we focus on adversarial attacks against ML where an adversary corrupts the collected data or model parameters through attacks. An attacker exploits ML model information to create malicious attacks. These attacks manipulate legitimate inputs (by adding really small amount of noise) and force a trained model to produce incorrect outputs leading to worse prediction performance. This can bring serious negative implications on a production environment such as undetected failures in the system.

2.2. *Adversarial Attack Methods Against Deep Learning*

Deep learning (DL) has become extremely popular for predictive maintenance (PDM) due to its superior prediction performance (Zhang, Yang, and H. Wang 2019). Since it is a common ML method, there are a lot of potential vulnerabilities such as program errors (leading to software crashes, an infinite loop, or full memory depletion), and attacks at the time of its testing (inserting little noise to test data causing worse performance) (Tariq et al. 2020). This creates vulnerabilities for data-driven PDM. There are three types of adversarial attacks against ML in the literature: evasion, poisoning, and exploratory (Chakraborty et al. 2018). Evasion attacks target compromising the test data, poisoning attacks contaminate the training data, and exploratory attacks gain knowledge about the learning algorithm without changing the data. We focus on evasion attacks since it is the most common type of attack in an adversarial setting (ibid.). Evasion attacks can further be categorized into two groups: white-box and black-box attacks. While white-box attacks have detailed knowledge about the model, black-box attacks assume no knowledge about the underlying model. We consider white-box attacks because they are stronger attacks and can be considered as worst-case scenarios to evaluate system resiliency. We analyze 4 white-box adversarial attack methods: fast gradient sign, basic iterative, momentum iterative, and robust optimization.

2.2.1. *Adversarial Attack Formulation on RUL prediction*

The multivariate time-series input data with its corresponding RUL values is illustrated in Figure 2. In this figure, we have



Time stamp	Sensor$_1$	Sensor$_2$	...	Sensor$_S$	RUL
1	Reading$_{1,1}$	Reading$_{2,1}$	...	Reading$_{S,1}$	RUL$_1$
2	Reading$_{1,2}$	Reading$_{2,2}$	...	Reading$_{S,2}$	RUL$_2$
...	...	...	...	...	...
N	Reading$_{1,N}$	Reading$_{2,N}$	...	Reading$_{S,N}$	RUL$_N$

Figure 2: Multivariate time-series data illustration

S sensor data from N consecutive time stamps where each cell represents individual sensor readings.

Accordingly, we make the following mathematical definitions:

1. $\tau_i \in \mathbb{R}^S$: $[Reading_{1,i}, Reading_{2,i}, \ldots, Readings_{S,i}]$ is the vector containing all sensor readings for the time stamp i, $\forall i = 1, \ldots, N$.
2. $T \in \mathbb{R}^{S\times N}$: $[\tau_1, \tau_2, \ldots, \tau_N]$ represents the multivariate time-series data.
3. $D \in \mathbb{R}^{(S+1)\times N}$: $[(\tau_1, RUL_1), (\tau_2, RUL_2), ..., (\tau_N, RUL_N)]$ denotes the supervised training data.
4. $f(.) \in F : \mathbb{R}^{S\times N} \mapsto \mathbb{R}^N$ is DL model which maps all sensor readings to the remaining useful life prediction values $R\hat{U}L$.
5. $L_f(.,.)$ denotes the loss function of the model f.
6. $\ddot{T} = T + \delta T$ is the crafted adversarial example. $\ddot{T}$ is obtained by adding a perturbation δT with the sample T such that $R\ddot{U}L \neq R\hat{U}L$ and $||\ddot{T} - T|| \leq \epsilon$ where $\epsilon \geq 0 \in \mathbb{R}$ is a maximum perturbation magnitude, $||.||$ is any norm w.l.o.g (e.g., L_∞), $f(T) = R\hat{U}L$, and $f(\ddot{T}) = R\ddot{U}L$.
7. Given a trained DL model f and original data T, adversarial example $\ddot{T}$ is found as a solution to the following box-constrained optimization problem:

$$\ddot{T} = T + \underset{\delta T}{\text{argmin}}\{||\delta T|| : f(T + \delta T) \neq f(T)\} \qquad (1)$$

This problem yields the minimum perturbation amount δT while ensuring that RUL prediction is altered.

Most DL models make this formulation (Equation 1) nonlinear and non-convex, making it hard to find a closed-form solution (ibid.). Hence, we implement different techniques to find an approximate solution to this optimization problem.

2.2.2. Fast Gradient Sign Method (FGSM)

FGSM was suggested as an efficient attack method to fool the GoogLeNet model (I. J. Goodfellow, Shlens, and Szegedy 2014). This method initially calculates the gradient of the cost function with respect to the input of the neural network. Adversarial examples are created based on a gradient direction:

$$\ddot{T} = T + \epsilon * sign(\nabla_\tau L_f(T, R\hat{U}L)) \tag{2}$$

where ϵ denotes the amount of the perturbation.

2.2.3. Basic Iterative Method (BIM)

BIM is an extension of FGSM where FGSM is applied multiple times with really small step size (Kurakin, I. Goodfellow, S. Bengio, et al. 2016). At each iteration of the algorithm, BIM perturbs the original data in the direction of the gradient multiplied by the step size α:

$$\ddot{T} = T + \alpha * sign(\nabla_\tau L_f(\ddot{T}, R\hat{U}L)) \tag{3}$$

where α is calculated by dividing the amount of perturbation by the number of iterations: $\alpha = \epsilon/I$. Then, BIM clips the obtained time series elements to make sure that they are in the ϵ-neighborhood of the original time series:

$$\ddot{T} = min\{T + \epsilon, max\{T - \epsilon, \ddot{T}\}\} \tag{4}$$

2.2.4. Momentum Iterative Mehod (MIM)

MIM integrates momentum into the BIM to stabilize the update directions and to escape from poor local maxima (Dong et al. 2018). At each iteration i, the variable g_i gathers the gradients with a decay factor μ:

$$g_{i+1} = \mu * g_i + \frac{\nabla_\tau L_f(\ddot{T}_i, R\hat{U}L)}{\|\nabla_\tau L_f(\ddot{T}_i, R\hat{U}L)\|_1} \tag{5}$$

where the gradient is normalized by the L_1 distance. Then, the perturbed data is generated in the direction of the sign of g_{i+1} with a step size α:

$$\ddot{T}_{i+1} = \ddot{T}_i + \alpha * sign(g_{i+1}) \tag{6}$$

In MIM, the algorithm also ensures that the crafted adversarial examples $\ddot{T}$ satisfy the L_∞ norm bound constraint:

$$\|\ddot{T} - T\|_\infty \leq \epsilon$$

2.2.5. Robust Optimization Method (ROM)

The general goal in a supervised learning problem is to find model parameters θ that minimize the empirical risk $\mathbb{E}_{(T,RUL)\sim\Xi}[L(T, RUL, \theta)]$ where Ξ is the underlying supervised data distribution. However, this formulation cannot handle data adversary properly. To solve that problem, set of allowed perturbations Δ is introduced initially.

