Project School Certificate

Title : FALSE DATA INJECTION ATTACK

Faculty Incharge : Raju

Session Duration : 10/09/2022 – 23/12/2022
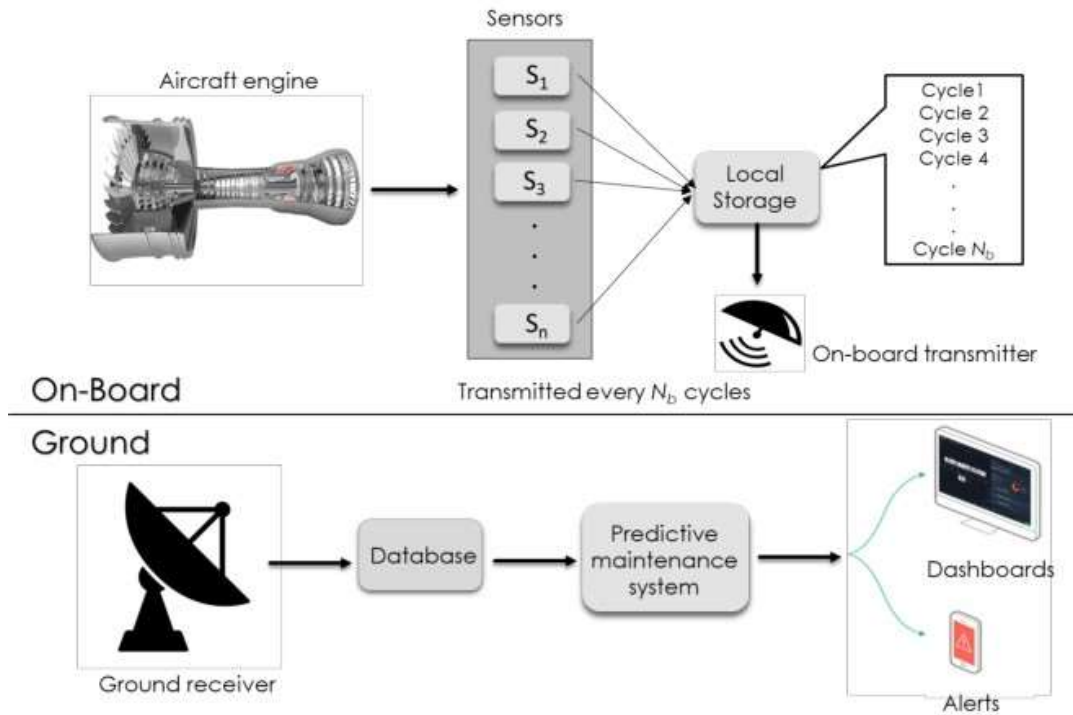
Name : B.Rukmini

Roll Number : 20BD1A054C

Class : CSE-C

Signature of faculty   **Signature of student**

ABSTRACT:

The concept of false data injection attack (FDIA) was introduced originally in the smart grid domain. While the term sounds common, it specifically means the case when an attacker compromises sensor readings in such tricky way that undetected errors are introduced into calculations of state variables and values. Due to the rapid growth of the Internet and associated complex adaptive systems, cyber attackers are interested in exploiting similar attacks in other application domains such as healthcare, finance, defense, governance, etc. In today's increasingly perilous cyber world of complex adaptive systems, FDIA has become one of the top-priority issues to deal with. It is a necessity today for greater awareness and better mechanism to counter such attack in the cyberspace. Hence, this work presents an overview of the attack, identifies the impact of FDIA in critical domains, and talks about the countermeasures. A taxonomy of the existing countermeasures to defend against FDIA is provided. Unlike other works, we propose some evaluation metrics for FDIA detection and also highlight the scarcity of benchmark datasets to validate the performance of FDIA detection techniques.

Domain : Cyber Security

Cybersecurity is the practice of protecting critical systems and sensitive information from digital attacks. Also known as information technology (IT)

security, cybersecurity measures are designed to combat threats against Networked systems and applications, whether those threats originate from inside or outside of an organization.

A strong cybersecurity strategy has layers of protection to defend against cyber crime, including cyber attacks that attempt to access, change, or destroy data; extort money from users or the organization; or aim to disrupt normal business operations. Countermeasures should address:

- **Critical infrastructure security** - Practices for protecting the computer systems, networks, and other assets that society relies upon for national security, economic health, and/or public safety. The National Institute of Standards and Technology (NIST) has created a cybersecurity framework to help organizations in this area, while the U.S. Department of Homeland Security (DHS) provides additional guidance.

- **Network security** - Security measures for protecting a computer network from intruders, including both wired and wireless (Wi-Fi) connections.

- **Application security** - Processes that help protect applications operating on-premises and in the cloud. Security should be built into applications at the design stage, with considerations for how data is handled, user authentication, etc.

- **Cloud security** - Specifically, true confidential computing that encrypts cloud data at rest (in storage), in motion (as it travels to, from and within the cloud) and in use (during processing) to support customer privacy, business requirements and regulatory compliance standards.

- **Information security** - Data protection measures, such as the General Data Protection Regulation or GDPR, that secure your most sensitive data from unauthorized access, exposure, or theft.

- **End-user education** - Building security awareness across the organization to strengthen endpoint security. For example, users can be trained to delete suspicious email attachments, avoid using unknown USB devices, etc.

- **Disaster recovery/business continuity planning** - Tools and procedures for responding to unplanned events, such as natural disasters, power outages, or cybersecurity incidents, with minimal disruption to key operations.

- **Storage security** – IBM Flash system delivers rock solid data resilience with numerous safeguards. This includes encryption and immutable and isolated data

copies. These remain in the same pool so they can quickly be restored to support recovery, minimizing the impact of a cyber attack.

- **Mobile security** – IBM security enables you to manage and secure your mobile workforce with app security, container app security and secure mobile mail.

Deep learning:

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of **labelled data**. For example, driverless car development requires millions of images and thousands of hours of video.
2. Deep learning requires substantial **computing power**. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Internet of things:

The **Internet of things** (**IoT**) describes physical objects (or groups of such objects) with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Internet of things has been considered

a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable.

The field has evolved due to the convergence of multiple technologies, including ubiquitous computing, commodity sensors, increasingly powerful embedded systems, as well as machine learning. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), independently and collectively enable the Internet of things. In the consumer market, IoT technology is most synomyns with products pertaining to the concept of the "smart home", including devices and appliances (such as lighting fixtures, thermostats home security systems, cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smart phones and smart speakers. IoT is also used in health care systems.

There are a number of concerns about the risks in the growth of IoT technologies and products, especially in the areas of privacy and security, and consequently, industry and governmental moves to address these concerns have begun, including the development of international and local standards, guidelines, and regulatory frameworks.

Machine learning:

Machine Learning tutorial provides basic and advanced concepts of machine learning. Our machine learning tutorial is designed for students and working professionals.

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more. This machine learning tutorial gives you an introduction to machine learning along with the wide range of machine learning techniques such as Supervised, Unsupervised, and Reinforcement learning. You will learn about regression and classification models, clustering methods, hidden Markov models, and various sequential models.

Project Description Or Problem Statement: False Data Injection encompasses a class of malicious data attacks that target critical infrastructures controlled by Cyber-Physical Information Systems. FDIA strategies involve the attacker compromising sensor readings, so undetected corrupt data is included in calculating values and variables used to define the system state.

Technical Description :

1.   we use three state-of-the-art DL algorithms, specifically, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN) for predicting the Remaining Useful Life (RUL) of a turbofan engine using NASA's C-MAPSS dataset.

2.   The obtained results show that the GRU-based PdM model outperforms some of the recent literature on RUL prediction using the C-MAPSS dataset.

3.   Afterward, we model two different types of false data injection attacks (FDIA) on turbofan engine sensor data and evaluate their impact on CNN, LSTM, and GRU-based PdM systems.

4.   The obtained results demonstrate that FDI attacks on even a few IoT sensors can strongly defect the RUL prediction.

5.   Our experiments reveal an interesting relationship between the accuracy, resiliency and sequence length for the GRU-based PdM models.

False Data Injection Attack:

False Data Injection encompasses a class of malicious data attacks that target critical infrastructures controlled by [Cyber-Physical Information Systems](#). FDIA strategies involve the attacker compromising sensor readings, so undetected corrupt data is included in calculating values and variables used to define the system state.

Attacks in power systems running on smart grids result in the loss of management for control devices, resulting in operational overheads and severe power outages. Successful, false data injection attacks also result in the corruption of transactions on power systems, leading to revenue loss. By acquiring knowledge of the circuit fault condition, FDIA attacks also lead to load distribution dysfunction that causes intermittent faults and power imbalance between demand and supply.

Wireless IoT device communication has found favorable adoption in major industries, such as air travel, autonomous vehicles, and healthcare. In such instances, false data injection vulnerabilities introduce the risk of privacy leakage, as these devices mostly process sensitive, personally identifying information. Successful data injection into these systems often results in computational overhead as unknown elements complicate the mathematical model used in decision-making based on IoT sensor input.

Attackers also target injection vulnerabilities in communication networks to alter data transmitted within the control device to hybrid IoT networks. A successful attack's severity depends on the injection attack type, the target system, and the deviation between original measurements and the altered data set.

Examples of FDIA attacks:

False Data Injection Attacks are categorized according to the subsystem affected and the level of access the attacker can obtain. The following section highlights the common forms of FDIA attacks and recently orchestrated real-world exploits as examples.

Types of FDIA attacks:

Depending on the level of access adversaries possess to the power systems, FDIA attacks can be classified into:

Internal attacks

This form of FDIA attack is carried out by those adversaries who possess precise knowledge of the system's bilinear pairing operations. An attacker also accurately understands the power system's network topology, capacity, cost function, and standard measurements of the target system. In most cases, the adversary is an internal threat, such as a disgruntled employee or a malicious

actor. He can access the power system's historical load data and control devices.

External attacks

Adversaries carry out this False Data Injection attack with incomplete information about the power network. As a result, the attacker relies on weaknesses in the physical network's security model to eavesdrop, replay and inject false data into the smart grid. One common approach to effecting such attacks is to target vulnerabilities in input validation and transport layer security for delivering false data through techniques such as [code injection](#) and [cross-site request forgery](#).

Real time examples of FDIA attacks:

Cyber attacks targeting IoT device communication networks that support modern power systems, patient monitoring devices, defense systems, and other smart grid systems have risen with the changing threat landscape.

Some successfully orchestrated FDIA cyber attacks from the recent past include:

December 2015 Ukraine Blackout

This is one of the first publicly acknowledged cyber attacks on power system automation software. On 23rd December 2015, attackers were able to hack Ukraine's power grid system and use spear-phishing techniques to install malware that led to a blackout affecting over 200,000 consumers.

Stuxnet Worm on Iranian Nuclear Power Stations

[Stuxnet](#) is a computer worm that targets programmable logic controllers used to automate industrial processes and power systems. The worm has been under development since the mid-2000s and usually targets computers that use Windows OS and run the Siemens Step 7 real-time data transmission software.

Maroochy Water System Attack

Considered one of the most infamous internal attacks, this attack was carried out by a disgruntled employee of a radio-controlled sewage equipment installation company.

2010 Stuxnet Attack on WinCC SCADA Software in Germany

The Stuxnet virus was detected in fifteen power, chemical, and industrial control plants in Germany that use SCADA and Siemens software.
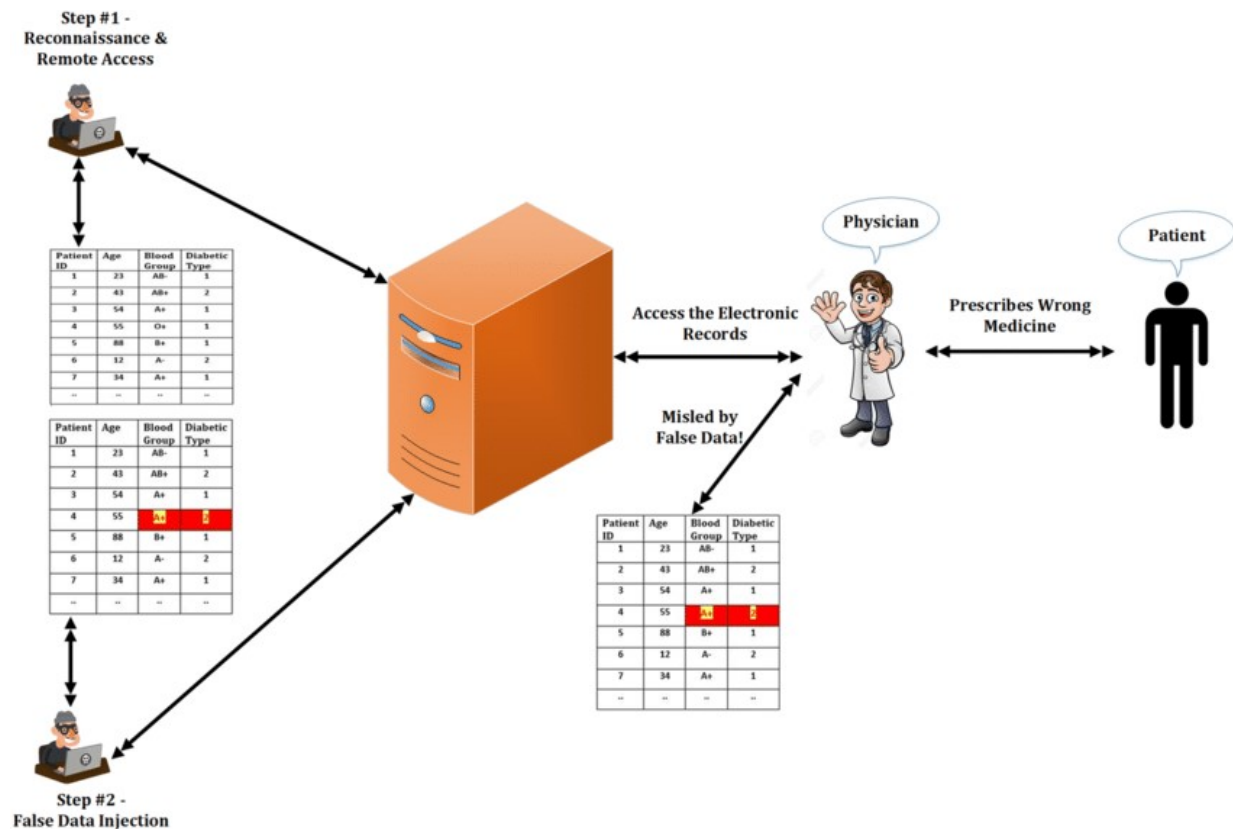
Prevention of FDIA attacks:

Some fundamental models/methods used to prevent FDIA attacks include:

- Deep Learning Techniques
- Kullback-Leibler Distance
- Spatio-Temporal Correlations
- Use of the Blockchain

We use the deep learning techniques to prevent FDIA attacks

FDIA Architecture:

Dataset:

- Data collection

The data collection process involves the selection of quality data for analysis. Here we used C-MAPSS dataset taken from turbon for machine learning implementation. The job of a data analyst is to find ways and sources of collecting relevant and comprehensive data, interpreting it, and analysing results with the help of statistical techniques.

- Data visualization

A large amount of information represented in graphic form is easier to understand and analyze. Some companies specify that a data analyst must know how to create slides, diagrams, charts, and templates. In our approach, the accuracy of data are shown as data visualization part.

- Data pre-processing

The purpose of pre-processing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- Data splitting

A dataset used for machine learning should be partitioned into three subsets — training, test, and validation sets.

Training set: A data scientist uses a training set to train a model and define its optimal parameters it has to learn from data.

Test set: A test set is needed for an evaluation of the trained model and its capability for generalization. The latter means a model's ability to identify patterns in new unseen data after having been trained over a training data. It's crucial to use different subsets for training and testing to avoid model overfitting, which is the incapacity for generalization we mentioned above.

- Model Training

After a data scientist has preprocessed the collected data and split it into train and test can proceed with a model training. This process entails "feeding" the algorithm with training data. An algorithm will process data and output a model that is able to find a target value (attribute) in new data an answer you want to get with predictive analysis. The purpose of model training is to develop a model.
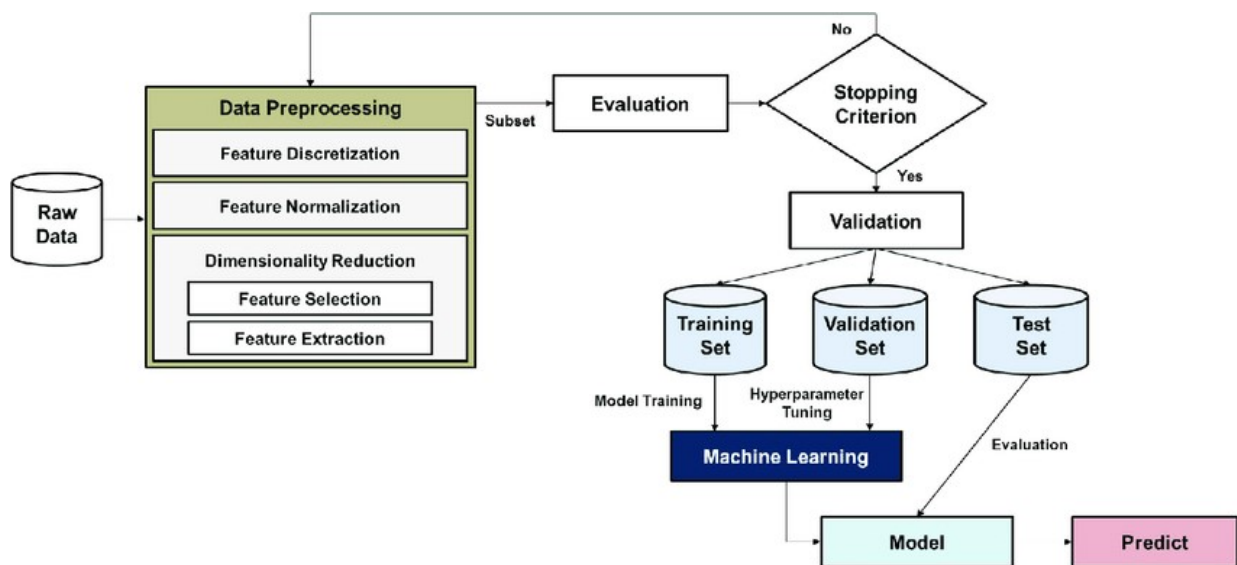
The project consists of C-MAPSS dataset.

To evaluate the performance of the CNN, LSTM, and GRU DL algorithms, we use a well-known dataset, NASA's turbofan engine degradation simulation dataset C-MAPSS (Commercial Modular Aero-Propulsion System Simulation). This dataset includes 21 sensor data with different number of operating conditions and fault conditions . In this dataset, there are four sub-datasets (FD001-04). Every subset has training data and test data. The test data has run to failure data from several engines of the same type. Each row in test data is a time cycle which can be defined as an hour of operation. A time cycle has 26 columns where the

1st column represents the engine ID, and the 2nd column represents the current operational cycle number. The columns from 3 to 5 represent the three operational settings and columns from 6-26 represent the 21 sensor values. The time-series data terminates only when a fault is encountered. For example, an engine with ID 1 has 192 time cycles of data, which means the engine has developed a fault at the 192nd time cycle. The test data contains data only for some time cycles as our goal is to estimate the remaining operational time cycles before a fault.

Models: we see what type of models are like CNN, LSTM, GRU.

We use the C-MAPSS(Commercial Modular Aero-Propulsion System Simulation) dataset . At first, to build an accurate predictive model, we train the Long Short-Term Memory (LSTM), Gated recurrent unit (GRU), and Convolutional neural network (CNN) algorithms using the C-MAPSS dataset. We evaluate these three predictive models, and the obtained results show that the GRUbased model predicts the Remaining Useful Life (RUL)2 most accurately. The obtained results from the GRU-based model outperforms the recent works that use DL for RUL prediction using the C-MAPSS dataset in (by predicting RUL 1.3-1.9 times more accurately). Afterward, we model two types of false data injection attacks (FDIA) on the C-MAPSS dataset and evaluate their impact on CNN, LSTM, and GRU-based PdM models. To be more realistic, we model attack only on 3 sensors among the 21 sensors in the turbofan engine. The obtained results show that all the PdM models are greatly defected by the FDIA even if only 3 out of the 21 sensors are attacked. However, the GRU-based PdM model is comparatively more accurate and resilient to FDIA when compared to the other evaluated PdM models. In terms of sensitivity, we also explore that CNN is way more sensitive to FDIAs when compared to the LSTM and GRU. This is indeed an important observation.

1) a popular turbofan engine degradation dataset published by NASA's Prognostics Center of Excellence (PCoE).
2) Remaining useful life (RUL) is the length of time a machine is likely to operate before it requires repair or replacement. since CNN-based techniques are quite popular in asset maintenance [23]–[25] and our results indicate that special measures should be taken for designing a CNN-based PdM. Afterward, we analyze the GRU-based PdM model using four different sequence lengths. The obtained results show an interesting relationship between the accuracy, the resiliency and the sequence length

of the models. To the best of our knowledge, this is the first work that demonstrates the effects of IoT sensor attacks on a deep learning-enabled PdM system. Paper organization: The rest of the paper is organized as follows. Section II briefly discusses the Engine Health Monitoring (EHM) systems and Predictive Maintenance (PdM). Section III introduces the DL algorithms used in this paper: LSTM, GRU, and CNN, and also describes NASA's C-MAPSS dataset used for our experiment. Section IV describes the modeling of FDIA in detail. Section V compares the performance of CNN, LSTM, and GRU in predicting RUL and analyses the impact on RUL prediction using CNN, LSTM, and GRU after both continuous and interim FDI.

- **CNN :** User need to insert a image of leaf captured here to get to know about the disease of that particular leaf.

Convolutional Neural Networks are a special type of feed-forward artificial neural network in which the connectivity pattern between its neuron is inspired by the visual cortex. The visual cortex encompasses a small region of cells that are region sensitive to visual fields. In case some certain orientation edges are present then only some individual neuronal cells get fired inside the brain such as some neurons responds as and when they get exposed to the vertical edges, however some responds when they are shown to horizontal or diagonal edges, which is nothing but the motivation behind Convolutional Neural Networks.

The Convolutional Neural Networks, which are also called as covnets, are nothing but neural networks, sharing their parameters. Suppose that there is an image, which is embodied as a cuboid, such that it encompasses length, width, and height.

Basically, a Convolutional Neural Network consists of adding an extra layer, which is called convolutional that gives an eye to the Artificial Intelligence or Deep Learning model because with the help of it we can easily take a 3D frame or image as an input as opposed to our previous artificial neural network that could only take an input vector containing some features as information.

In CNN, we will start with importing the libraries, data preprocessing followed by building a CNN, training the CNN and lastly, we will make a single prediction. All the steps will be carried out in the same way as we did in ANN, the only

difference is that now we are not pre-processing the classic dataset, but some images, which is why the data preprocessing is different and will consist of doing two steps, i.e., in the first, we will pre-process the training set and then will pre-process the test set.

In the second part, we will build the whole architecture of CNN. We will initialize the CNN as a sequence of layers, and then we will add the convolution layer followed by adding the max-pooling layer. Then we will add the second convolutional layer to make it a deep neural network as opposed to a shallow neural network. Next, we will proceed to the flattening layer to flatten the result of all the convolutions and pooling into a one-dimensional vector, which will become the input of a fully connected neural network. Finally, we will connect all this to the output layer.
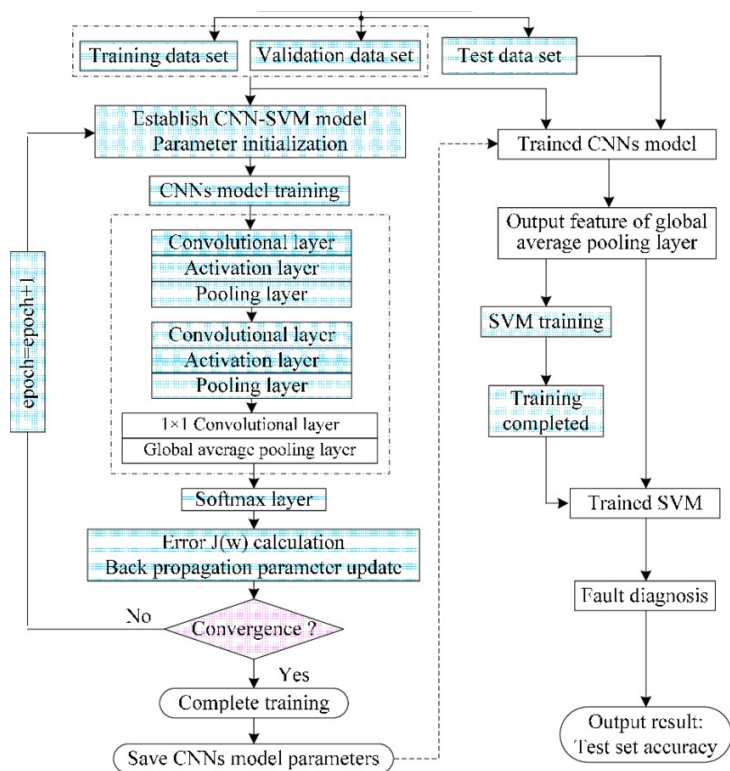
- **LSTM:** An LSTM is a special kind of Recursive Neural Network (RNN), capable of learning long-term dependencies. LSTM is explicitly designed

to avoid long term dependency problems, which is prevalent in RNN. It has achieved great praise in the field of machine learning and speech recognition. Some of the neural networks have a dependency problem, but an LSTM can overcome the problem of dependency by controlling the flow of information using input, output and forget gate. The input gate controls the flow of input activation into the memory cell. The output gate controls the output flow of cell activation into the rest of the network. Suppose that training data has N equipment of the same make and type that provide failure data, and each equipment provides set multivariate time-series data from the sensors of the equipment. Also, assume that there are r sensors of the same type on each equipment. Then data collected from each equipment can be represented in a matrix form X_n = $[x_1, x_2, ..., x_t, ..., x_{T_n}] \in R^{r \times T_n}$ (n = 1, ..., N) where T_n is time of the failure and at time t the r-dimensional vector of sensor measurements is $x_t = [s_1^t, ..., s_r^t] \in R^{r \times 1}$, t = 1, 2, ..., T_n. The data of each equipment in X_n is fed to LSTM network and the network learns how to model the whole sequence with respect to target RUL. At time t, LSTM network takes r-dimensional sensor data x_t and gives predicted RUL_t. Let the LSTM cell has q nodes, then $c_t \in R^{q \times 1}$ is output of cell state, $h_t \in R^{q \times 1}$ is output of LSTM cell, $o_t \in R^{q \times 1}$ is output gate, $i_t \in R^{q \times 1}$ is input gate, and $f_t \in R^{q \times 1}$ is forget gate at time t. At time t−1, the output h_{t-1}, and hidden state c_{t-1} will serve as input to LSTM cell at time t. The input x_t is fed as input to the cell. In LSTM, the normalized data are calculated using the following equations: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$, (1) $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$, (2) $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$, (3) $ec_t = act(W_c \cdot [h_{t-1}, x_t] + b_c)$, (4) $c_t = f_t * c_{t-1} + i_t * ec_t$, (5) $h_t = o_t * act(c_t)$, (6) Where σ is the sigmoid layer. c_t and ec_t are each internal memory cell and temporary value to make a new internal memory cell at time t. $*$ is element-wise multiplication of two vectors.
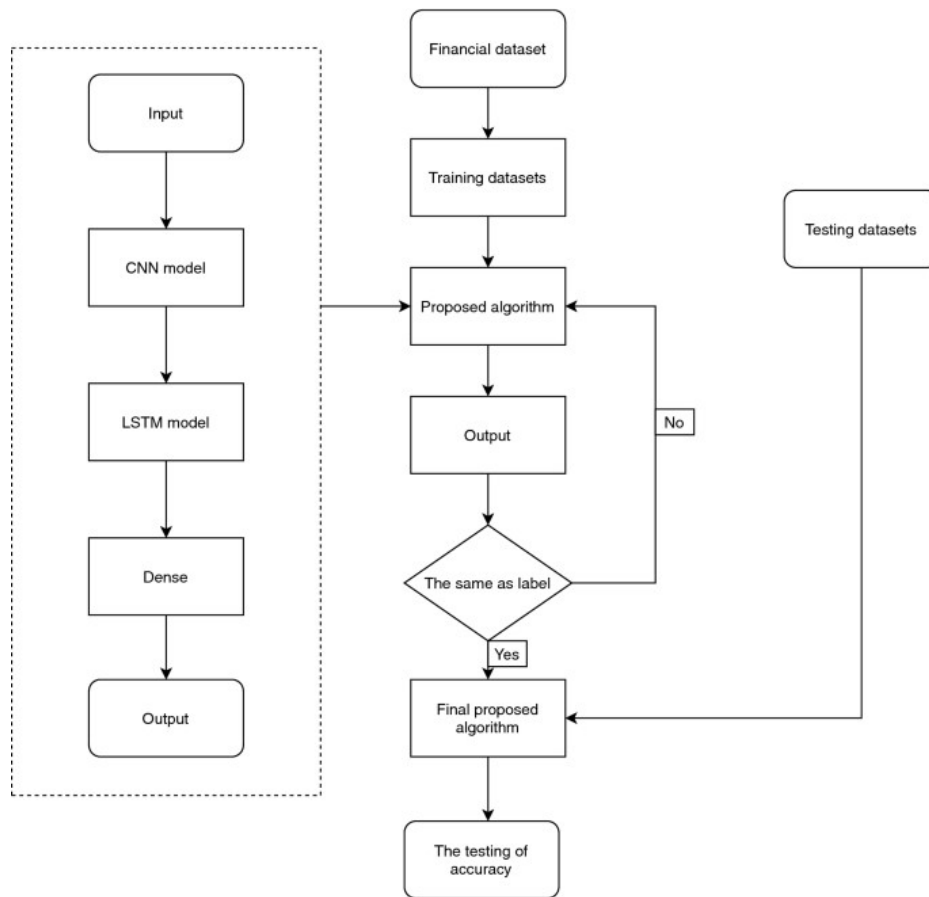
- **GRU :** To solve the Vanishing-Exploding gradients problem often encountered during the operation of a basic Recurrent Neural Network, many variations were developed. One of the most famous variations is the Long Short Term Memory Network (LSTM). One of the lesser-known but equally effective variations is the Gated Recurrent Unit (GRU).
  The GRU was proposed by Cho et al. It operates using a reset gate and an update gate. The GRU is an improved version of standard recurrent neural networks. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information, however, without having a separate memory cell. GRU's performance on certain tasks of polyphonic music modelling and speech signal modelling was found to be similar to that of LSTM. GRUs have been shown to exhibit even better performance on certain smaller datasets. The memory block of GRU is simpler than that of LSTM. The forget, input and output gates are replaced with an update and a reset gate. Also, GRU combines the hidden state and the internal memory cell. In GRU, the normalized data are calculated using the following equations:
  $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$,
  $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$,
  $eh_t = act(W \cdot [r_t * h_{t-1}, x_t] + b_h)$,
  $h_t = (1 - z_t) * h_{t-1} + z_t * eh_t$,

where zt and rt are the update gate and reset gate at time t, respectively. eht is a temporary value to make new hidden state at time t.

Unlike LSTM, it consists of only three gates and does not maintain an Internal Cell State. The information which is stored in the Internal Cell State in an LSTM recurrent unit is incorporated into the hidden state of the Gated Recurrent Unit. This collective information is passed onto the next Gated Recurrent Unit. The different gates of a GRU are as described below:-

Update Gate(z): It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit.

Reset Gate(r): It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.

Current Memory Gate: It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the future.

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.
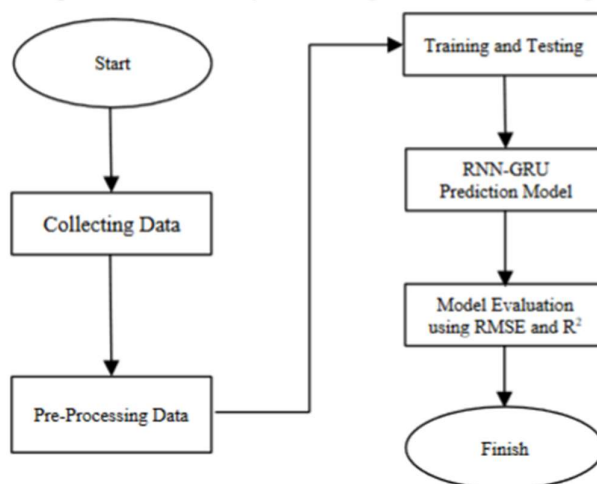
Statistics:

CNN:

RMSE value for CNN Without False Data:7.5

RMSE value for CNN With Biased RMSE: 85

RMSE value for CNN With Random RMSE:197

LSTM:

RMSE value for LSTM Without False Data:6.6

RMSE value for LSTM With Biased RMSE: 31

RMSE value for LSTM With Random RMSE: 47

GRU:

RMSE value for GRU Without False Data:6.6

RMSE value for GRU With Biased RMSE: 41

RMSE value for GRU With Random RMSE: 67

Hyperparameter settings

Model	Hidden neuron	Dropout	Batch size	Epochs	Act. func.
CNN	64	0.2	200	100	ReLu
LSTM	100	0.2	200	100	Tanh
GRU	100	0.2	200	100	tanh

Discussion:

In this work, we first evaluate different deep learning (DL) algorithms on the C-MAPSS dataset and obtained results show a great prospect for deep learning in PdM. It is also observed that sequence length and network architecture are crucial in predicting accurate RUL. Our work shows that the GRU performed 1.3-1.9 times better than the recent works that use deep learning on the C-MAPSS dataset [20], [22], [47]. The impact analysis of FDIA on aircraft sensors in the CMAPSS dataset provides some interesting insights. We observe that CNN based PdM model is greatly affected by both random and biased FDIA. In the case of interim FDIA, CNN's random and biased RMSE are 18 and 11 times higher than the true RMSE, respectively, and in the case of continuous, the random and biased RMSE are 6 and 4 times higher than the true RMSE, respectively. We also observe that the GRU-based PdM model is more resilient to both random and biased in comparison with CNN and LSTM-based PdM models. Even though the GRU is least affected by both random and biased FDIA, their RMSE is 8 and 6 times higher than the true RMSE in the case of continuous FDIA, respectively. In the case of interim FDIA, the random and biased RMSE are 4 and 3 times higher

than the true RMSE, respectively, making it disastrous for the PdM system. This may result in the delay of timely maintenance for the aircraft engine and eventually result in engine failure at some point. Note, the attack signature of FDIA is very close to the original sensor output (within the boundary conditions of the sensor measurements) making it harder to be detected by common defense mechanisms in an engine health monitoring (EHM) system.

This paper compares the performance of LSTM, GRU, and CNN for RUL prediction using the C-MAPSS dataset, and explores the impacts of continuous and interim FDI attacks on these deep learning algorithms. We observe that the GRU is a better suited DL technique when compared to LSTM and CNN in terms of accuracy. The obtained results show that both continuous and interim FDIA have a substantial impact on the RUL prediction even if only a few IoT sensors are attacked. We also observed that the GRU-based PdM model is more resilient to FDIA, whereas CNN is dramatically sensitive to both continuous and interim FDIA. Finally, we explored that there exists a relationship between the accuracy and sequence length in the GRU-based PdM model which can serve as empirical guidance to the development of data-driven PdM systems. In the future, we plan to develop an end-to-end methodology for the detection and mitigation of sensor attacks in a PdM system.

MERN Stack :

MERN Stack is a Javascript Stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises of 4 technologies namely: [MongoDB](#), [Express](#), [React](#) and [Node.js.](#) It is designed to make the development process smoother and easier.

Each of these 4 powerful technologies provides an end-to-end framework for the developers to work in and each of these technologies play a big part in the development of web applications.

Advantages of using MERN STACK :

Several advantages of the MERN stack include numerous capabilities and plugins that operate efficiently with the backside development platform to construct fast web apps and APIs.

MERN Architecture
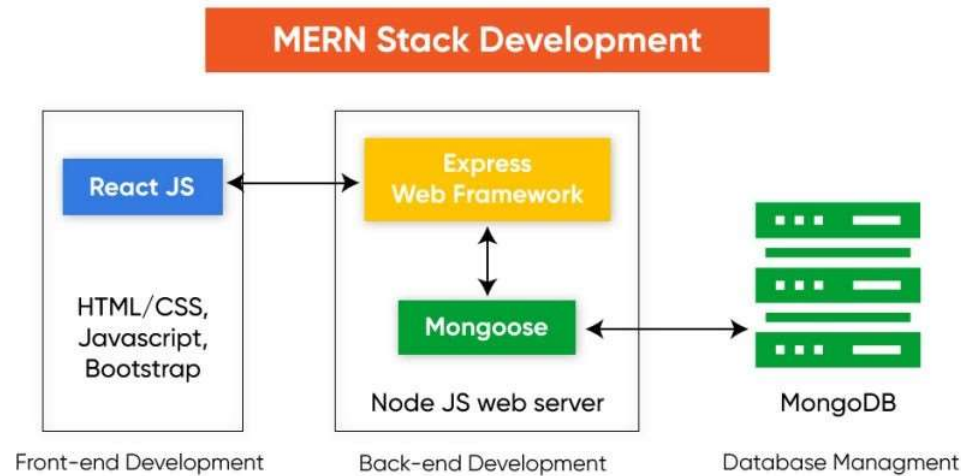
- **React.js**: The javascript React Javascript framework allows for the efficient creation of HTML-based user interfaces. In the MVC model, React serves as the View layer and provides declarative views. JavaScript, a full-featured scripting language, is used to create repeating DOM components and works well for SPAs Node. It stands for the JavaScript operating system. MERN's component employs automated programming built on Google's search V8 JavaScript engine. With an efficient database, you gain faster loading times and faster web programming.

