Creating a Deep learning model (based on LeNet) for machine fault detection:

- Multiclass classification problem
- 13 classes: Helical 1, Helical 2, Helical 3, Helical 4, Helical 6, spur 1, spur 2, spur 3, spur 4, spur 5, spur 6, spur 7, spur 8
- Dataset: PHM data challenge 2009
- Evaluation: using Confusion matrix.
- Programming language: MATLAB
- Literature:
  1) "A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox" by Jing et. Al.
  2) "Bearing Fault Diagnosis Based on Convolutional Neural Networks with Kurtogram representation of Acoustic Emission Signals".

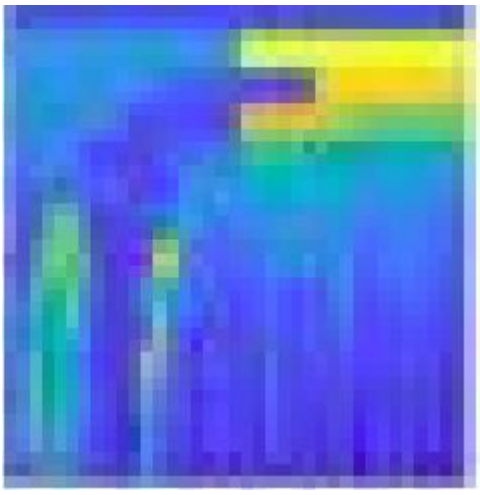
Data set preparation:

Details:
- dataset consisting of 20 files for each class
- Each file consisting of readings from 3 different sensors.
- There are 266656 readings for each sensor.

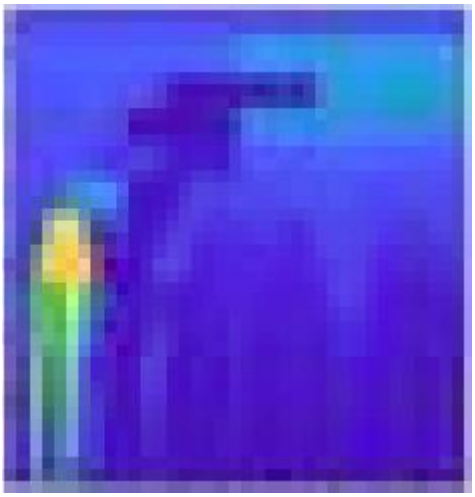
Preprocessing:
- Sampled reading from 1st sensor at an interval of 10,000 for each class.
- Created kurtogram from the sampled data.
- kurtogram is saved as an image on disk with the size of 48*48*3.
- each class consisting of 539 images.

Sample Kurtogram for each class:

