Design Credit Report

Academic Year - 2022/23

Project Under: Dr. Ravi Yadav

Problem Statement: **"Cyber threat** modeling **and machine learning method for detection"**

Team:

- Govind Mali (B21EE021)
- Kartik Sanjay Narkhede (B21EE041)
- Neel Anirudhha Barve (B21EE042)
- Samarth Sudhirkumar Bhalerao (B21EE060)

Problem Statement :
We are given the original problem statement to detect the attack on the Automatic Gain Control or AGC & to make the output unhampered; predict the continuous output signal.

About the provided MATLAB Model

The matlab model provided has been given with 4 input signal functions to create the the scenario of the cyber attack:

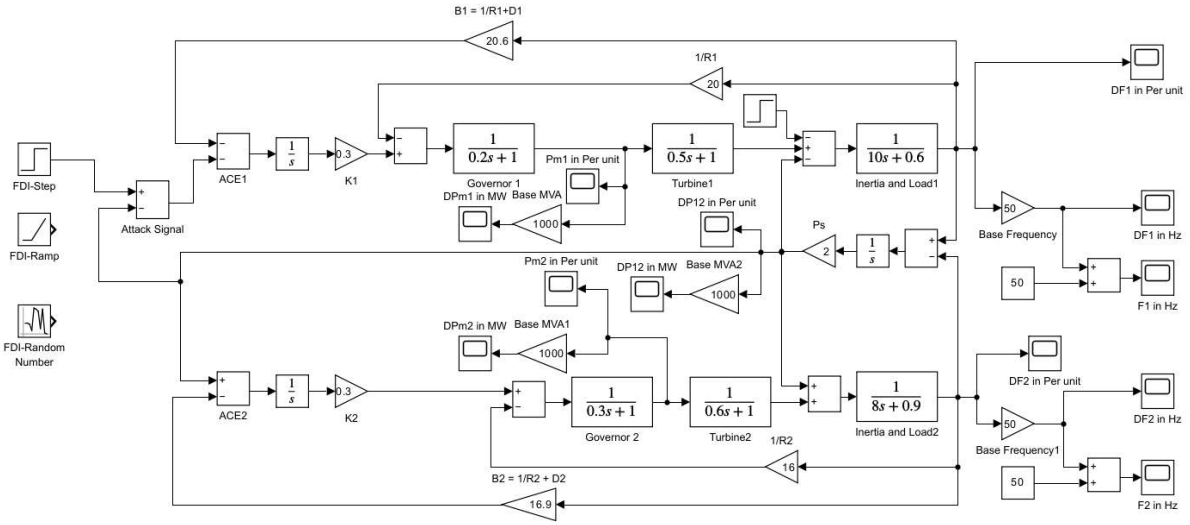
- Ramp signal function
- Random signal function
- Unit signal function
- Step signal function

Parameters that can be altered for making the cyber attack:

-The AGC (“automatic gain control”) , a closed-loop feedback regulating circuit in an amplifier or chain of amplifiers, the purpose of which is to maintain a suitable signal amplitude at its output, despite variation of the signal amplitude at the input.

-Now the constant parameters of AGC can be altered by any kind of cyber attack on the system and thus the output will be manipulated.

-The altered parameters are the input signal functions in the original signal changing the output signal.



Workflow :
We have broken up the original problem statement into two broad parts to work more efficiently.

Problem Statement 1 - We need to detect the attack scenario in the given system. We are provided with the continuous input from the system for the output values of frequency and power per unit.
Now we need to create the warning when the system is under attack so that the output can be passed over.

Solution Approach
Now to train the model and see whether the system is under attack, we are creating the time series data for attacked and normal situations by recording the outputs from the model.
We labeled the obtained data according to the type of attack as:
‘0’ - Normal situation
‘1’ - Step impulse signal attack
‘2’ - Ramp function signal attack
‘3’ - Impulse function signal attack
‘4’ - Random noise addition attack

CodeFile Path:
https://colab.research.google.com/drive/1CfAJlkpiURaIJRZmvNIXLnGLIGwA6thh?usp=sharing

Why we use RNN: Recurrent Neural Network (RNN) models are typically used for natural language processing (NLP) tasks that involve text classification, sentiment analysis, and speech recognition or time-series data .
RNNs are advantageous for time series sequential data because they are designed to work with sequences of data, where each element in the sequence is dependent on previous elements.

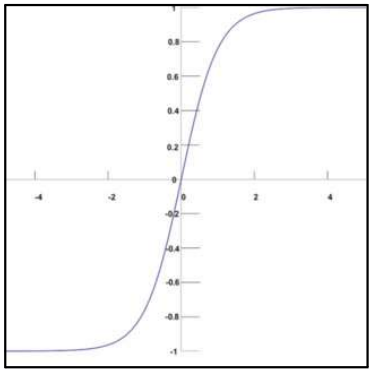
Traditional neural networks process each input independently, without considering the previous inputs.
However, in time series sequential data, the order of the data points matters, and the relationship between each data point and the previous ones can be crucial in understanding and predicting the data.

About Code

- Import the required libraries i.e. NumPy, Pandas, and TensorFlow. Then, we loaded the training and testing datasets using Pandas.

- Normalize the data using StandardScaler and encode the labels using LabelEncoder.

- Then, we define the RNN model using the Sequential model from Keras, and add an LSTM layer with 64 units followed by a Dense layer with softmax activation function.



Here we use Softmax as an activation function.

- Then we compiled the model with the categorical cross-entropy loss function, RMSprop optimizer, and accuracy metric.

  Categorical cross-entropy loss function:
  Used as loss function for multi-class classification problems.
  Used to measure the difference between the predicted probability distribution and the actual probability distribution of a set of classes.

    - H(P,Q)= -sum(x*P(x)*log(Q(x))). P(x): True probability distribution, Q(x): Predicted probability distribution. The lower it is, the better is the model prediction.
- Further we also find the confusion matrix and calculate evaluation metrics such as accuracy, precision, recall, and F1 score.

Outputs from machine learning model:

```
Confusion matrix:
[[50  5 14]
 [ 6 64  0]
 [ 1  0 69]]

Classification report:
              precision    recall  f1-score

     class_0       0.88      0.72      0.79
     class_1       0.93      0.91      0.92
     class_2       0.83      0.99      0.90

    accuracy                           0.88
   macro avg       0.88      0.87      0.87
weighted avg       0.88      0.88      0.87

Accuracy: 0.8756
Precision: 0.8787
Recall: 0.8749
F1 score: 0.8722
```

Problem Statement 2

The objective of this problem statement is to predict future signal values based on past signal output values. We are given a signal that can be affected by external changes in parameters, and our goal is to use machine learning models to make accurate predictions.

Solution

We will be continually providing the train data to the model created and just after the attack is detected, the output will be generated by the model based on the previous input data on which our model is trained and the continuous series of values will be predicted as output.
Thus we can provide the output created by our model replacing the output given by AGC so that the generated output is unaffected by the attack.

About Matlab code

'adam': This specifies the optimization algorithm used for training the network. In this case, the 'adam' algorithm is used, which is an adaptive learning rate optimization algorithm that is well-suited for training deep neural networks.

'MaxEpochs': This specifies the maximum number of epochs (i.e., full passes through the training data) that the network will be trained for. In this case, the maximum number of epochs is set to 250.

'Gradient Threshold': This specifies the maximum gradient value allowed during backpropagation. This is useful for preventing exploding gradients and unstable training. In this case, the maximum gradient value is set to 1.

'Learn Rate Schedule': This specifies the learning rate schedule used during training. A learning rate schedule adjusts the learning rate over time, which can be useful for improving the convergence and stability of the network. In this case, the learning rate schedule is set to 'piecewise', which means that the learning rate will be adjusted at specific intervals (as specified by the 'LearnRateDropPeriod' option).

'**Learn Rate DropFactor**': This specifies the factor by which the learning rate will be reduced when it is adjusted during training. In this case, the learning rate will be reduced by a factor of 0.2.

'**Verbose**': This specifies the level of verbosity during training. A value of 0 means that no information will be printed to the console during training

'Learn Rate Drop Period': This specifies the interval at which the learning rate will be adjusted during training. In this case, the learning rate will be adjusted every 125 epochs.

Code Description

The training data is standardized using the mean and std functions to ensure that all input values have similar scales.

The LSTM model is defined using the “LSTM“ Layer function. The model has one input feature, one output response, and 200 hidden units. The model architecture consists of a sequence input layer, a LSTM layer, a fully connected layer, and a regression layer.

The model is trained using the train Network function with the defined options for the optimizer, learning rate, and other training parameters.

The test data is standardized using the mean and standard deviation calculated from the training data.

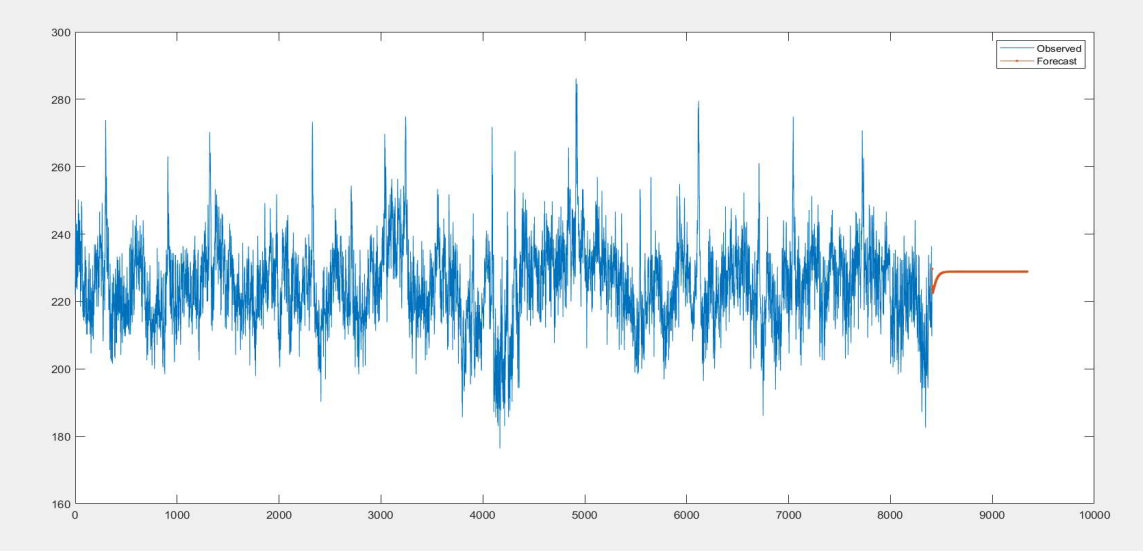
The trained model is used to make predictions on the test data. The predictAndUpdateState function is used to update the state of the network after each prediction.

The predicted values are converted back to the original scale using the mean and standard deviation of the training data.
The root-mean-squared error (RMSE) between the predicted and actual values is calculated.

Two figures are created to display the results. The first figure shows the observed and predicted values for the training and test sets. The second figure shows the predicted values, the actual values, and the difference between the two.

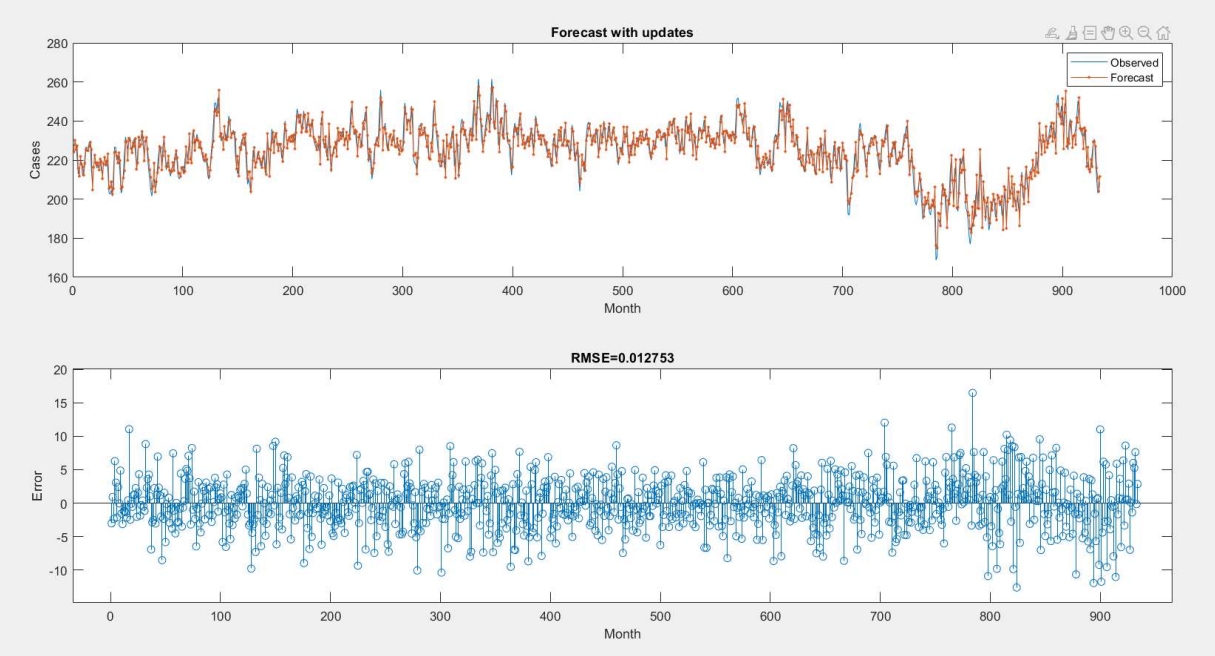
The LSTM model is updated with the test data, and the predictions are made again. The RMSE is calculated, and two figures are created to display the results.

Input Data Graph



https://drive.google.com/file/d/1D1FdMqRt_ZgTtqIcK4R91la0xtsnppeW/view?usp=share_link

Output of MATLAB Code



References:

1. https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmlayer.html
2. https://www.javatpoint.com/recurrent-neural-network-in-tensorflow
3. https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5#:~:text=As%20per%20our%20business%20requirement,use%20in%20last%20output%20layer%20.