Then, we modify the empirical risk formulation by feeding samples from the distribution Ξ directly into the loss L which leads to the following min-max optimization formulation (Madry et al. 2017):

$$\min_\theta \zeta(\theta), \quad where\ \zeta(\theta) = \mathbb{E}_{(T,RUL)\sim\Xi}[\max_{\delta\in\Delta} L(T + \delta, RUL, \theta)]. \tag{7}$$

Here, while inner maximization finds an adversarial version of a given data point T that achieves a high loss, outer minimization discovers model parameters to minimize the adversarial loss given by the inner attack problem. ROM replaces every instance with its FGSM-perturbed counterpart to solve this problem.

While all these four methods use the gradient information of the loss function, they modify the test data by adding different amounts of perturbation representing separate attack scenarios. An attacker, who is able to access the trained DL methods, can implement these methods and harm the prediction performance without being detected.

2.3. Adversarial Attacks in Predictive Maintenance (PDM)

Adversarial attacks targeting PDM applications can bring serious outcomes such as delayed maintenance/replacement of a machine (Mode and Hoque 2020). There are few studies that analyze the impact of adversarial attacks on data-driven PDM. Mode, Calyam, and Hoque (2019) focus on false data injection attack (FDIA) on PDM systems which alters the collected sensor data by a very small margin. They demonstrate the impact of different FDIA techniques (e.g., continuous, random) on different DL methods e.g., gated recurrent unit (GRU), convolutional neural network (CNN) using NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset (Saxena et al. 2008). Their results show that CNN is extremely sensitive to attacks while GRU is the most resilient method. Further work by Mode and Hoque (2020) analyze the effect of adversarial attacks against ML methods. Specifically, they utilize Fast Gradient Sign Method (FGSM) and Basic Iterative Method (BIM) to create adversarial examples and compare performances of different DL models under those attacks. Again, by using NASA C-MAPSS dataset, they show that these attacks can cause up to 5× worse prediction performance. These two similar works consider limited number of DL methods and attack scenarios. Besides, their experimental analysis includes the simplest and most predictable data set from C-MAPSS. They also did not propose a novel ML solution to increase system resiliency against adversarial attacks.

2.4. Ensemble Methods

Under different adversarial attack scenarios, single ML method prediction performance can change significantly (Mode, Calyam, and Hoque 2019). Ensemble learning is an effective solution towards more generalizable and robust models. It combines a variety of ML algorithms based on three well-known methods (Polikar 2012): 1) bagging combines similar types of learners from different subsamples of the training data (e.g., random forest), 2) boosting fixes the prediction errors of a prior model in the sequence of models (e.g., AdaBoost), 3) stacking combines the predictions of different types of learners using a second-level learner (meta-learner). In PDM domain, there are multiple ensemble learning approaches towards more accurate predictions. We summarize some of the most recent ensemble works in Table 1 including our paper. In this table, we provide the authors, publication year, the research goal, and the proposed ensemble method of the corresponding work. Ensemble learners are especially useful to provide additional security against cyber-attacks since they can learn more robust features (Kurakin, I. Goodfellow, S. Bengio, et al. 2018; Liao et al. 2018). Against adversarial attacks, different ensemble learners are proposed for image classification. Pang et al. (2019) present a diversity promoting ensemble improving adversarial robustness while maintaining state-of-the-art accuracy. Mirzaeian et al. (2020) propose a resilient ensemble where each member learns a radically distinct latent space through diverse knowledge distillation. This method improves security of the state-of-the-art defense methods.

To the best of our knowledge, our work is the first to use ensemble learning towards more resilient PDM. Our ensemble results are also more generalizable since we increase the number of attack scenarios, deep learning models, and experimental datasets significantly compared to the state-of-the-art.

3. Proposed Stacking Ensemble Learner Framework

3.1. Selected Deep Learning (DL) Methods

We select 10 different DL models from recurrent (RNN, LSTM, BLSTM, GRU, BGRU), convolutional (CNN, WAVE), and hybrid architectures (CLSTM, CGRU, GLSTM). With these 10 models, we cover a good range of DL methods from different architectures, increasing the generalizability of our study.

1) Recurrent Neural Network (RNN): RNN is a time-aware feedforward neural network (Géron 2019). Our network contains 3 RNN layers having 64, 32, and 16 units which are consecutively connected to 2 fully connected feed forward neural networks (each with 8 units). Final 1-dimensional output layer provides the RUL prediction.

2) Long Short-Term Memory (LSTM): LSTM has special memory cells to store information for longer. Updates in this cell can happen by the activation of three distinctive gates: 1) forget gate (the memory cell is cleared completely), 2) input gate (memory cell stores the received input), and 3) output gate (next neurons obtain the stored knowledge from the memory cell) (Gensler et al. 2016). We adapt a similar network structure where RNN layers are replaced with LSTM layers.

Table 1: Ensemble Methods for PDM Related Work

Author	Research Goal	Proposed Method
Z. Li, Goebel, and Wu (2019)	More accurate RUL prediction	Combine multiple traditional ML methods (e.g., random forest, elastic net) using particle swarm optimization and sequential quadratic programming to discover optimal weights of the base learners.
Shi et al. (2020)	More accurate RUL prediction	Combine multiple traditional ML methods (e.g., extra tree, random forest) by using the most diverse base learners and features from different degradation stages.
Gungor, Rosing, and Aksanli (2021a)	Minimize retraining overhead while keeping RUL prediction accuracy at a certain level	Combine different DL methods (e.g., convolutional neural network, long short-term memory) by discovering the most accurate and diverse base learners iteratively.
Our paper (STEWART)	Resilient stacking ensemble learner framework against adversarial attacks in RUL prediction	Combine different DL methods (e.g., gated recurrent unit, convolutional neural network) using a stacking ensemble to find the most resilient base learners.

3) Bi-directional LSTM (BLSTM): BLSTM also considers future data by adding a backward direction to LSTM networks (J. Wang et al. 2018). The overall network structure is similar to LSTM model where LSTM layers are replaced with BLSTM layers.

4) Gated Recurrent Unit (GRU): GRU is a simplified version of LSTM (Cho et al. 2014). Specifically, forget and input gates are controlled by a single gate controller and there is no output gate, instead a new gate controller decides which part of the information to be transferred. We use the same network structure as LSTM except we change the LSTM layers to GRU.

5) Bi-directional GRU (BGRU): Similar to BLSTM, BGRU takes future data into consideration (She and Jia 2021). We construct this model by simply replacing BLSTM layers with BGRU layers.

6) 1-D Convolutional Neural Network (CNN): 1D convolutional layer slides kernels across a sequence, producing a 1D feature map per kernel and each kernel learns to detect a single very short sequential pattern (Géron 2019). We adopt the 1-D CNN network proposed by Li et al. (X. Li, Ding, and Sun 2018) which contains five consecutive CNN layers, Flatten (Dropout) layer, one fully-connected layer (with 100 nodes) and an output layer with 1 node.

7) Wavenet (WAVE): We implement the model proposed by Géron (2019) where we stack 4 layers of 1-D causal convolutional layers with 1, 2, 4, and 8 dilation rates (with 20 filters of size 2) two times. These layers are connected to another convolutional layer with 10 filters of size 1 and Flatten layer. Lastly, connection to fully connected neural network (with 100 units) and output layer (with 1 unit) is performed.

8) CNN-LSTM (CLSTM): We combine our LSTM architecture with 1-D CNN where we utilize in parallel connected CNN and LSTM layers. These two paths are then concatenated and connected to fully connected neural network (with 100 units) and output layer (with 1 unit). Similar parallel CNN-LSTM structure was recently proposed by Al-Dulaimi et al. (2019).