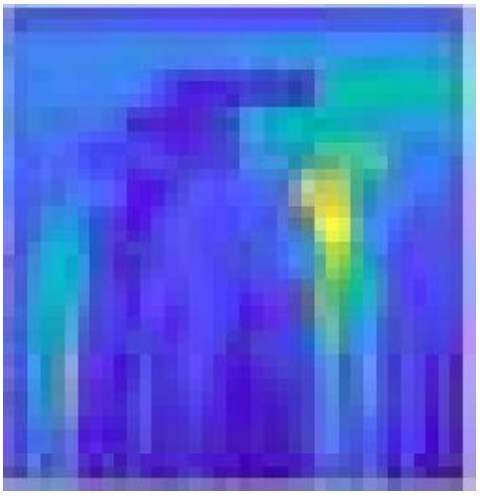
1) Helical 1



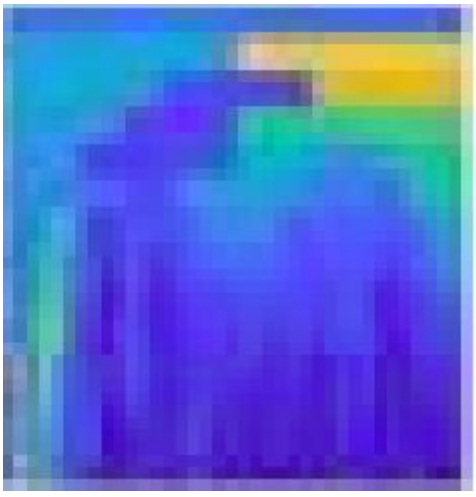
2) Helical 2



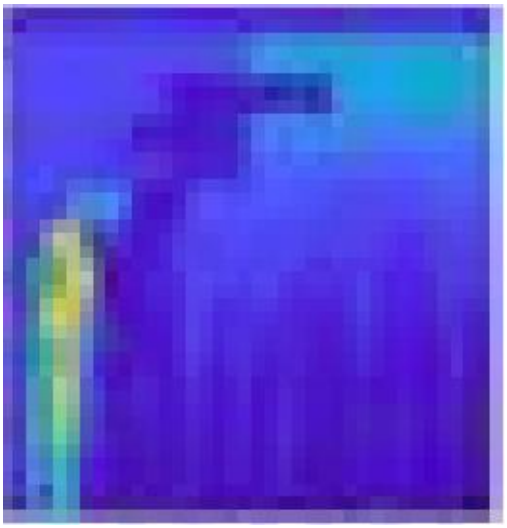
3) Helical 3



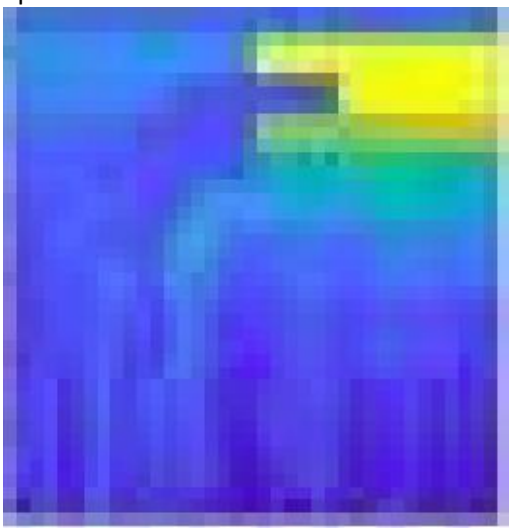
4) Helical 4



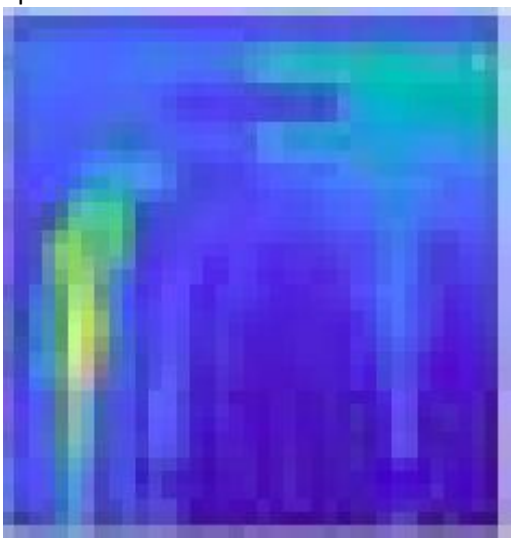
5) Helical 6



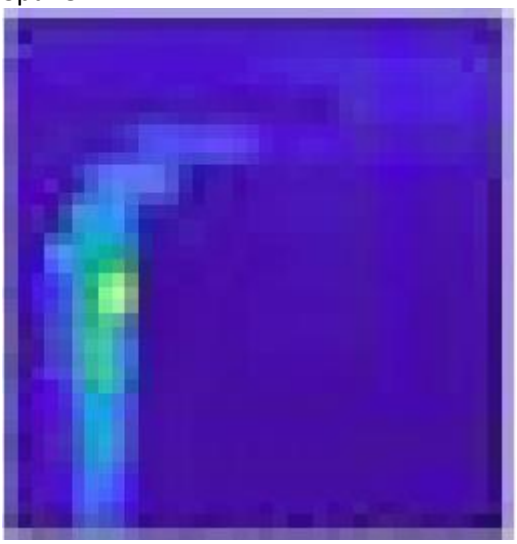
6) Spur 1



7) Spur 2

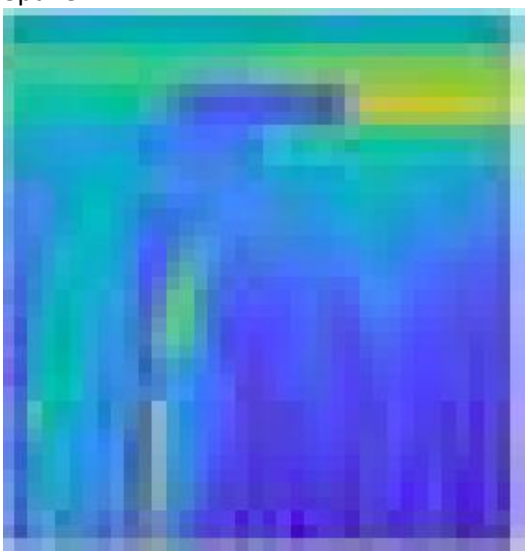


8) Spur 3

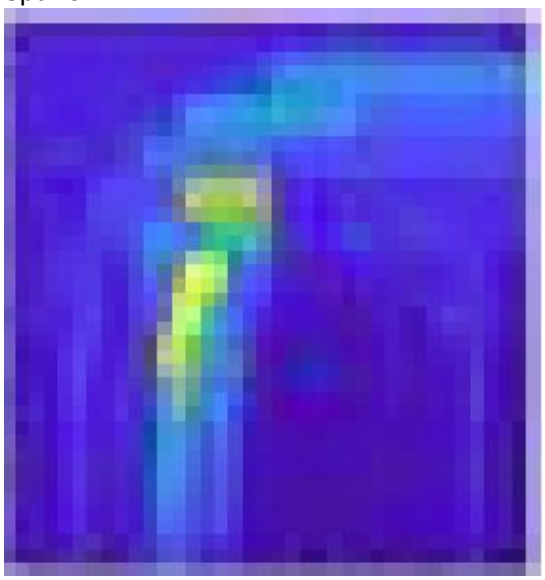


9) Spur 4

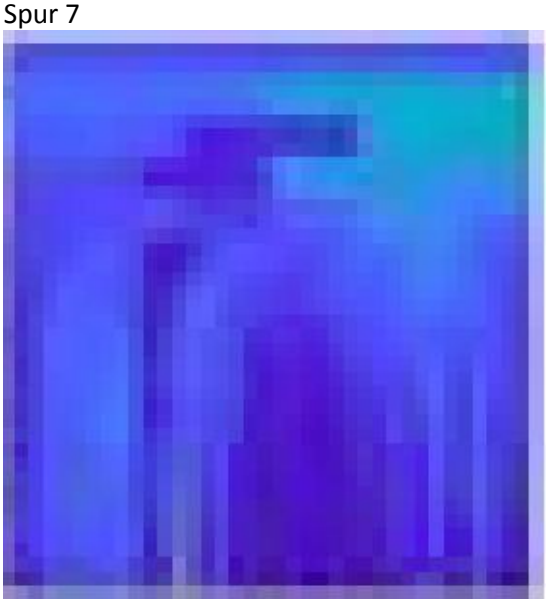
10) Spur 5



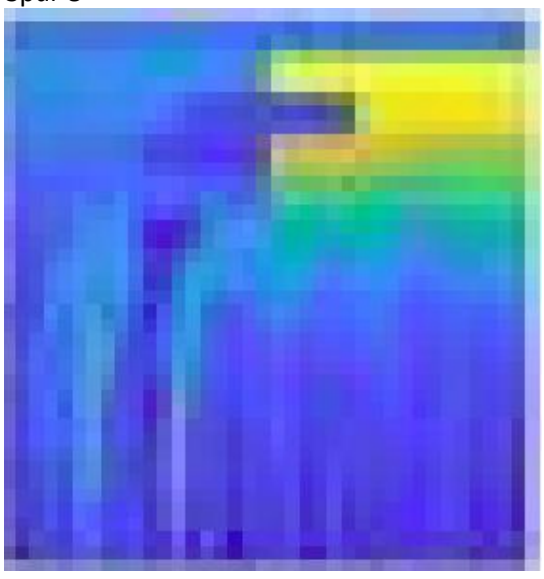
11) Spur 6



12) Spur 7



13) Spur 8



Code Snippet to preprocess the data and save kurtogram

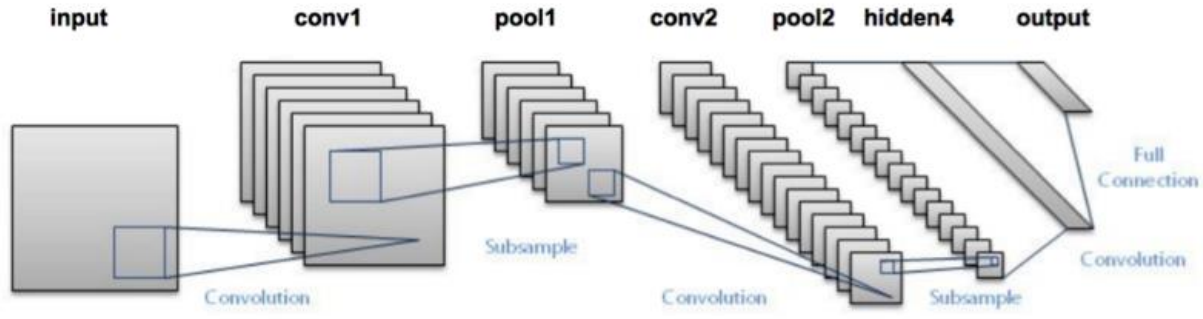
```
Files =
dir(['C:/Users/nmadan/Desktop/deeplearning_onMatlab/phm_datachellenge_200
9/data/train/spur 1/*']);
    for k=1:length(Files)
        if ~(Files(k).bytes == 0)
            Data=