- **No context changing**: For the whole program, JavaScript generates both client-side and server-side components; the web technology does not require response time and delivers efficient web apps.

- **MVC**: A significant feature of the MERN stack is that it provides an architecture that makes it easier for developers to construct online applications.

- **Entire**: Because there is no context switching, you get philosophically aligned and strong technologies that function tangibly together and efficiently handle client and virtual machine programming faster.

Front-End

Back-End
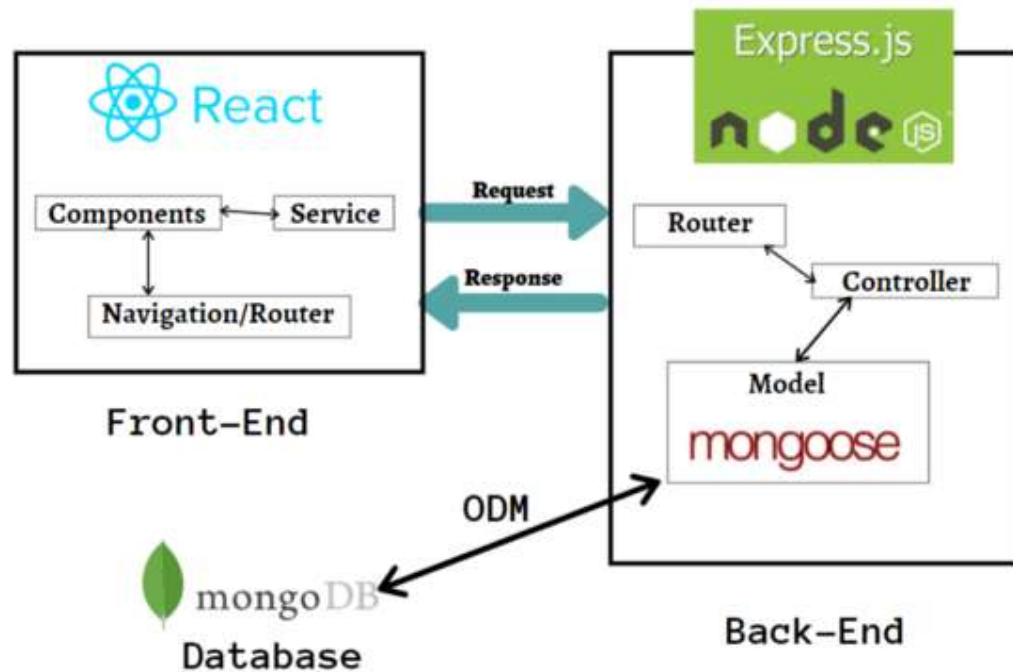
Database

Simple Training Curve: To get MERN stack benefits when designing web apps, professional developers need to be proficient in JS and JSON

We use the mongo db to connect the database.

SYSTEM WORK FLOW

Prerequisite:

Nodejs should be installed and react extensions in vs code

1.npx create-react-app app_name

2.create a backend folder in this folder and go into the created folder and run following commands for installations

3.npm init -y

4.npm start on project directory
At server port 3000 the application will run
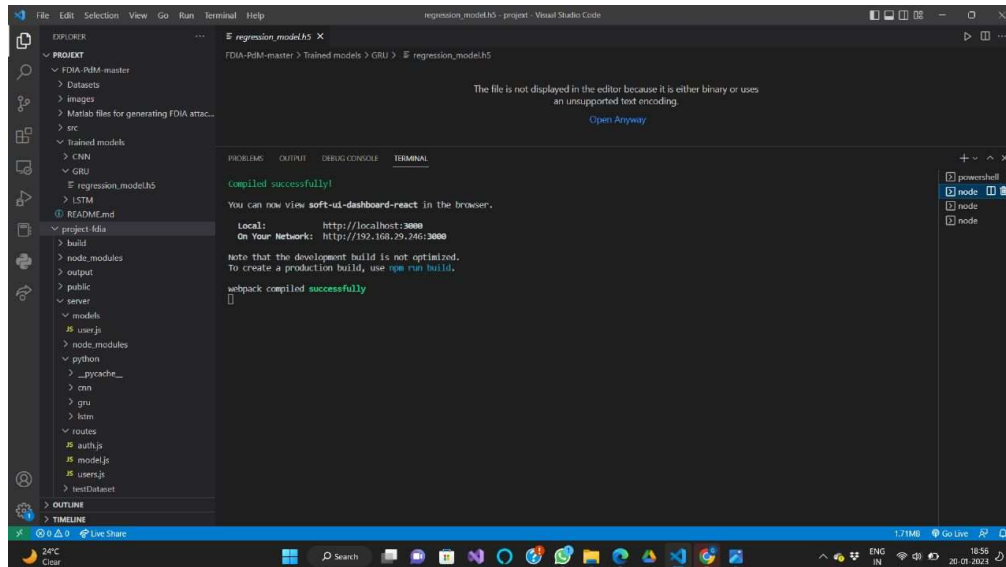
5) At the terminal we have start the server

To start type commands:

a) npm start

b) server@1.0.0.start

c) nodemon index.js

it connects the database successfully so that we can achieve the tables and graphs of predicted values
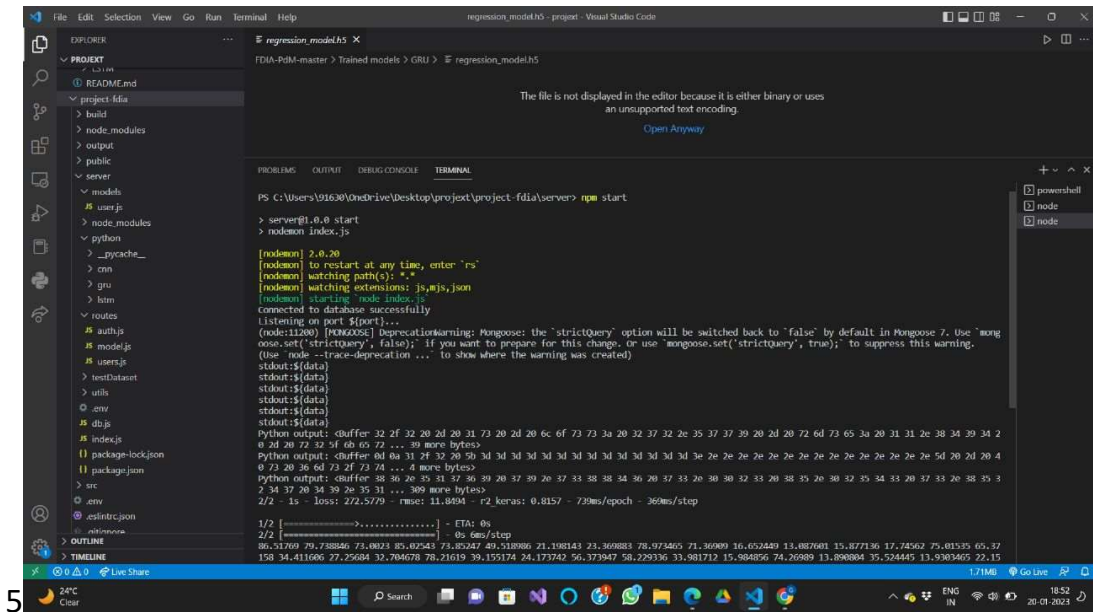
Home Screen : Starting page of the web app look as follows :
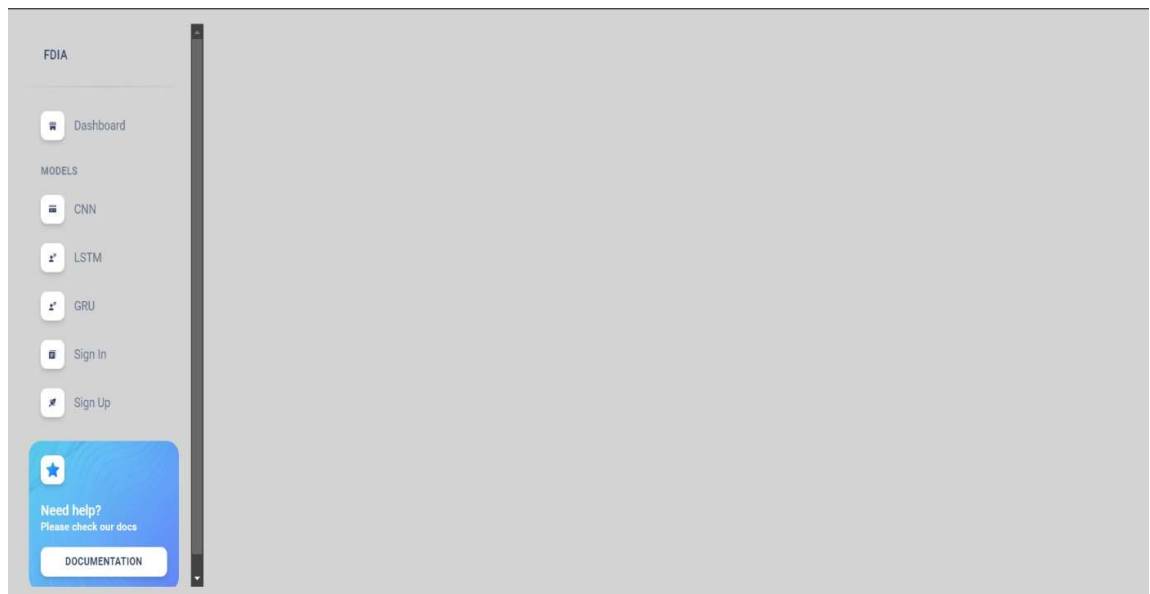
2) **Dashboard :**it describes what is FDIA and some examples of fdia and a reference paper..
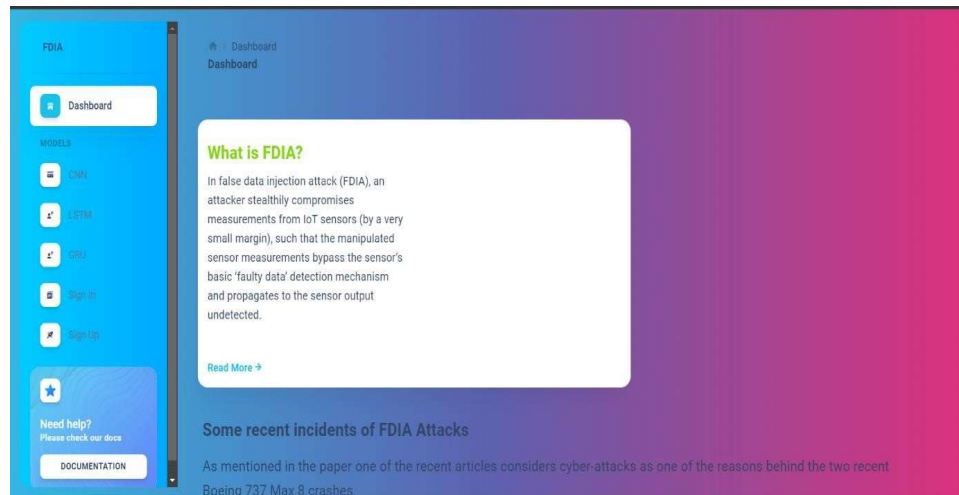
3)**Models:**

