Aim of this project is to detect conveyor belt faults using deep learning. Before understanding project methodology it is necessary to understand common faults that can arise in conveyor belts.

Mistracking of conveyor belts is one of the common faults observed. Conveyor belt is bound to track a specific path. When belt is not following that specified path is nothing but is tracking of that belt it will lead to the malfunctioning of entire plant as all the operations carried out on the objects placed on the conveyor belt are programmed to happen at specific time and if belt is not aligned with its track will leads to failure of chain of operations. This can even lead to shutdown of the plant so finding miss tracking faults is very important.

Belt slippers is another common issue observed in case of conveyor belt mechanism. if there is excessive tension or or a less tension on the belt will lead to slip of the belt. Overweight of the load, working in extreme cold or extreme hot temperature, configuring incorrect belt tension are some of the factors that will cause belt slippage. Conveyor belt works on Pulley system so the sleepy age of the belt causes the over usage of Pulleys and that leads to excess power consumption. Monitoring belt slippage also includes monitoring factors which can cause belt slippage such as overweight, belt tension. Slippage of the belt can cause damage to the belt as well as head pulleys so monitoring them and preventing breakdown is one of the important preventive measures.

Along with belt slippage material slippage is also one of the problems related to the conveyor belt system. causes for material slippage are like Less friction between material surface and the belt this can be because of ageing of conveyor belt. This will not cause any wear and Tear to the belt but it will disturb the processing line as well as miss the timeframes. Monitoring moving object velocity along with belt velocity can warn of these kinds of incidents.

In most cases conveyor belts are made using metallic material such as Steels that will make them long lasting and heavy duty. However, it also means that when the rollers on the conveyor belt seize up, they have an unfortunate tendency to develop sharp edges. Those sharp ages cause the mustracking of the belt. Also can damage the object on the belt.

Another kind of a fault which is observed commonly is motor failure which is driving the conveyor belt. There can be many reasons such as torque generated by the motor is not correct for the lubrication between the motor and the conveyor belt is not proper.

Problems with power supply to the motor and those all factors can be well monitored in advance so the focus of the project is to monitor various parameters related to the belt and go preventive for fault detection. Technologies such as deep learning will provide an appropriate solution for this problem. neural networks will process different input parameters for the conveyor belt and to predict the faults understanding basics of Neural network will help to understand project well.

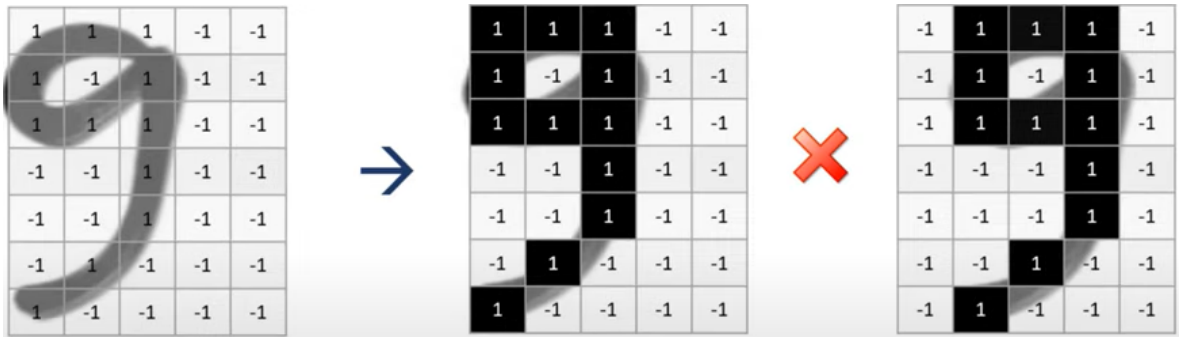
Convolution neural network and Rectified Linear Unit (ReLU) forms the base of the entire project. To understand Working of the project it is necessary to know how conversion neural networks identify objects in simple words, how they come to the conclusion from the incoming data and the training set and how a rectified linear unit is part of the entire process.

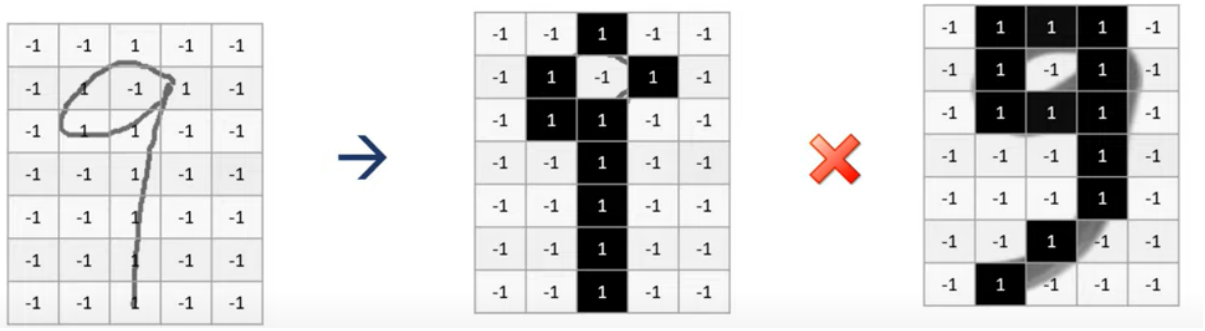
How does the Convolution neural network work?

A very simple explanation of convolutional neural will be to understand its working without using much mathematics. Simple example is to understand how computers recognize the handwritten digit. Way the computer looks at this is as a grid of numbers as shown below its grid of -1 and 1.
in reality it will use rgb numbers from 0 to 255.

the issue with this presentation is that this is too much hard-coded if there is a little shift in digit 9, for example 9 here was in the middle but in this case it is in the left and the representation of numbers just changes it doesn't matches match with original number

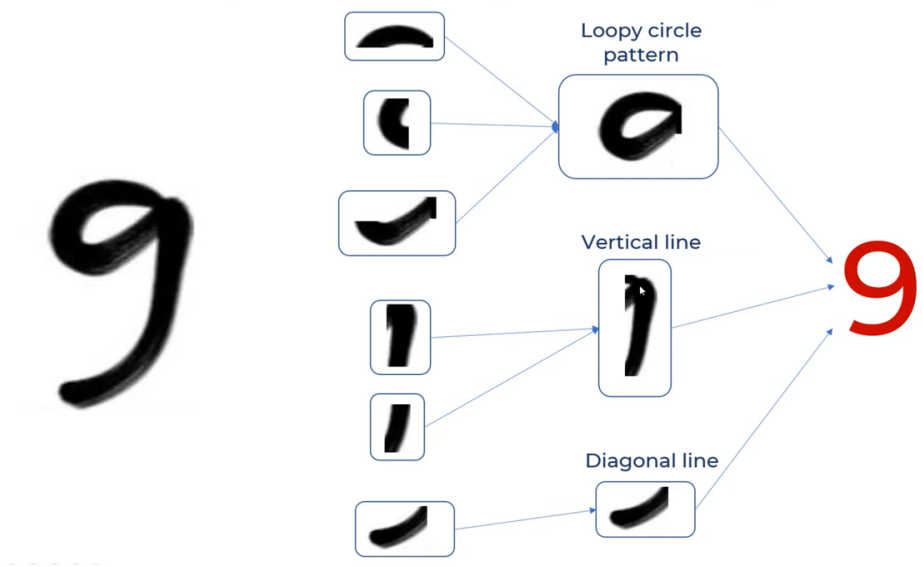
grid and computer will not be able to recognize that this is number nine there could be a variation since it is a handwritten digit there could be variation in writing style, so it will not be able to match it with the original grid.



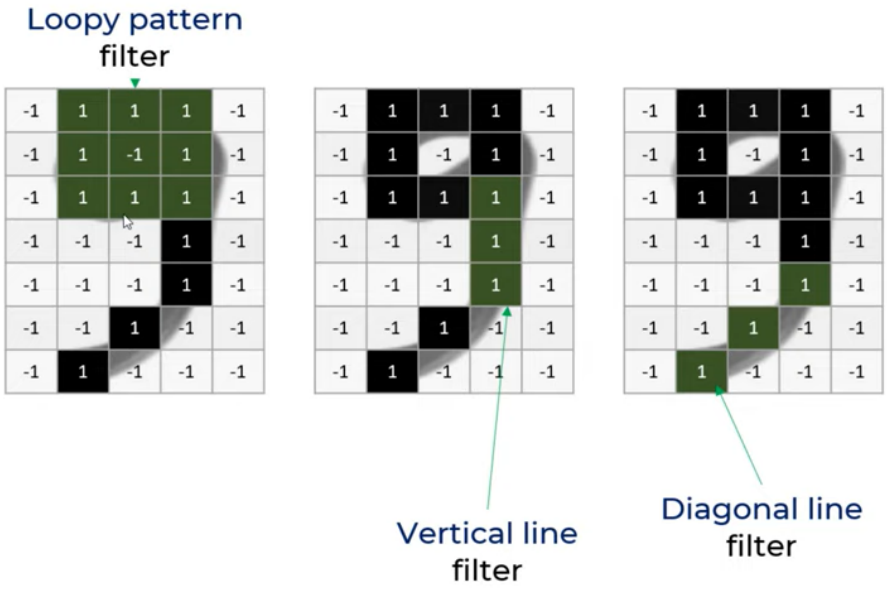


There comes to the role of artificial neural network

Before computers, how humans interpret it is a digit 9. As shown below the human eye will look for three patterns to identify it is 9. And they are a head of digit nine, in the middle there is a vertical line and at the bottom there is a diagonal line

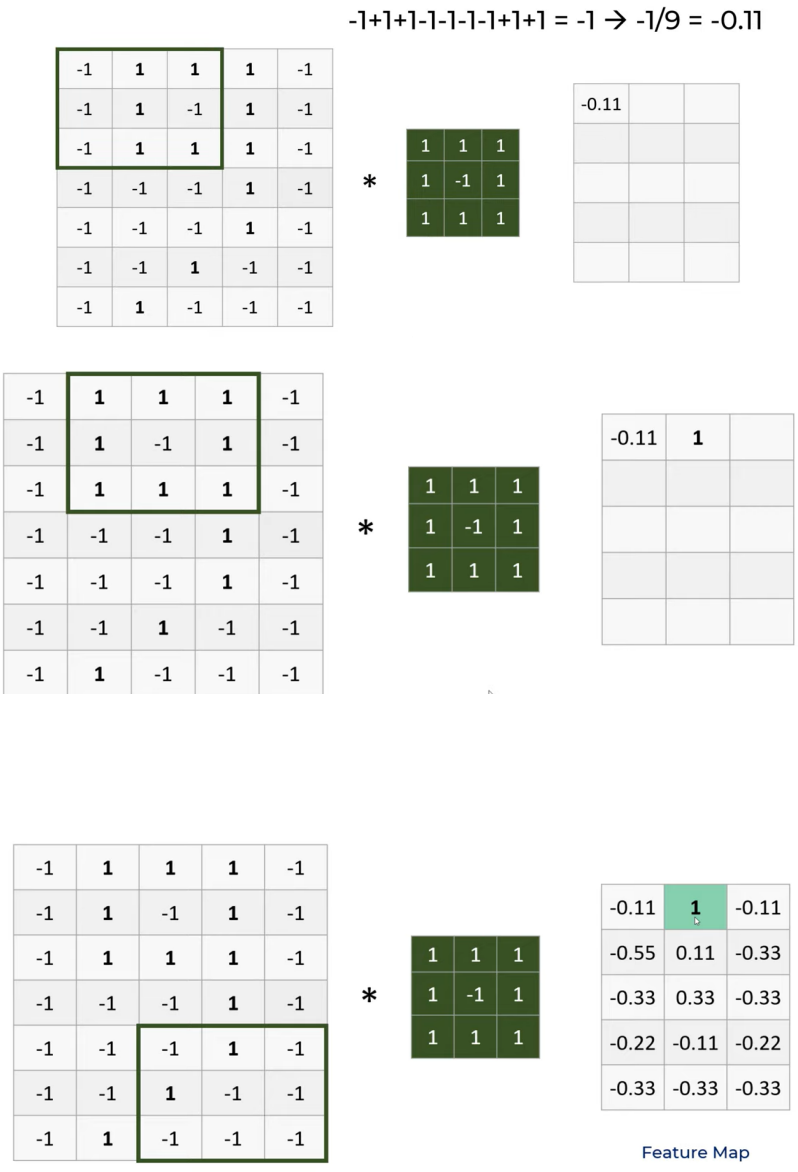


Now how can computers recognize these tiny features, Computers use the concept of filter. In case of nine There are three filters the first one is the head which is a loopy circle pattern in the middle there is a vertical line and in the end there is a diagonal filter.



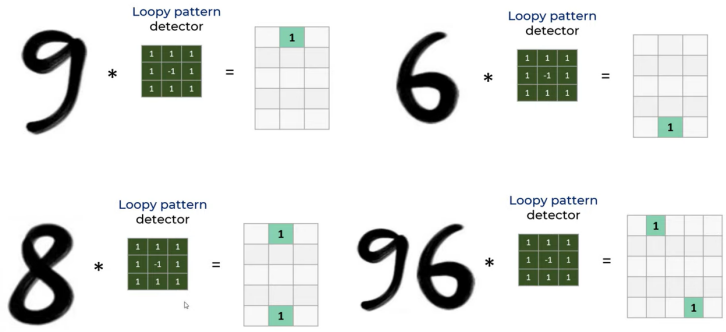
so computer take the original image and apply a convolution operation or a filter operation so here computer will have a loopy circle pattern or a head filter, the way

convolution operation works is computer will take three by three grid from original image and multiply individual numbers with head pattern filter so each digit of the filter is multiplied with the original image number and then average is taken and mentioned at first place on feature map. Feature map is nothing but the map created by multiplying all grids of the original image with a head pattern filter.
Image below will help to understand.

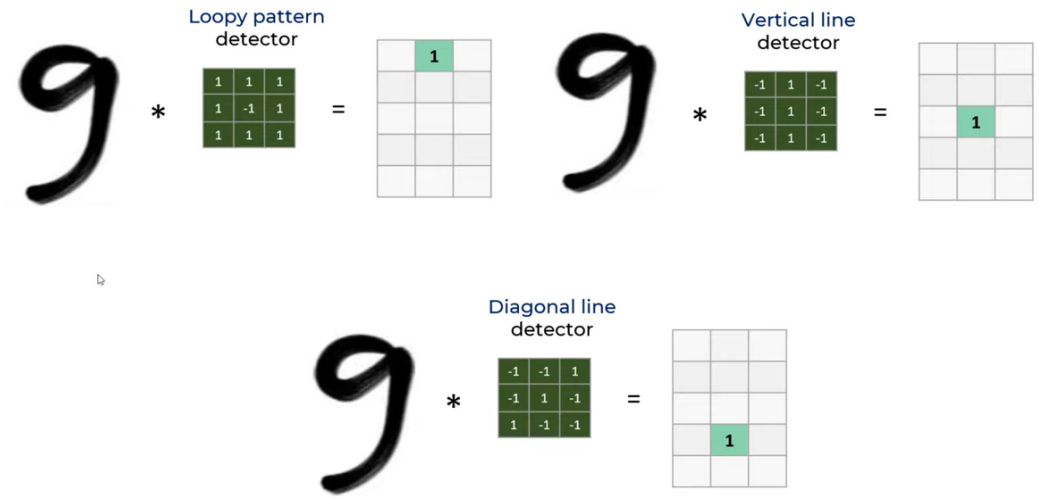


Here the array size is 3X3 and stride is 1. Array size and stride (shift of array) can be any it can be 4X4 and 2 or 2X2 and 1.

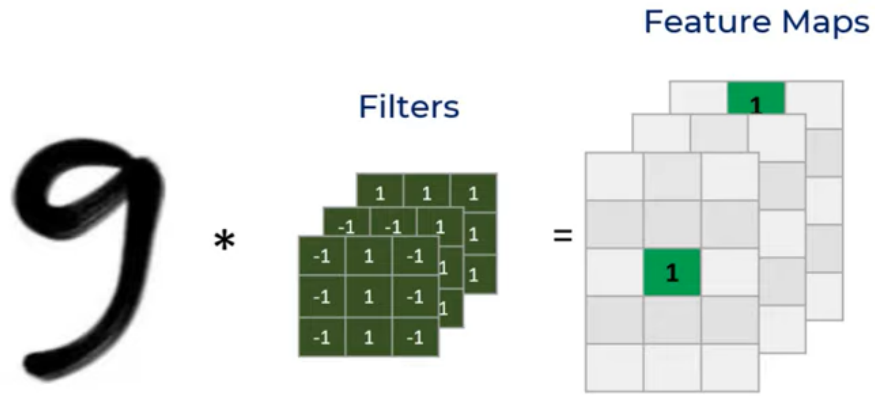
wherever the computer sees number one or a number that is close to one it means the image has a loopy circle pattern so by applying a loopy pattern detector the computer got this one here in the feature map also called as the feature is activated. Same way for number six it will be activated in the bottom, if the image has two loopy patterns the feature will be activated at top and bottom then it may be number “8”. For number “96” it will be activated in different areas.



in summary when computer apply this filter or a convolution operation computer are generating a feature map that has that particular feature detected so in a way filters are nothing but the
feature detectors.



In the case of number 9 computers, need to apply three filters the head the middle part and the tail and when computer apply those computer get three feature maps.

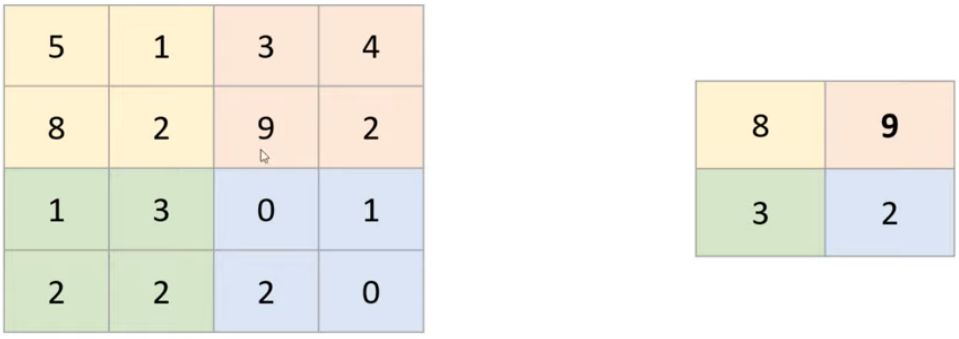


So the first part of convolution neural networks id feature extraction. second part where the computer uses a dense neural network for classification. there are two other components one is

ReLU. ReLU will bring non-linearity in the model so what it will do is it will take an image feature map and whatever negative values are there it just replaces that with zero and the value is more than zero it will keep it as it is.

Introducing non-linearity in the model computer will make it capable of detecting more Complex inputs. If there is no non linearity then the output will be a linear function of the input which is similar to input so the model will learn only the linear relationship between input and output. So in order to learn the non-linear relationship between output and input there is a need to introduce non-linearity.

After introducing non linearity the time is to reduce the computation and that will be done by reducing the size of the image and that is called Pooling. The main purpose of Pooling is to reduce the dimensions. Max Pooling, here the computer takes a window of 2x2 and the computer picks the maximum number from that window and generates a new feature map, But the new feature map is of small size. so it's too much or saving in computer computation.



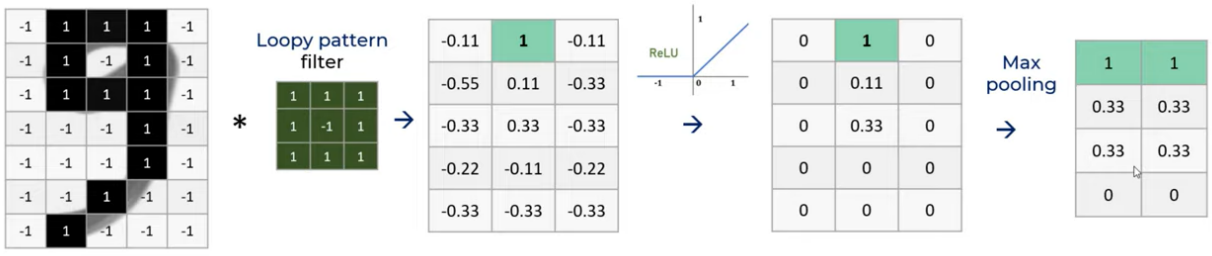
2 by 2 filter with stride = 2

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

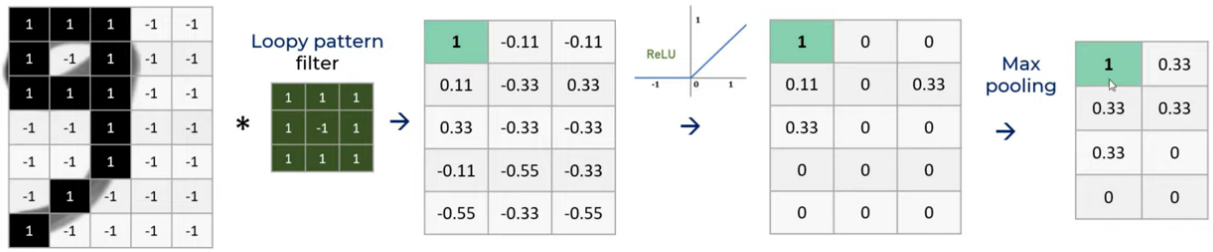
1	1

2 by 2 filter with stride = 1

In case of Digit “9” computer apply max pooling and detect loopy pattern at top also even if number is shifted loopy pattern is in top as below.



Shifted 9 at different position



so max pulling along with convolution helps computers with position invariant feature detection. there is average pooling also instead of max computer just make an average. so benefits of max pooling are reducing computer dimension and computation, the second benefit is reduce overfitting because there are less parameters and the third one is model is variant tolerant towards variation and distortion because if there is a

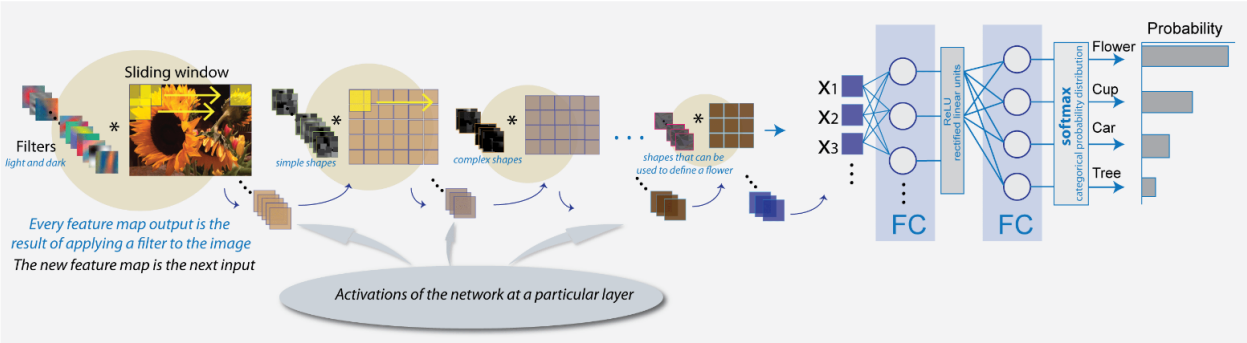
distortion and if computer are picking just a maximum number then computer are capturing the main feature and filtering all the noise. Image will go through such multiple cycles of convolutional and pooling. Detecting numbers is a simpler task so it may not require many cycles of convolutional and pooling, but detection of complex images such as the human face will go through multiple cycles of convolutional and pooling. The main idea behind convolutional neural networks is feature extraction. Flattening transforms the pooled feature map into a single column that propagates to a fully connected layer. Depending on the problem and network, there may be multiple fully connected layers. The final fully connected layer is responsible for the output of predictions.

In this project matlab is used for designing convolution neural networks. The Designed model will identify two Type of faults:

1.Belt broken
2.Motor need lubrication

Before fault detection lets go through the process of creating CNN model training of it and classification of incoming data. Convolution neural network architecture primarily consists of Layers of neurons (Convolution layers) , Output of one layer is transferred to the second layer and a network of search interconnected layers will define the total architecture. These players can be predefined matlab layers or they can be custom layers.

Below image will show a working of CNN in short and project will run on similar basis to find the faults.



Below is the part of the code that shows the layers.

```
layers = [ ...
    imageInputLayer([28 28 1])
    convolution2dLayer(5,20)
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer]
```

Here “layers” is the object of the class layer.

mageInputLayer([28 28 1]) : This layer will input the two dimensional image to the network and content in a square bracket represents the size of the image in pixels.

convolution2dLayer(5,20) : This layer will define the filter size, stride , number of filters and apply the convolution to the input 2-D layer, The layer convolves the input by moving the filters along the input vertically and horizontally. (5,20) specify there are 20 filters of size 5X5.

reluLayer : This layer will add non-linearities in the model by Performing a threshold operation to each point of the input where it set the value as 0 if the input value is less than zero.
layer = reluLayer('Name','relu1') : defines the relyLyaer with name relu1.

maxPooling2dLayer(2,'Stride',2) : Max pooling will basically reduce the size of the incoming data so that it will take less computation. creates a max pooling layer with pool size [2 2] and stride 2.

fullyConnectedLayer(10) : Neurones present in a fully connected layer are having connections with all the neurons in the preceding layer. This layer will combine all the features detected by the previous layers to identify a large pattern. and that combination of all the patterns will be used for classification.

softmaxLayer : This layer will apply the softmax function to the incoming data. Output of the softmax layer will have positive numbers and they will be added together to one digit that will be passed to the classification layer to match possibilities.

classificationLayer : This layer is mostly the last stage of convolution neural network where classification is done. On the basis of output of the last stage this layer suggests the number of classes.

add = additionLayer(2,'Name','add_1') : this additional layer will add input from multiple neural networks.

batchNormalizationLayer: This layer generally plays cold between convolution layer and relu layer, this layer will normalize data across multiple batches so the training process can be speed up.

There are many other layers and this layers will form an array of 6 stages.

```
layers =
  6x1 Layer array with layers:

     1   ''   Image Input             28x28x3 images with 'zerocenter' normalization
     2   ''   2-D Convolution         10 5x5 convolutions with stride [1  1] and padding [0  0  0  0]
     3   ''   ReLU                    ReLU
     4   ''   Fully Connected         10 fully connected layer
     5   ''   Softmax                 softmax
     6   ''   Classification Output   crossentropyex
```

layerGraph : this function will connect all layers together in sequential manner.

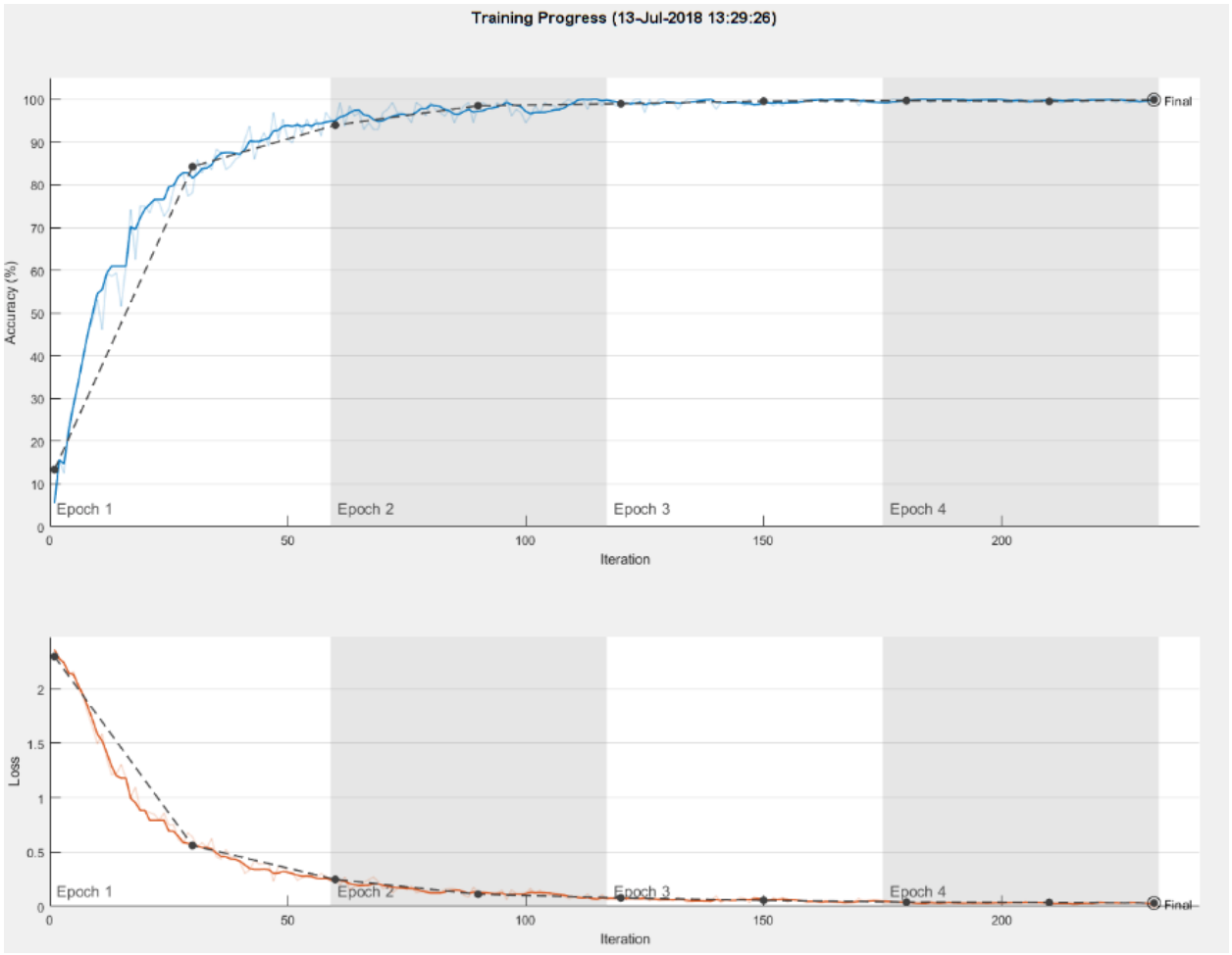
Once Model structure is defined, model needs to train so that it can recognize incoming data. For training stochastic gradient descent with momentum is used. Gradient descent is an optimization algorithm commonly used in machine learning applications to find model parameters that provide the best fit between predicted and actual outputs.In simple words this algorithm will perform iterative operation on inputs from Previous layer and move towards close approximation to predict the result.

Below is the code for training:

```
[XTrain,YTrain] = digitTrain4DArrayData;
[XValidation,YValidation] = digitTest4DArrayData;
options = trainingOptions('sgdm', ...
    'MaxEpochs',8, ...
```

```
    'Shuffle','every-epoch', ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
net = trainNetwork(XTrain,YTrain,lgraph,options);
net
YPredicted = classify(net,XValidation);
accuracy = mean(YPredicted == YValidation)
```

digitTrain4DArrayData : this function will load the training data and validation data. Training options will define the options such as which method will get used, training interval also plotting of training progress is defined here. Number of epochs are defined here. Epoch is the number of training iterations to perform on the training data set to get an accurate model. Training progress will plot the Graph of Accuracy and corresponding loss through each Epoch.



trainNetwork : this function will Train the neural network by using layers and the options specified in the function.

Then code will predict the labels of the validation data with the help of a trained network. Based on how many labels are predicted, accuracy of the model is defined.

Once training is done the classification is done by using LSTM (Long short-term memory (LSTM) layer).
LSTM is basically a layer with hidden states and used to solve the problem with traditional RNN. in LSTM will build a state which holds the long term memory.

inputSize = 12;
numHiddenUnits = 100;

```
numClasses = 9;

layers = [ ...
   sequenceInputLayer(inputSize)
   lstmLayer(numHiddenUnits)
   fullyConnectedLayer(numClasses)
   softmaxLayer
   classificationLayer]
```

```
layers =
  5x1 Layer array with layers:

     1   ''   Sequence Input          Sequence input with 12 dimensions
     2   ''   LSTM                    LSTM with 100 hidden units
     3   ''   Fully Connected         9 fully connected layer
     4   ''   Softmax                 softmax
     5   ''   Classification Output   crossentropyex
```

LSTM will help in better prediction.

End part of the project is fault detection.

One of the faults under consideration is related to motor lubrication. There is no formula to find the direct lubrication of the motor but finding direction will help to know about lubrication faults. Extended kalman filter is required to predict friction data. Extended kalman filters are used to estimate the vehicle position, velocity and that is applicable to the non linear models.

Motor is simulated and charged with input voltage pulses and non linearity in speed will be introduced with changing the pulse frequency. Motor is simulated in code and initial parameters such as angular velocity are defined in code.

Below code will create the extended kalman filter

```
ekf = extendedKalmanFilter(...
   @pdmMotorModelStateFcn, ...
   @pdmMotorModelMeasurementFcn, ...
   x0,...
   'StateCovariance',          [1 0; 0 1000], ...[1 0 0; 0 1 0; 0 0 100], ...
   'ProcessNoise',             Q, ...
```

```
    'MeasurementNoise',          R, ...
    'StateTransitionJacobianFcn', @pdmMotorModelStateJacobianFcn, ...
    'MeasurementJacobianFcn',     @pdmMotorModelMeasJacobianFcn);
```

Simulation model will be created for motor assembly and friction value is monitored using the kalman filter to detect the need of lubrication.

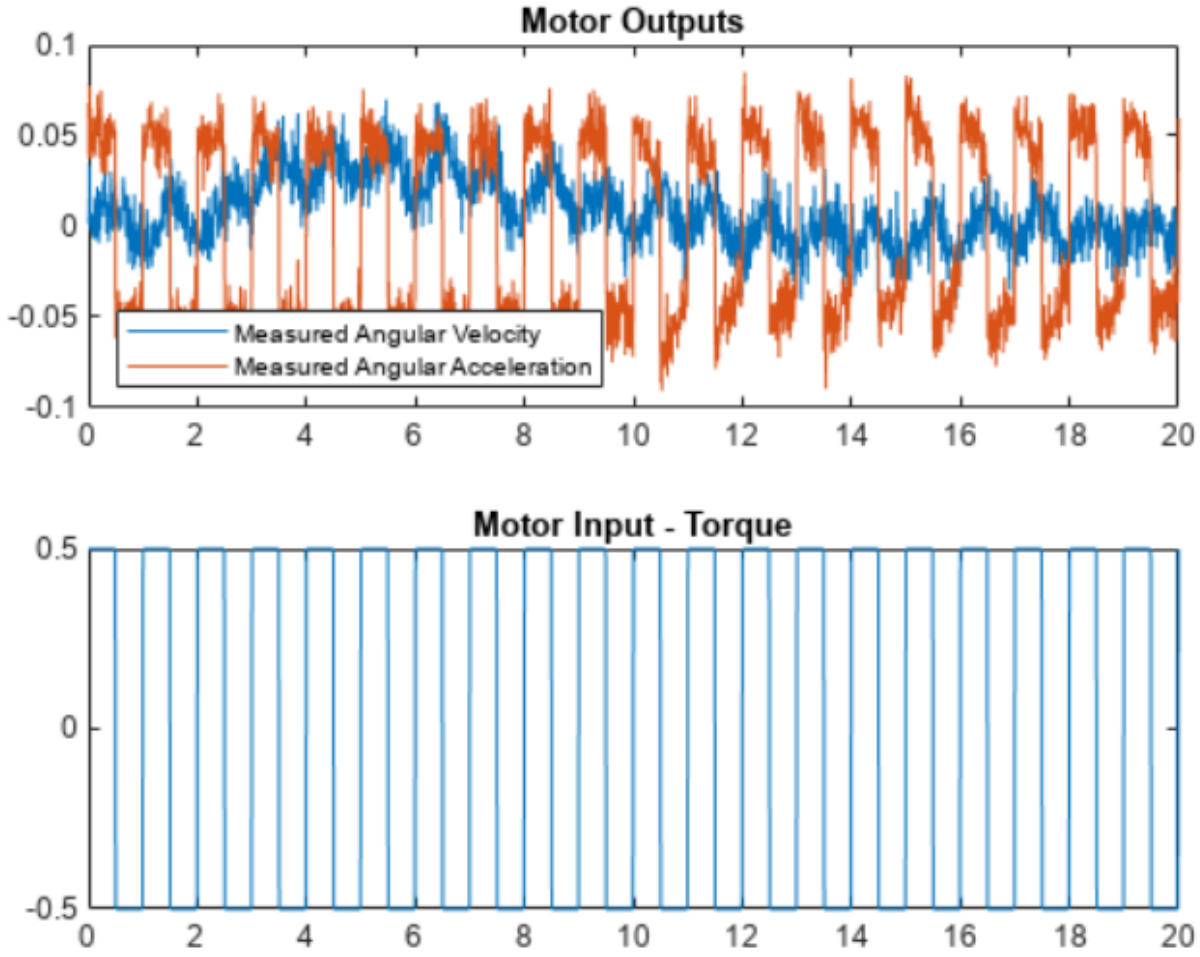
Under ideal situations friction value should be the same for normal working of the motor and changes in standard fix value will be used to determine the fault. Whenever there are changes in any values and those ranges are just moving up and down the fix value the mean and standard deviation are good tools to determine how much they are deviated. In this project also friction mena values are computed and standard deviation from the mean is calculated and that is done with 4 moving windows. Lock the calculated mean and standard deviation after the first 7 seconds. This first computed average is the average without friction prediction error. If the estimated friction after 7 seconds is more than 3 standard deviations from the expected error-free mean, this indicates that the friction has changed significantly.

Code for that

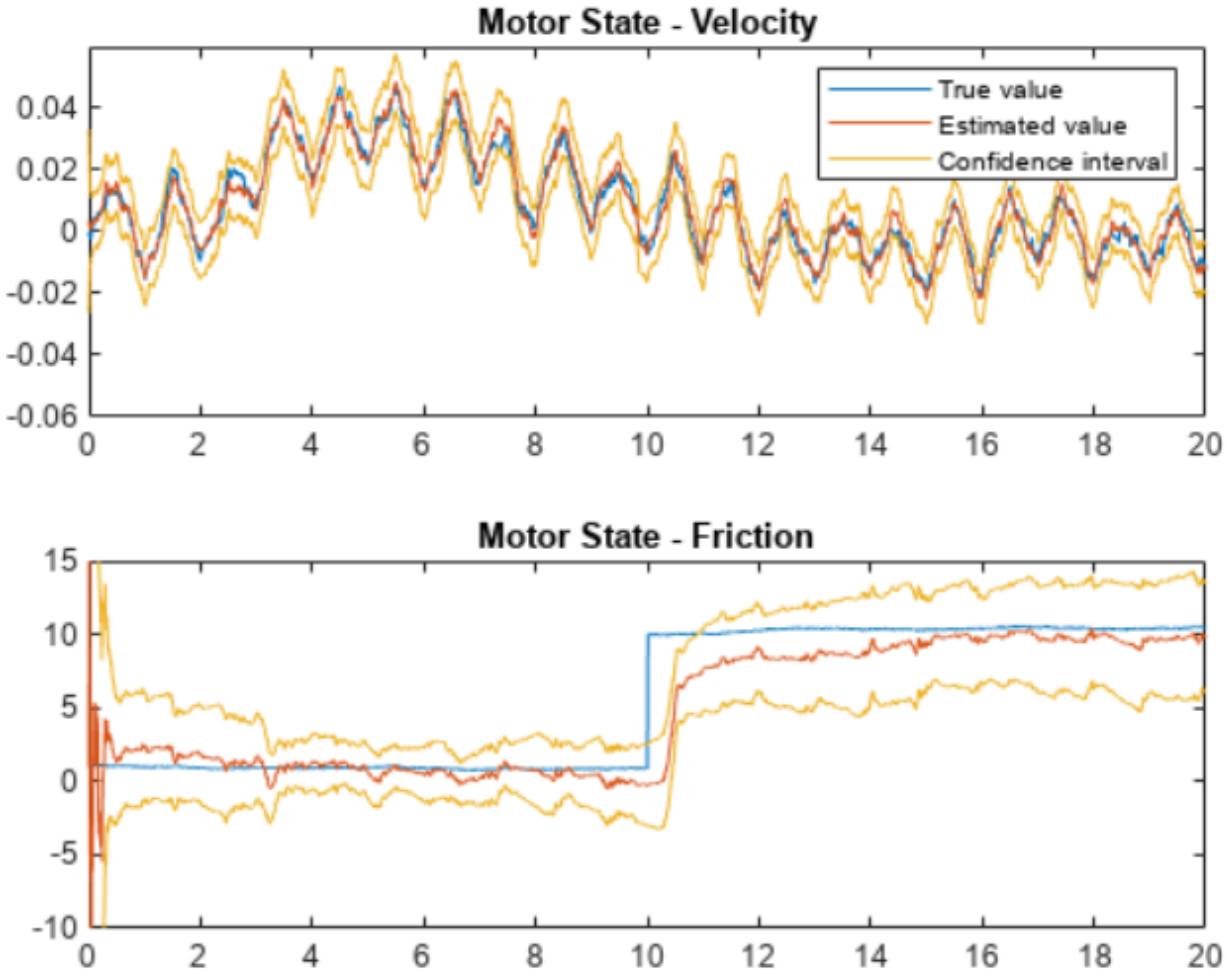
```
    if t(ct) < 7
     idx = max(1,ct-400):max(1,ct-1); % Ts = 0.01 seconds
     fMean(ct) = mean( xSigEst(2, idx) );
     fSTD(ct)  = std( xSigEst(2, idx) );
  else
     fMean(ct) = fMean(ct-1);
     fSTD(ct)  = fSTD(ct-1);
     estFriction = mean(xSigEst(2,max(1,ct-10):ct));
              fChanged(ct) = (estFriction > fMean(ct)+3*fSTD(ct)) || (estFriction < fMean(ct)-3*fSTD(ct));
  end
  if fChanged(ct) && ~fChanged(ct-1)
     fprintf('Significant friction change at %f\n',t(ct));
  end

   ySigEst(:,ct) = pdmMotorModelMeasurementFcn(x_corr,u(ct),J,Ts);
  idx = max(1,ct-400):ct;
  fKur(:,ct) = [...
```

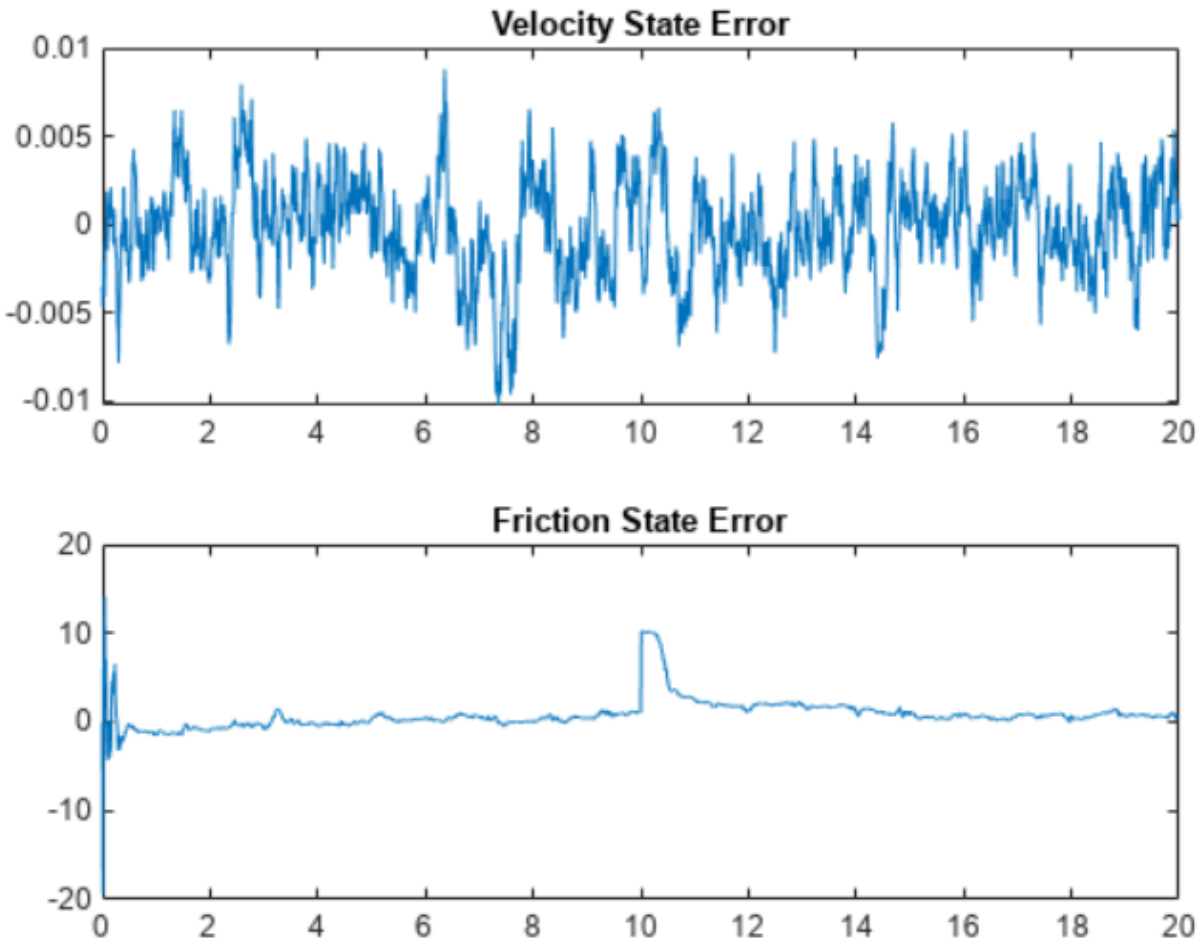
```
    kurtosis(ySigEst(1,idx)-ySig(1,idx)); ...
    kurtosis(ySigEst(2,idx)-ySig(2,idx))];
End
```



From the graphs above it is not easy to detect the friction change directly so an extended kalman filter will help to determine the friction state.

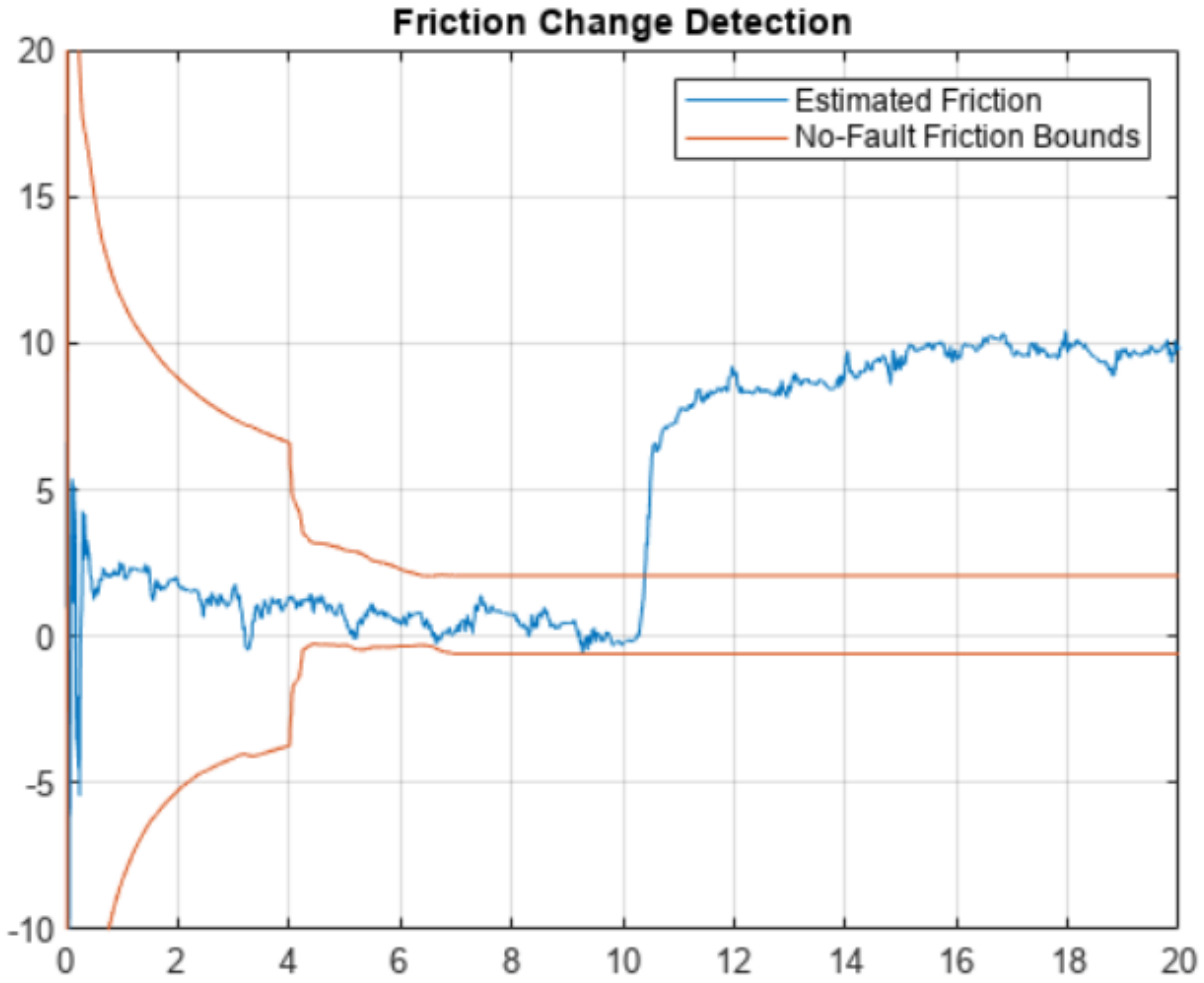


Above the graphs are the real values and below the graphs are error state (error is nothing but the difference between previous and current.)



Estimated friction change is a good way to detect the engine faults. When there is no error the bound for friction value is same or fix and when error is introduced friction values will deviate and show the fault detection which is nothing but the need of lubrication to the motor.

Below graph shows that deviation and that is nothing but detection.


Friction Change Detection
Estimated Friction
No-Fault Friction Bounds
20
15
10
5
0
-5
-10
0
2
4
6
8
10
12
14
16
18
20