dlmread(['C:/Users/nmadan/Desktop/deeplearning_onMatlab/phm_datachellenge
_2009/data/train/spur 1/',Files(k).name]);
                itr = 0;
                for i = 1:26
                    filename =
sprintf('C:/Users/nmadan/Desktop/deeplearning_onMatlab/phm_datachellenge_
2009/data/Images/spur 1/Image_%.5d',file_count);
                    data{1, 1}  = kurtogram(Data(itr+1:itr+10000, 1));
                    figure('visible', 'off');
                    imagesc(data{1,1})
                    saveas(gcf, filename, 'jpg')
                    itr = itr + 10000;

                    image = filename+".jpg";
                    I = imread(image);
                    J = imresize(I,[48 48]);
                    imwrite(J, image);
                    file_count = file_count + 1;
                    disp(image)
                    disp(itr)
                    clear filename image I J
                    close(gcf)
                end
```

Creating deep Model:

-Used “imageDatastore” to load the data from disk.

-Train and test split:
For each class: 500 train images and 39 test images.

Network architecture:



First layer:
The input for LeNet-5 is a 48×48×3 RGB image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one.

Second Layer:
Then the LeNet-5 applies average pooling layer or sub-sampling layer with a filter size 2×2 and a stride of two.

Third Layer:
Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1.

Fourth Layer:
The fourth layer is again an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer except it has 16 feature maps

Fifth Layer:
The fifth layer is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the units in fifth layer are connected to all the nodes in the fourth layer.

Sixth Layer:
The sixth layer is a fully connected layer with 84 units.