9) CNN-GRU (CGRU): We connect our CNN and GRU networks in parallel. On one path, we have our 1-D CNN architecture, on another path we have our GRU model. These are then connected to fully connected neural network with 100 units and output layer (with 1 unit).

10) GRU-LSTM (GLSTM): We combine our GRU and LSTM models in parallel. On one path, we have our GRU architecture, on another path we have our LSTM model. These two paths are concatenated and connected to fully connected neural network (with 100 units) and output layer (with 1 unit).

3.2. Deep Learning Methods Compromise Calculation

In order to quantify the resiliency of DL models, we use our framework presented in Figure 3. The process starts with training 10 DL algorithms with training data (see Section 3.1 for model details). The trained models are then evaluated under two different test data: 1) normal, and 2) perturbed data. Adversary creates the perturbed data by adding imperceptible noise to the normal test data. This noise generation process is obtained by using one of the selected adversarial methods in Section 2.2. Predictive models output two different remaining useful life (RUL) estimations: normal RUL predictions, and compromised RUL predictions. Given true RUL values, our error metric root mean squared error (RMSE) is calculated for both normal and compromised prediction scenarios based on the following formulation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\epsilon_i^2} \tag{8}$$

where N is the number of samples, ϵ is the difference between the estimated RUL (RUL_{est}) and the true RUL (RUL_{true}). Using these error values, we calculate the DL model compromise which is formulated as:

$$Compromise = \frac{RMSE_{compromised}}{RMSE_{normal}} \tag{9}$$

where $Compromise > 1$ (under the assumption that attacks lead to worse prediction performance). The **smaller** the compromise value, the **more resilient** the model is against the adversarial attack. For instance, given two methods CNN and LSTM, and their compromise values 8, and 5 respectively, we can conclude that LSTM is **more resilient** against the adversarial attack. If we have M number of adversarial attacks (where $M > 1$), then we need to calculate the mean compromise value for each DL method as follows:

$$Compromise_{mean} = \left(\sum_{i=1}^{M}\frac{RMSE^{i}_{compromised}}{RMSE_{normal}}\right)/M \tag{10}$$

Since we have multiple attack techniques (in our case $M = 4$), this metric gives a more accurate idea about single model resilience. Overall, we obtain mean compromise values for each DL model from Section 3.2.

3.3. Stacking Ensemble Learner

Stacking (short for stacked generalization) is one of the most-used ensemble learning methods (other than bagging, and boosting) where single method predictions are aggregated using a second-level learner, or meta-learner (Géron 2019). Since our ensemble learner combines different DL model predictions, we select stacking as the most suitable ensemble approach.

3.3.1. Ensemble Learner Training

We present the general framework for stacking ensemble training in Figure 4. We start the training process by splitting training data into two subsets. We use the first subset (subset 1) to train the DL models. Here, for the sake of simplicity, the figure shows only two methods, namely CNN and RNN. After model training is completed, we obtain our predictive models. These models are used to make prediction on the subset 2. Basically, each DL method outputs RUL predictions using subset 2. Then, these RUL predictions are given to the stacking ensemble training for which different meta-learners are trained. This training part is different from DL model training. In DL training, as an input we have time series data, yet in ensemble training, we have the RUL prediction values obtained from different DL methods. As an output of ensemble learner training, we obtain our predictive ensemble models. For illustration purposes, linear regression (LR) is used as the meta-learner to map single model RUL predictions to real RUL values in Figure 4. Overall, we train 4 different meta-learners to find out the most resilient one against adversarial attacks:

1) Linear Regression (LR): This linear model makes a prediction by calculating a weighted sum of the input features, plus

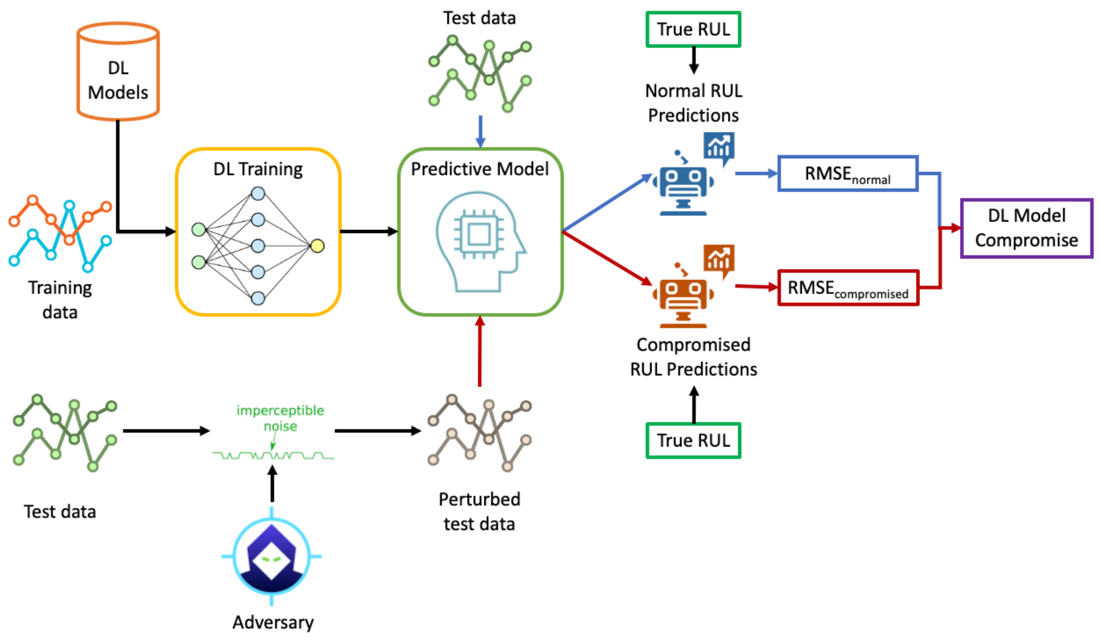


Figure 3: Framework for DL Methods Compromise Calculation

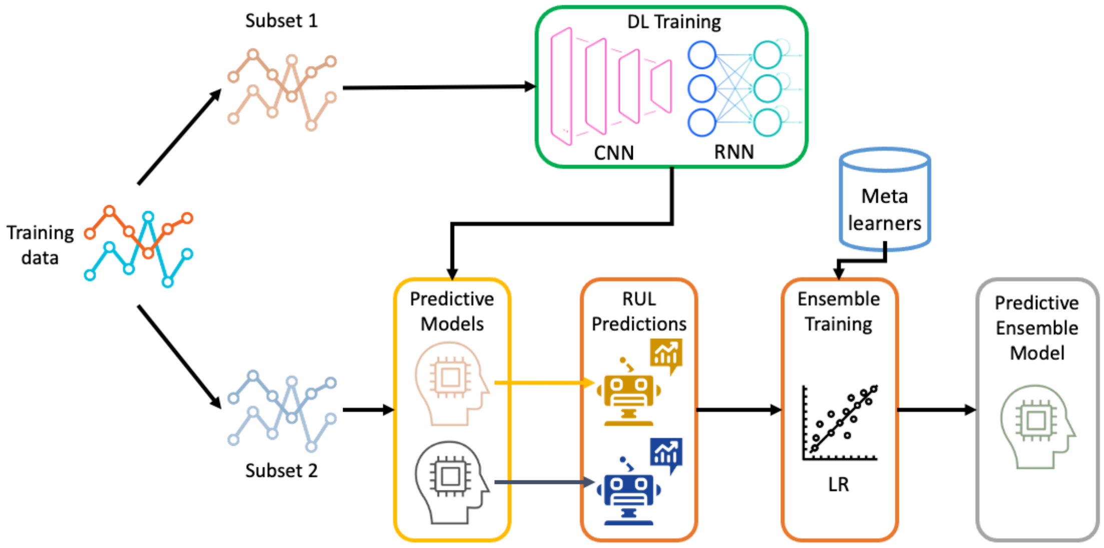


Figure 4: Framework for Stacking Ensemble Training

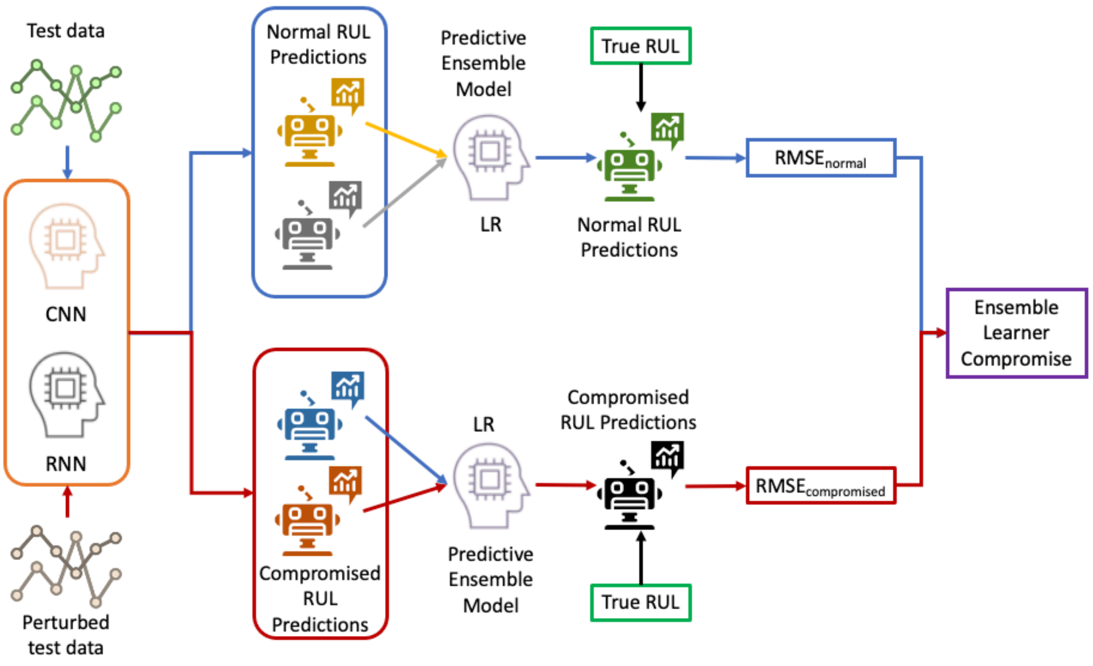

Test data
CNN
RNN
Perturbed test data
Normal RUL Predictions
Compromised RUL Predictions
Predictive Ensemble Model
LR
LR
Predictive Ensemble Model
True RUL
Normal RUL Predictions
Compromised RUL Predictions
True RUL
RMSE_normal
RMSE_compromised
Ensemble Learner Compromise

Figure 5: Framework for Stacking Ensemble Testing

a bias term (Géron 2019): $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ where $\hat{y}$ indicates the predicted ensemble RUL value, n is the number of DL models, x_i is the ith DL model RUL prediction, θ_i ($\forall i = 1, \ldots, n$) is the ith DL model weight, and θ_0 is the bias term.

2) Random Forest (RF): RF is an ensemble of decision trees trained by the bagging method (ibid.). The algorithm constructs multiple decision trees at training time and outputs the mean prediction of the individual trees.

3) AdaBoost: AdaBoost is one of the most famous boosting approaches where focus is given to the training instances that the predecessor underfitted. The weights of instances are adjusted according to the error of the current prediction (ibid.). That is, subsequent estimators focus more on difficult cases.

4) Extreme Gradient Boosting (XGBoost): XGBoost is an an efficient and effective implementation of the gradient boosting algorithm. Gradient boosting, different than AdaBoost, fits the new predictor to the residual errors made by the previous predictor (ibid.). The two main reasons why XGBoost is heavily used are execution speed and model performance.

3.3.2. Ensemble Learner Test

We test our stacking ensemble learner based on the framework provided in Figure 5. Similar to single DL model testing, we obtain the compromise value for our ensemble learner as an output. Given normal test data and perturbed test data (crafted by the adversary using adversarial attack methods described previously), pre-trained (predictive) DL models (e.g. CNN, RNN) make normal and compromised RUL predictions. These single method predictions are then given to our pre-trained (predictive) ensemble model (e.g. LR) to generate ensemble normal and compromised RUL predictions. Similarly, the compromise value is calculated by dividing the compromised RMSE by the normal RMSE. Since we have multiple attack scenarios, we need to calculate mean compromise for our ensemble learner formulated in Equation 10. To show the benefit of our ensemble learner, we calculate the ensemble improvement over single method based on the following formulation:

$$Improvement = \left(\frac{Compromise_{single} - Compromise_{ensemble}}{Compromise_{single}} \right) \tag{11}$$

where $Compromise_{single}$ denotes the single DL model mean compromise value, and $Compromise_{ensemble}$ is the ensemble mean compromise value. We report the improvement in percentage (%). Here, improvement demonstrates the resiliency of our ensemble learner against adversarial attacks compared to a single learner. The higher the improvement is, the more resilient our ensemble learner is compared to single DL model.

3.3.3. Most Resilient Stacking Ensemble Selection

In order to determine the most resilient ensemble learner configuration, we follow our proposed solution procedure presented in Algorithm 1. This algorithm increases the number of base learners (to be used in the ensemble) iteratively, and finds the most resilient ensemble configuration where resiliency can no longer be improved. Given single method mean compromise values C, the algorithm first sorts C in an ascending order. We start the ensemble search with the 2 most resilient methods. We train the ensemble, test it, and calculate the ensemble mean compromise value using these two methods. The function that calculates ensemble mean compromise is also provided in Algorithm 2. This function first trains the ensemble learner given true RUL values and base learner RUL predictions (Figure 4). Then, it makes ensemble RUL predictions for both normal and perturbed test data. As an input, it uses single DL method normal and perturbed test data RUL predictions. The algorithm then calculates the RMSE for both normal and compromised scenarios using real RUL values and ensemble RUL predictions. It finally finds out the ensemble mean compromise (Figure 5). After we obtain ensemble compromise value, we check if this value is smaller than the single best method compromise and update the best compromise accordingly. We then continue with the next most resilient method selection and add this method to our base learner subset. For this new ensemble configuration, we calculate its compromise value (Algorithm 2) and update the best compromise if it improves the best compromise value. If there is no improvement, we increment the variable *worsenedcounter*. This variable controls whether we should continue or terminate the ensemble search process. We allow only a fixed number of iterations with performance decrease, *worsenedtolerance*, after which we terminate the search process and return the best compromise value.

Algorithm 1: Most Resilient Stacking Ensemble Selection

```
Input  : $C = [C_1, C_2, \dots, C_N]$ (single method mean
         compromise values)
Output: bestcompromise
1  ensemblecompromise = $\infty$, i = 2, worsenedcounter = 0,
     worsenedtolerance = 2;
2  $[C_{sorted}, I] = sort(C, ascend)$;
3  bestcompromise = $C_{sorted}(1)$;
4  while $i \leq N$ do
5      if $i == 2$ then
6          $model_{indexes} = I(1:i)$;
7          $ensemblecompromise =$
             $calculateCompromise(\dots, model_{indexes})$;
8          if $ensemblecompromise < bestcompromise$ then
9              $bestcompromise = ensemblecompromise$
10         end
11     end
12     else
13         $model_{indexes} = [model_{indexes}, I(i)]$;
14         $ensemblecompromise =$
             $calculateCompromise(\dots, model_{indexes})$;
15         if $ensemblecompromise < bestcompromise$ then
16             $bestcompromise = ensemblecompromise$;
17         end
18         else
19             worsenedcounter++;
20             if $worsenedcounter == worsenedtolerance$ then
21                 break;
22             end
23         end
24     end
25     $i++$;
26 end
27 return bestcompromise;
```

Algorithm 2: Calculate Compromise

```
Input  : $RUL_{subset2}$ (true RUL values for training subset 2),
         $RUL_{test}$ (true RUL values for test data),
         $R\hat{U}L = [R\hat{U}L_1, R\hat{U}L_2, \dots, R\hat{U}L_N]$ (single method
         RUL predictions for training subset 2),
         $R\tilde{U}L = [R\tilde{U}L_1, R\tilde{U}L_2, \dots, R\tilde{U}L_N]$ (single method
         RUL predictions for normal test data),
         $R\ddot{U}L = [R\ddot{U}L_1, R\ddot{U}L_2, \dots, R\ddot{U}L_N]$ (single method
         RUL predictions for perturbed test data), N (number
         of base learners), M (number of attack methods),
         $model_{indexes}$
Output: meancompromise
1  meancompromise, i = 0;
2  while $i < M$ do
3      $ensemble = train_{ensemble}(R\hat{U}L(model_{indexes}), RUL_{subset2})$;
4      $R\tilde{U}L_{ensemble} = test_{ensemble}(R\tilde{U}L)$;
5      $R\ddot{U}L_{ensemble} = test_{ensemble}(R\ddot{U}L)$;
6      $RMSE_{normal} = calculate_{ensemble}(RUL_{test}, R\tilde{U}L_{ensemble})$;
7      $RMSE_{compromised} = calculate_{ensemble}(RUL_{test}, R\ddot{U}L_{ensemble})$
        ;
8      ensemblecompromise = $\frac{RMSE_{compromised}}{RMSE_{normal}}$;
9      meancompromise += ensemblecompromise;
10     $i++$;
11 end
12 meancompromise /= M;
13 return meancompromise;
```

4. Experimental Analysis

4.1. Dataset Description

To validate the improved resiliency of our proposed ensemble learner framework against adversarial attacks, we use two different datasets: NASA C-MAPSS (Saxena et al. 2008), and UNIBO Powertools (Wong et al. 2021).

NASA C-MAPSS is a benchmark dataset for remaining useful life (RUL) estimation. This dataset includes multiple aircraft engines simulated under different operating and fault conditions. Fig. 6 depicts the simplified version of simulated engine diagram and its major components: fan, turbine, compressor, and combustor. The data is collected using various sensors (e.g. temperature, pressure) placed on these components. NASA C-MAPSS comprises of 4 different datasets in increasing complexity: FD001~FD004. Table 2 presents the dataset and their corresponding features. We can observe that while FD001 is the simplest data set, FD004 is the most complicated one (i.e. the highest number of operating and fault conditions). For each dataset, we have separate training and test data where the goal is to predict RUL for the test data. Our feature columns include the engine ID, cycle index, three operational settings, and 21 sensor measurements.

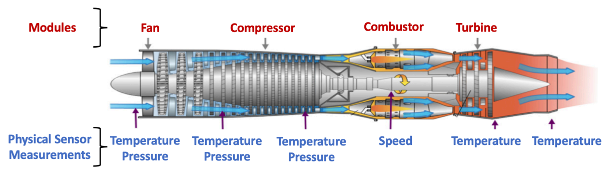


Figure 6: Engine Diagram Simulated in C-MAPSS (Saxena et al. 2008)

Table 2: C-MAPSS Data Set

Data Set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Max/Min cycles for train	362/128	378/128	525/145	543/128
Max/Min cycles for test	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

UNIBO Powertools is a lithium-ion (Li-Ion) battery dataset collected in a laboratory test by an Italian Equipment producer (ibid.). It contains 27 batteries which are run until their end of life. We use 17 of these batteries for training and 10 of them for testing. These batteries have different nominal capacities and they are tested under different conditions: 1) standard test: battery was discharged at 5A current in main cycles, 2) high current test: battery was discharged at 8A current in main cycles, 3) preconditioned test: battery cells are stored at 45°C environment for 90 days before conducting the test. The following procedure is used to create the dataset where during discharge, the sampling period is set to 10 seconds (ibid.): 1) Charge cycle: Constant Current-Constant Voltage (CC-CV) at 1.8A and 4.2V (100mA cut-off), 2) Discharge cycle: Constant Current until cut-off voltage (2.5V), 3) Repeat steps 1 and 2 (main cycle) 100 times, 4) Capacity measurement: charge CC-CV 1A 4.2V (100mA cut-off) and discharge CC 0.1A 2.5V, 5) Repeat the previous steps until the battery cell end of life. We have different columns in this dataset: battery id, time, voltage, current, charging capacity, discharging capacity, watt hour (wh) measurements during charge and discharge, temperature, and cycle count.

4.2. Experimental Setup

4.2.1. Adversarial Attack Methods

We use the following parameters for the selected adversarial methods fast gradient sign, basic iterative, momentum iterative, and robust optimization (Fawaz et al. 2019; Dong et al. 2018; Madry et al. 2017): amount of perturbation(ϵ)=0.1, step size(α)=0.001, number of iterations(I)=100, decay factor(μ)=1.

4.2.2. Deep Learning Methods

Although we use the same model structures (see Section 3.1 for the model details) for both datasets, we select different hyper-parameters to run the models so as to obtain the best possible performance. We replicate each experiment 10 times and report average **compromise** values where we run all experiments on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor.

NASA C-MAPSS: *Adam* optimizer with learning rate 0.001, *elu* activation function, batch size of 128, and a max number of epochs of 150 where callback is activated (patience is set to 10 for validation data), and sliding time window size of 80.

UNIBO: *Adam* optimizer with learning rate 0.0001, *selu* activation function, batch size of 256, and a max number of epochs of 100 where callback is activated (patience is set to 10 for validation data), and sliding time window size of 500.

4.2.3. Stacking Ensemble

For the selected meta-learners, we perform hyperparameter optimization using a grid search (Bergstra and Y. Bengio 2012). This gives us the optimal hyper-parameters to combine predictions from different DL models using stacking ensemble. For the ensemble training, we split training data into two subsets using the ratio 70% (subset 1) to 30% (subset 2). We use subset 1 for DL model training, and subset 2 for ensemble training. We set *worsenedtolerance* to 2 since it leads to the selection of optimal ensemble configuration.

Table 3: Single DL Models Mean Compromise

DL Model / Dataset	FD001	FD002	FD003	FD004	UNIBO
CLSTM	8.0	120.3	6.8	86.2	27.2
CNN	20.6	72.0	13.5	12.5	22.6
WAVE	17.6	25.6	14.4	5.6	7.2
CGRU	9.0	13.6	7.8	6.5	32.8
BLSTM	6.7	8.4	8.7	6.0	10.5
GLSTM	6.4	8.4	7.9	6.2	6.8
BGRU	6.1	7.4	7.5	5.9	7.5
LSTM	5.7	7.9	7.2	5.4	6.3
GRU	**5.2**	7.7	6.5	7.1	**4.5**
RNN	5.3	**4.3**	**5.0**	**4.6**	7.1

4.3. Single DL Models Resiliency

Table 3 presents mean compromise values for each DL method. In this table, each row represents a different DL model and each column corresponds to a distinct dataset. We first observe that DL model performance is impacted poorly by the adversarial attacks where there is up-to 120× compromise. We also notice that the resiliency of a DL method changes with respect to the dataset. Here, we present the most resilient methods at the bottom of the table. We observe that GRU is the most resilient algorithm at FD001, and UNIBO while RNN is the best at the remaining datasets. We can conclude that recurrent architectures (e.g., GRU) are superior over others. CNN-based methods are extremely sensitive to the attacks where the prediction performance degrades by up to 72×. Hybrid methods can be resilient if solely recurrent architectures are combined (e.g., GLSTM). For the most resilient ensemble selection, we utilize these compromise values.

4.4. *Proposed Stacking Ensemble Learner Resiliency*

4.4.1. *Meta-learner Resiliency Analysis*

We first analyze the meta-learner resiliency of our ensemble learner. Figure 7 illustrates the meta-learner mean compromise values where each meta-learner is represented with a distinct color. In each sub-figure, x-axis shows the number of base learners, and the y-axis provides the ensemble compromise. We first note that the meta-learner resiliency fluctuates considerably with respect to the number of base learners. To illustrate, at FD003 (Figure 7c), AdaBoost (ADA) is the most resilient at 5 and 6 base learner ensemble scenarios, yet it is the worst if we only select 3 base learners. The best performing meta-learner also changes based on the number of base learners. However, this is not the case for all datasets. For instance, ADA is always the most resilient meta-learner at FD004 (Figure 7d), and UNIBO (Figure 7e). When we analyze the average performance of each meta-learner over all dataset and ensemble learner configurations, we obtain 6.43, 4.91, 4.88, and 4.47 compromise values for LR, RF, XGB, and ADA respectively. This shows that ADA is the most resilient meta-learner while LR being the least resilient (on average). For the rest of our ensemble analysis, we select the best meta-learner for each ensemble configuration and report those measurements. To exemplify, for UNIBO dataset and any ensemble configuration, we present the ADA compromise values since its value is the smallest (Figure 7e). However, ADA is not selected for any ensemble configuration at FD001.

4.4.2. *Stacking Ensemble Analysis*

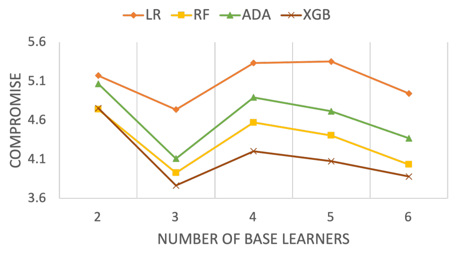
We first analyze the resiliency of ensemble learners having different number of base learners. Figure 8 demonstrates a variety of stacking ensemble learner compromise values under the selected adversarial attack scenarios. In each sub-figure, x-axis shows the attack method, the y-axis denotes the ensemble compromise. Each figure shows the best single method and multiple ensemble learner configurations for different attack methods. Note that in these figures, EnL represents our ensemble learner with n most resilient learners. We consider different number of base learners (from 2 to 7, e.g., 'E2L' uses 2 most resilient base learners) and the most resilient single method ('Best Single'). All methods in each figure are represented with distinct colors and the legend of each figure shows the order in which these methods are presented. Each ensemble compromise value in this figure corresponds to the compromise value of the best meta-learner. We can find the most resilient ensemble configuration from Figure 8 which is the right most bar in each subfigure in Figure 8. For instance, at FD003 (Figure 8c), E5L (represented with yellow color) is the most resilient configuration. E3L (i.e., ensemble learner using 3 most resilient base learners), E2L, E5L, E6L, and E4L are the most resilient ensemble configuration for FD001, FD002, FD003, FD004, and UNIBO respectively. Besides, we observe that increasing the number of base learners does not always lead to more resilient learner. To illustrate, the best performing ensemble at FD002 only uses 2 base learners (Figure 8b). This result motivates us for a more clever ensemble method selection approach which can both terminate the search process early (i.e., it might not be necessary to try all base learners) while it can find the most resilient ensemble configuration.

4.4.3. *Adversarial Attacks Compromise Analysis*

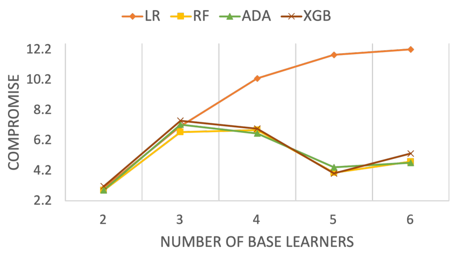
Based on the results in Figure 8, we also analyze the impact of an adversarial attack on the model compromise. Figure 9 shows the average compromise values for each attack method. On the y-axis, we calculate the average compromise over all ensemble and the best single method scenarios, x-axis corresponds to the dataset. Momentum iterative method (MIM) leads to highest compromise (up to 5.8×) whereas BIM is the least strong attack among all.

4.4.4. *Most Resilient Stacking Ensemble Selection*

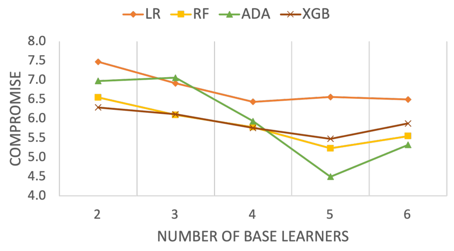
Table 4 presents the results of the most resilient ensemble configuration search process. In this table, the columns show the dataset, best ensemble configuration, mean ensemble compromise, best single method mean compromise, and the ensemble resiliency improvement over the single method respectively.



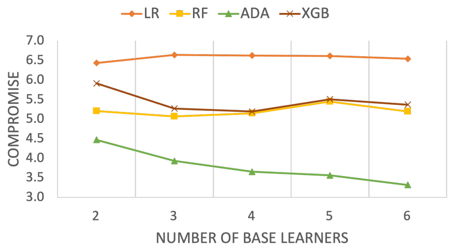
(a) **FD001**



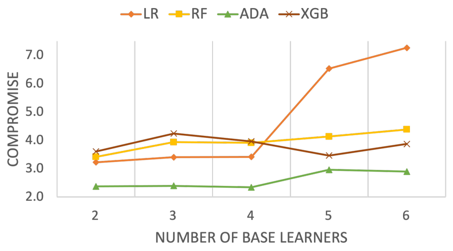
(b) **FD002**



(c) **FD003**

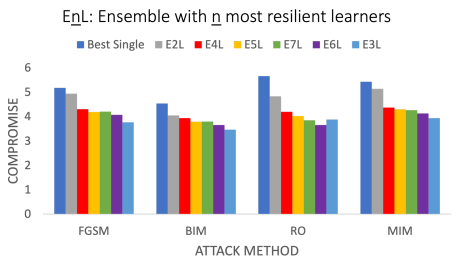


(d) **FD004**

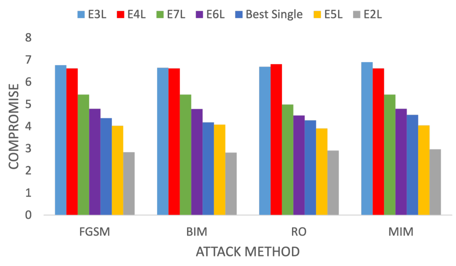


(e) **UNIBO**

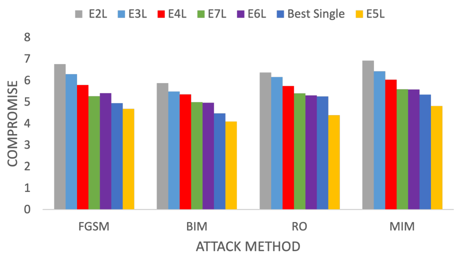
Figure 7: Meta-learner Compromise Analysis



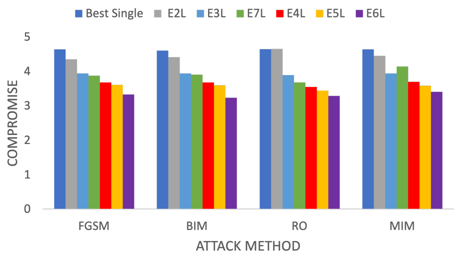
(a) **FD001**



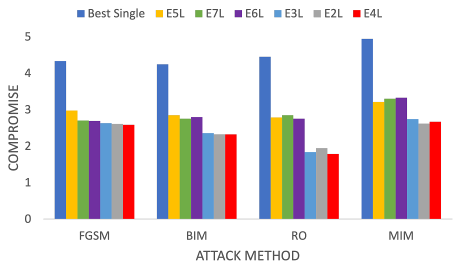
(b) **FD002**



(c) **FD003**



(d) **FD004**



(e) **UNIBO**

Figure 8: Stacking Ensemble Compromise Analysis

Table 4: Most Resilient Stacking Ensemble Configuration

Dataset	**Ensemble Configuration**	**Compromise**	**Best Single Compromise**	**Improvement (%)**
FD001	3 Learners (E3L)	3.76	5.21	27.83%
FD002	2 Learners (E2L)	2.88	**4.34**	33.64%
FD003	5 Learners (E5L)	4.49	5.00	10.20%
FD004	6 Learners (E6L)	3.32	4.63	28.29%
UNIBO	4 Learners (E4L)	**2.34**	4.49	**47.88%**

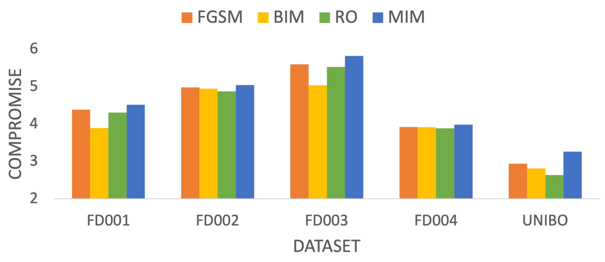


Figure 9: Adversarial Attacks Compromise Analysis

We can observe that the best ensemble configuration is different at each dataset, e.g., E3L (i.e., ensemble learner using 3 most resilient base learners) for FD001, E2L for FD002, and so on. In Section 4.4.2, we find out the most resilient ensemble configurations. We can validate that our proposed algorithm is able to select those ensemble configurations successfully. While we obtain the best ensemble mean compromise at UNIBO (2.34×), the smallest single method compromise is obtained at FD002 (4.34×). Our stacking ensemble approach achieves up to 47.9% mean compromise improvement. We also analyze the proposed stacking ensemble compromise improvement for the adversarial attack methods individually. Table 5 shows our proposed ensemble method's resiliency improvement over the best single method under each attack scenario. We reach up-to 59.9% improvement at UNIBO. For FD001, FD002, FD003, and FD004, the maximum improvements are 31.5%, 35%, 16.5%, and 29.8% respectively.

Table 5: Proposed Stacking Ensemble Compromise Improvement Over the Most Resilient DL Method (%)

Dataset / Attack Method	**FGSM**	**BIM**	**RO**	**MIM**
FD001	27.4	23.6	**31.5**	27.5
FD002	**35.0**	32.6	32.0	34.5
FD003	5.3	8.4	**16.5**	10.1
FD004	28.2	**29.8**	29.2	26.6
UNIBO	40.4	45.3	**59.9**	46.0

5. Conclusions and Future Work

In this work, we propose a stacking ensemble learning framework which is more resilient against adversarial attacks compared to single deep learning (DL) methods. We use 4 different attack methods (fast gradient sign, basic iterative, momentum iterative, and robust optimization) and 10 distinct DL models from recurrent, convolutional, and hybrid architectures. We find that recurrent neural network based architectures provide more resilient learning whereas convolutional neural network structures are extremely sensitive to the attacks. We observe that the most resilient single ML method changes based on the data set or attack method. To address this issue, we propose a framework that finds the most resilient ensemble configuration against multiple attacks. The results show that our proposed ensemble learner framework can improve the resiliency of the most resilient single method by up to 60%. From research perspective, this means that the proposed ensemble solution can still perform well under adversarial attacks. In management level, this leads to more accurate replacement and maintenance decisions even under cyber-attacks.

Limit and Constraints: As we provided in Figure 1, DL-enabled I-IoT systems contain different layers. Cyber-attacks against those systems can target different components such as communication protocols, smart devices, and DL models. Our proposed stacking ensemble learning framework can provide a cyber-security solution against only DL model attacks, not all type of attacks in an I-IoT system. Hence, our proposed method would be a part of wider cyber-security solution towards more resilient I-IoT systems.

Future Work: To overcome these limitations, as a future work, we are first planning to add black-box attack methods which do not have any knowledge about the attacked models. Thus, we can analyze a more realistic attack scenarios and examine the performance of the proposed stacking ensemble learning framework, generalizing the resiliency of our approach.

References

Anthi, Eirini et al. (2021). "Adversarial attacks on machine learning cybersecurity defences in industrial control systems". In: *Journal of Information Security and Applications* 58, p. 102717.

Bampoula, Xanthi et al. (2021). “A Deep Learning Model for Predictive Maintenance in Cyber-Physical Production Systems Using LSTM Autoencoders”. In: *Sensors* 21.3, p. 972.

Bergstra, James and Yoshua Bengio (2012). “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2.

Chakraborty, Anirban et al. (2018). “Adversarial attacks and defences: A survey”. In: *arXiv preprint arXiv:1810.00069*.

Cho, Kyunghyun et al. (2014). “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint*.

Corallo, Angelo et al. (2020). “Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts”. In: *Computers in industry* 114, p. 103165.

— (2021). “Cybersecurity Challenges for Manufacturing Systems 4.0: Assessment of the Business Impact Level”. In: *IEEE Transactions on Engineering Management*.

Corallo, Angelo et al. (2022). “Cybersecurity awareness in the context of the Industrial Internet of Things: A systematic literature review”. In: *Computers in Industry* 137, p. 103614.

Dong, Yinpeng et al. (2018). “Boosting adversarial attacks with momentum”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193.

Al-Dulaimi, Ali et al. (2019). “Hybrid deep neural network model for remaining useful life estimation”. In: *IEEE ICASSP*. IEEE, pp. 3872–3876.

Fawaz, Hassan Ismail et al. (2019). “Adversarial attacks on deep neural networks for time series classification”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Gensler, André et al. (2016). “Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks”. In: *2016 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, pp. 002858–002865.

Géron, Aurélien (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media.

Good Practices for Security of Internet of Things in the context of Smart Manufacturing (2018). `https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot`.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572*.

Gungor, Onat, Tajana S Rosing, and Baris Aksanli (2021a). “DOWELL: Diversity-induced Optimally Weighted EnsembLe Learner for Predictive Maintenance of Industrial Internet of Things Devices”. In: *IEEE Internet of Things Journal*.

— (2021b). “OPELRUL: OPtimally Weighted Ensemble Learner for Remaining Useful Life Prediction”. In: *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, pp. 1–8.

Güngör, Onat, Barış Akşanlı, and Reyhan Aydoğan (2019). “Algorithm selection and combining multiple learners for residential energy prediction”. In: *Future Generation Computer Systems* 99, pp. 391–400.

He, Wu et al. (2019). “Improving employees’ intellectual capacity for cybersecurity through evidence-based malware training”. In: *Journal of Intellectual Capital*.

Khan, Wazir Zada et al. (2020). “Industrial internet of things: Recent advances, enabling technologies and open challenges”. In: *Computers & Electrical Engineering* 81, p. 106522.

Kopuru, M Sri Krishna, S Rahimi, and KT Baghaei (2019). “Recent Approaches in Prognostics: State of the Art”. In: *ICAI*, pp. 358–365.

Kurakin, Alexey, Ian Goodfellow, Samy Bengio, et al. (2016). *Adversarial examples in the physical world*.

Kurakin, Alexey, Ian Goodfellow, Samy Bengio, et al. (2018). “Adversarial attacks and defences competition”. In: *The NIPS’17 Competition: Building Intelligent Systems*. Springer, pp. 195–231.

Lezzi, Marianna et al. (2018). “Cybersecurity for Industry 4.0 in the current literature: A reference framework”. In: *Computers in Industry* 103, pp. 97–110.

Li, Xiang, Qian Ding, and Jian-Qiao Sun (2018). “Remaining useful life estimation in prognostics using deep convolution neural networks”. In: *Reliability Engineering & System Safety* 172, pp. 1–11.

Li, Zhixiong, Kai Goebel, and Dazhong Wu (2019). “Degradation modeling and remaining useful life prediction of aircraft engines using ensemble learning”. In: *Journal of Eng. for Gas Turbines and Power* 141.4.

Liao, Fangzhou et al. (2018). “Defense against adversarial attacks using high-level representation guided denoiser”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787.

Löwe, Mathias et al. (2021). “Dealing with Adversarial Player Strategies in the Neural Network Game iNNk through Ensemble Learning”. In: *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, pp. 1–10.

Madry, Aleksander et al. (2017). “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083*.

Mirzaeian, Ali et al. (2020). “Learning diverse latent representations for improving the resilience to adversarial attacks”. In: *arXiv e-prints*, arXiv–2006.

Mode, Gautam Raj, Prasad Calyam, and Khaza Anuarul Hoque (2019). “False data injection attacks in internet of things and deep learning enabled predictive analytics”. In: *arXiv preprint arXiv:1910.01716*.

Mode, Gautam Raj and Khaza Anuarul Hoque (2020). “Crafting adversarial examples for deep learning based prognostics (extended version)”. In: *arXiv preprint arXiv:2009.10149*.

Pang, Tianyu et al. (2019). “Improving adversarial robustness via promoting ensemble diversity”. In: *International Conference on Machine Learning*. PMLR, pp. 4970–4979.

Polikar, Robi (2012). “Ensemble learning”. In: *Ensemble machine learning*. Springer, pp. 1–34.

Predictive Maintenance Market (2020). `https://www.marketsandmarkets.com/\PressReleases/operational-predictive-maintenance.asp`.

Qiu, Tie et al. (2020). “Edge computing in industrial internet of things: Architecture, advances and challenges”. In: *IEEE Communications Surveys & Tutorials* 22.4, pp. 2462–2488.

Saxena, Abhinav et al. (2008). “Damage propagation modeling for aircraft engine run-to-failure simulation”. In: *IEEE ICPHM*. IEEE, pp. 1–9.

She, Daoming and Minping Jia (2021). “A BiGRU method for remaining useful life prediction of machinery”. In: *Measurement* 167, p. 108277.

Shi, Junchuan et al. (2020). “Remaining Useful Life Prediction of Bearings Using Ensemble Learning: The Impact of Diversity in Base Learners and Features”. In: *Journal of Computing and Information Science in Engineering*, pp. 1–35.

Smart Futures (2019). https://smartmachinesandfactories.com/news/fullstory.php/aid/459/Cyber-attacksonsmartfactoriesareontherise.html.

Subramanian, Venkatachalam et al. (2013). “Examining the characteristics and implications of sensor side channels”. In: *2013 IEEE International Conference on Communications (ICC)*. IEEE, pp. 2205–2210.

Tariq, Muhammad Imran et al. (2020). “A review of deep learning security and privacy defensive techniques”. In: *Mobile Information Systems* 2020.

Thames, Lane and Dirk Schaefer (2017). *Cybersecurity for industry 4.0*. Springer.

Thomas, Anthony et al. (2019). “Hierarchical and distributed machine learning inference beyond the edge”. In: *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, pp. 18–23.

Tuptuk, Nilufer and Stephen Hailes (2018). “Security of smart manufacturing systems”. In: *Journal of manufacturing systems* 47, pp. 93–106.

Understanding the Cybersecurity Threat Landscape in Asia Pacific (2019). https://news.microsoft.com/apac/features/cybersecurity-in-asia/.

Wang, Jiujian et al. (2018). “Remaining useful life estimation in prognostics using deep bidirectional lstm neural network”. In: *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*. IEEE, pp. 1037–1042.

Wang, Kun et al. (2016). “Green industrial Internet of Things architecture: An energy-efficient perspective”. In: *IEEE Communications Magazine* 54.12, pp. 48–54.

Wong, Kei Long et al. (2021). “Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles”. In: *Proceedings of the Conference on Information Technology for Social Good*, pp. 85–90.

Wu, Dazhong et al. (2018). “Cybersecurity for digital manufacturing”. In: *Journal of manufacturing systems* 48, pp. 3–12.

Zhang, Weiting, Dong Yang, and Hongchao Wang (2019). “Data-driven methods for predictive maintenance of industrial equipment: A survey”. In: *IEEE Systems Journal* 13.3, pp. 2213–2227.

Zhao, Rui et al. (2016). “Deep learning and its applications to machine health monitoring: A survey”. In: *arXiv preprint arXiv:1612.07640*.

Zheng, Shuai et al. (2017). “Long short-term memory network for remaining useful life estimation”. In: *IEEE ICPHM*. IEEE, pp. 88–95.