CNN: It gives explanation about the CNN(Convolutional neural networks) model.
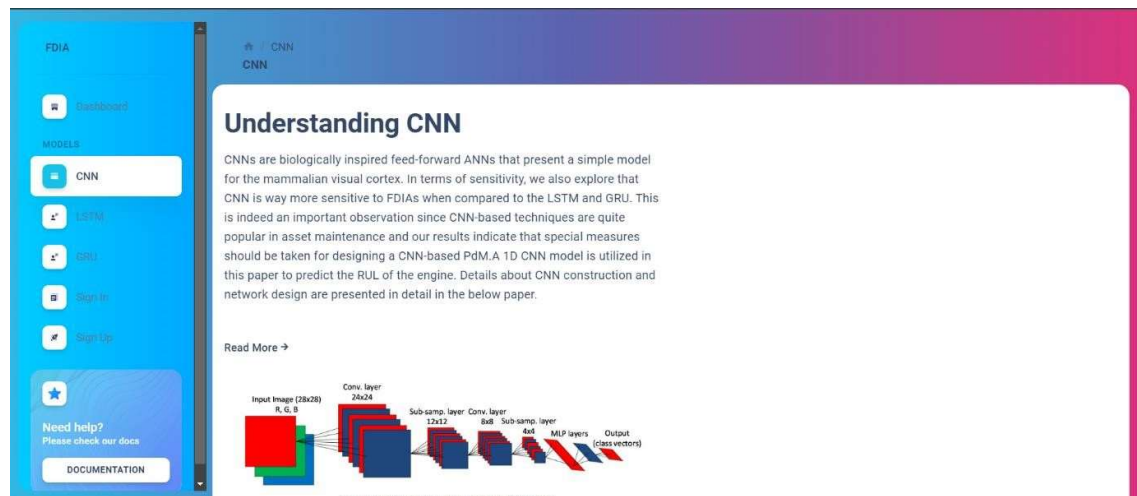
LSTM: It gives the information about the LSTM(Long short-term Memory) model.
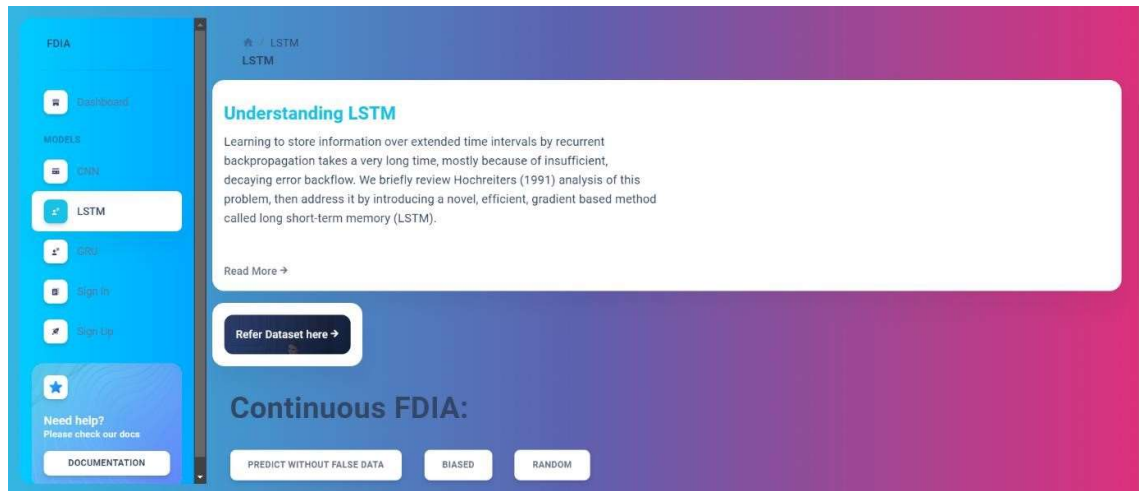
GRU: It gives the information about GRU(Gated recurrent unit) model.
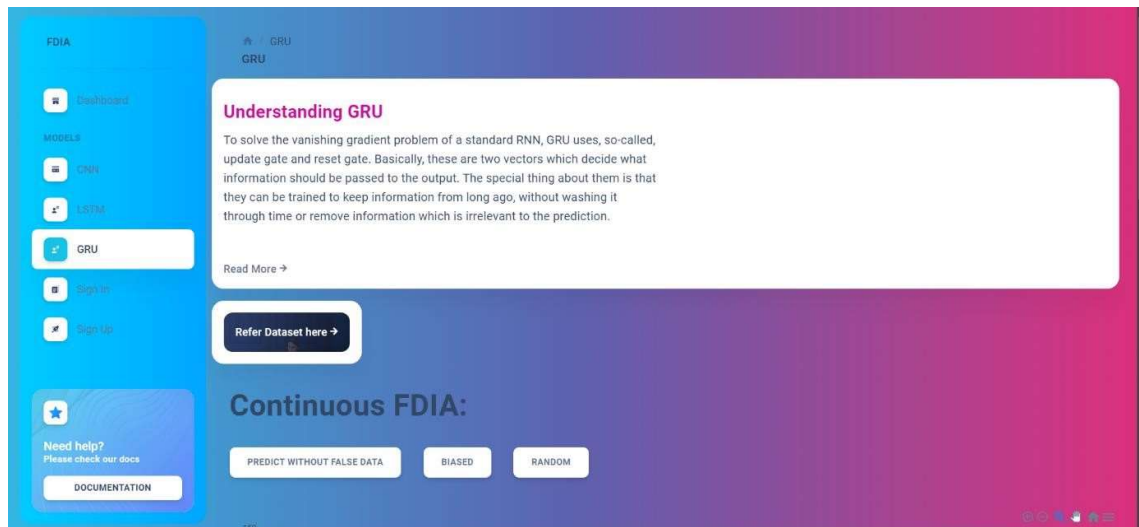
CNN with Prediction graph : It can viewed when the CNN is scrolled

It contains the graph of prediction of without false data and biased data and random data.

→ **truth - Blue - true value**

→ **predict - Green - without false data**

→ **biased - Yellow - with False Data**

→ **random - Red - with false data**

LSTM with Predicted Graph: It can viewed when the LSTM is scrolled

It contains the graph of prediction of without false data and biased data and random data.

→ **truth - Blue - true value**

→ **predict - Green - without false data**

→ **biased - Yellow - with False Data**

→ **random - Red - with false data**

GRU with Predicted Graph:

It can viewed when the GRU is scrolled

It contains the graph of prediction of without false data and biased data and random data.

→ **truth - Blue - true value**

→ **predict - Green - without false data**

→ **biased - Yellow - with False Data**

→ **random - Red - with false data**

CNN Comparison table :
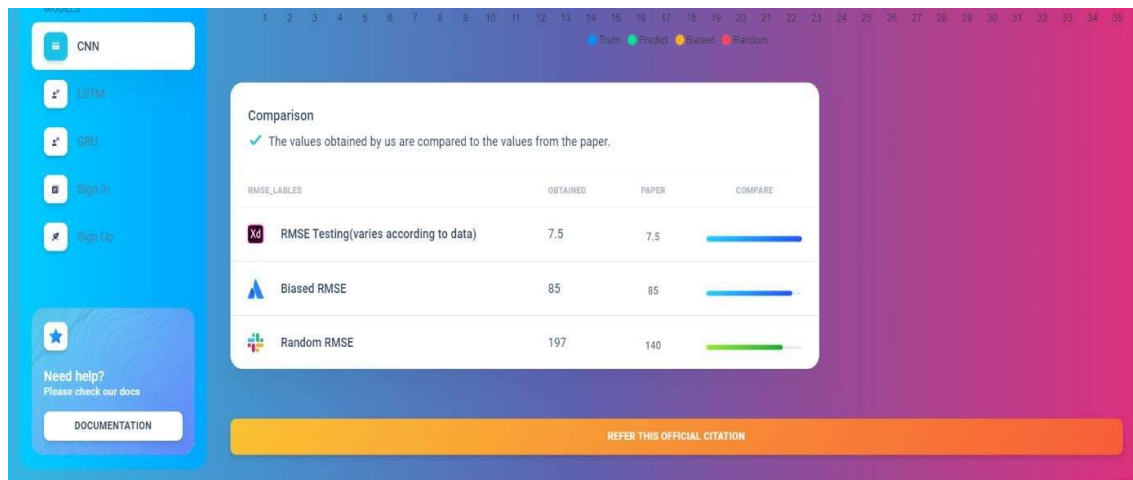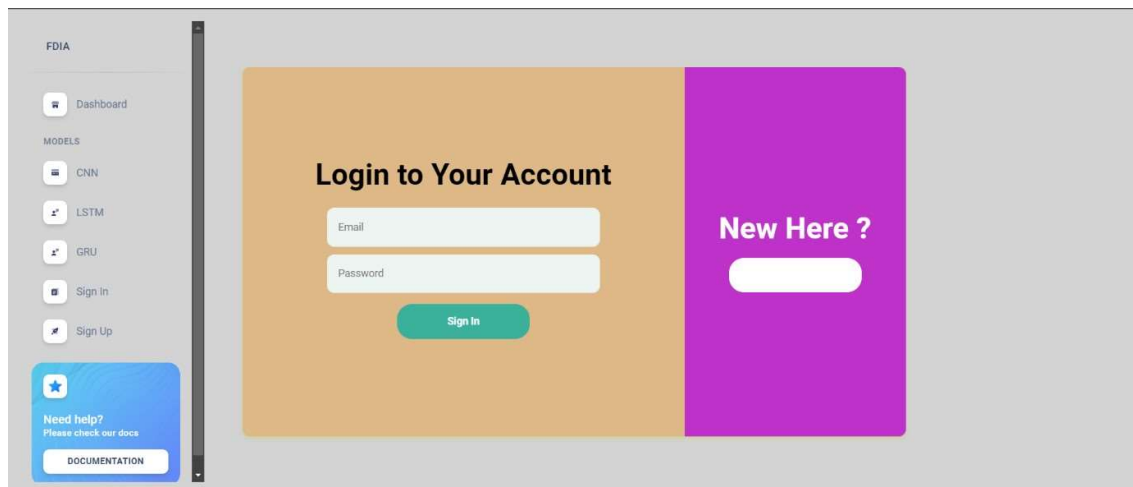
LSTM Comparison Table:

GRU Comparison Table:

Sign in: Given an access to login

Sign out: Given an access to create account data .