Output Layer:
Finally, there is a fully connected softmax output layer ŷ with 13 possible values corresponding to classes Helical 1, Helical 2, Helical 3, Helical 4, Helical 6, spur 1, spur 2, spur 3, spur 4, spur 5, spur 6, spur 7, spur 8

Code Snippet for deep model i.e. Le-Net

```
layers = [
    imageInputLayer([48 48 3])

    convolution2dLayer(5,6,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(5,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(5,120,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(84)
    fullyConnectedLayer(13)
    softmaxLayer
    classificationLayer];
```

Training details:

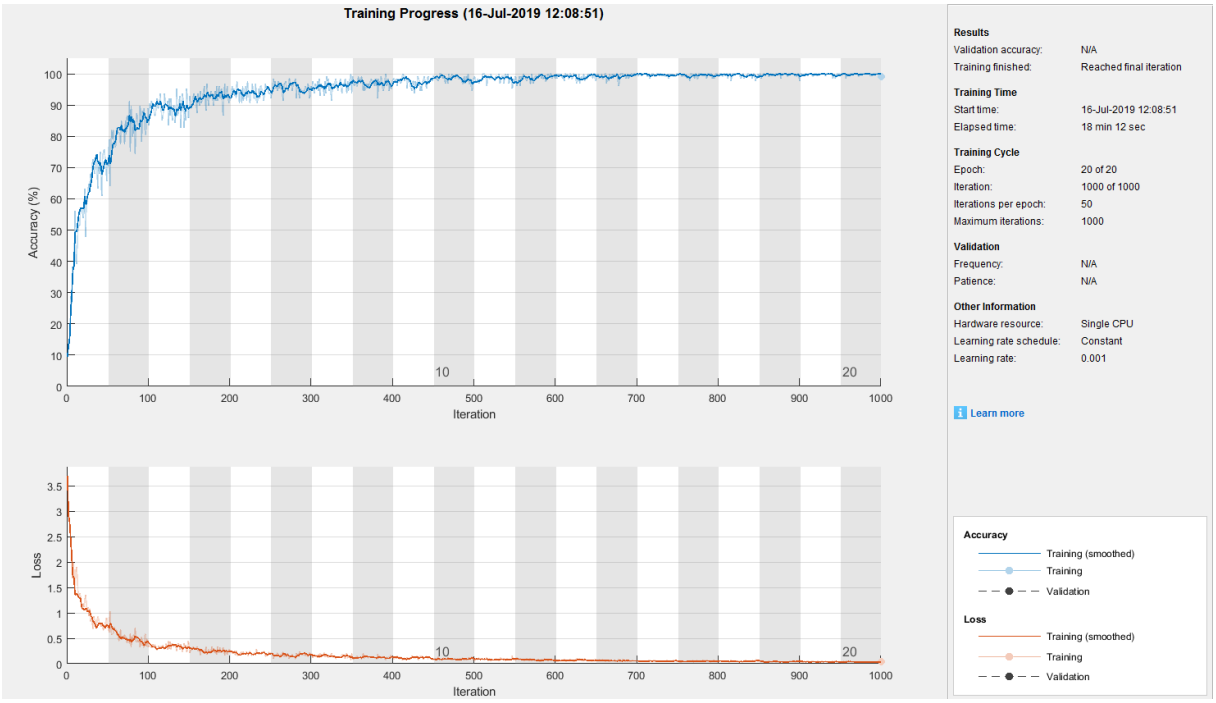
dataset is trained for 20 epochs using optimizer SGD (stochastic gradient descent) with the learning rate of 0.001. The whole process took 20 min to train.

Sample training images (kurtogram):

Code Snippet for training network:

```
%% determining the train option
options = trainingOptions('sgdm', ...
    'MaxEpochs',20,...
    'InitialLearnRate',0.001, ...
    'Shuffle','every-epoch', ...
    'Verbose',false, ...
    'Plots','training-progress');

%% training the network
net = trainNetwork(imdsTrain,layers,options);
```

Training progress per epoch:



Results:

Testing phase consist of 39 images from each class and total accuracy (number of correct predictions by the model) come out to be 91.6%.

Code snippet of testing phase:

```
%% testing it on validation set
YPred = classify(net,imdsTest);
YTest = imdsTest.Labels;

plotconfusion(YTest,YPred)

accuracy = sum(YPred == YTest)/numel(YTest);
```

Confusion matrix for the classification results: