ANOMALY DETECTION IN CASE OF IMAGE AND .MAT DATASET

Amit Aryan (11713263).
Prakash Roy ().
Mohammad Adil (11713162).
Omkar Prabhakar Ghadage (11713303).
School of Computer Science and Technology,
Lovely Professional University,Jalandhar, India.

---

1. Abstract:

Deep Learning techniques have provided us with a lot of emerging methods that can create magical optimum predictions, classifications, clusterings with real ease without over-working on data. These techniques of deep learning make it more powerful for today's developer's who are working in a pool of data for analytics. Anomaly detection is one of these algorithms which is well known in the world of Data Science and Machine Learning. Anomaly detection is the process of identifying unexpected items or events in data sets, which differ from the norm. This algorithm is well known in the field of statistics, signal processing, finance, ecometrics, manufacturing, networking, data mining, etc. These various domain fields use anomaly detection for intrusion detection, fraud detection, system health monitoring, fault detection, event detection in sensors networks and detecting ecosystem disturbances. Considering these wide applications we are going to work on two different data models to understand behaviour of this algorithm. The first Data we are considering is the accelerometer vibration dataset provided by IBM and second dataset is Image dataset consisting of cloud and non-cloud images. For the first dataset we are going to checkout for abnormal behaviour of signal and in further steps we are going to predict future anomalies too. In the case of Cloud VS Non-cloud dataset we are going to find cloud images which are working as Anomaly to the normal images. It can be considered as a Disregularity scenario in case of Healthy data. In upcoming sections we are going to dive deeper in the case-study of anomaly detection.

Keywords : Deep Learning, Anomaly detection, Anomaly Supervised Learning, Dataset, Tensorflow, etc.

2. Introduction:

Anomaly detection is nothing but the identification of rare items in a dataset. Typically the anomalous items will translate to some kind of problem such as Bank fraud, a structural defect, medical problems or errors in the data. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions. Anomalies are observations in a dataset that don't fit in some way. Perhaps the most common or familiar type of anomaly is the observations that are far from the rest of the observations or the center of mass of

observations. This is easy to understand when we have one or two variables and we can visualize the data as a histogram or scatter plots, although it becomes very challenging when we have several different input variables defining high-dimensional input feature space. In this case, simple statistical methods for identifying anomaly can break down, such as methods that use standard deviations or the interquartile range.

It can be important to identify and remove anomalies from data when training machine learning algorithms for predictive modeling. Anomalies can skew statistical measures and data distributions, providing a misleading representation of the underlying data and relationships. Removing anomalies from training data prior to modeling can result in a better fit of the data and, in turn, more skillful predictions.

Till the time we have just considered why it is necessary to remove anomaly from our dataset. But now I am going to actually explain how we are going to do that. So let's analyze what Anomaly Detection problem statement is all about. Let's consider that our dataset consists of N data points. It can be represented as-

Data = x**1** , x**2**, x**3**,.....,x**N .**

Where, each x**i** = Є R^d .

Let's say that this dataset is a mixture of 'nominal' and 'anomaly' points. Where anomaly points are generated by a different generative process than the nominal points. There are three settings (modes we can call) in case of Anomaly detection like Supervised, Clean and Unsupervised. In case of Supervised setting we will have training data labelled with nominal or anomaly (Example Cloud Vs Non-cloud dataset which we are going to use). Where in case of Clean setting training data are all nominal and test data is contaminated with anomaly points (For example the Healthy data and unhealthy data we are using in signal anomaly detection). Finally, unsupervised settings will have training data of a mixture of nominal and anomaly points (We are not going to cover this!)

Let's Dive into mathematical interpretation of Anomaly detection. As assumpted previously -

Data = x**1** , x**2**, x**3**,.....,x**N .**

Where, each x**i** = Є R^d .

Now, Let's consider Density Estimation for further use here.

P(x) = P(x**1**,μ**1**, $\sigma 1^2$) * p(x**2**,μ**2**, $\sigma 2^2$) * P(x**3**,μ**3**, $\sigma 3^2$)* ..... *P(x**N**,μ**N**, σN^2) ------(1)

Where x is defined in gaussian distribution with some mean(μ) and variance(σ^2).

Mean = μ = (∑x)/N .

Variance= σ^2 = $(\sum x - \mu)^2/(N-1)$

If we consider this equation which we just wrote in terms of statistics then we can understand that it corresponds to an independence assumption on the values of features x**1** to x**N**. On further simplification we can write equation (1) as -

P(x) = ∏i=0 to n [P(x**i**,μ**i**, σi^n)] ------(2)

Where, ∏ means nothing but multiplication symbol. This is one statistical approach to locate anomalies. As we have a foundation statistical block of Anomaly detection so we can dive further into algorithms.

Step 1- Choose features xi that you think might be indicative of anomalous examples.

Step 2- Fit parameters μ1,.....,μ**N,** $\sigma 1^2$,...., σN^2 .

Step 3- Given new example x, compute P(x):

P(x)= ∏i=0 to n [P(x**i**,μ**i**, σi^2)]

P(x)=Ⅱ(j=1toN) (The function passed as param is as below)
$[1/sqrt(2\pi\sigma j^2) * exp(-(\sum x - \mu)^2/2\sigma j^2)]$

Anomaly if P(x) < ε.

Where, ε = Norm value.

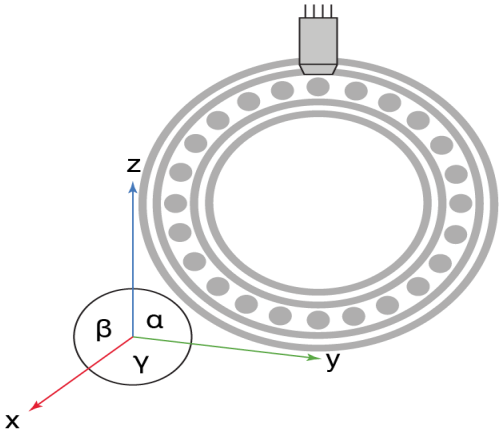
3. Exploring Datasets:

Every data project starts with data. Data is a very broad term. It can be structured or unstructured, big or small, fast or slow, and accurate or noisy. To effectively demo the process of creating a deep learning solution on these different technologies, we need data. We need structured, fast, and big data, which can be noisy too.

3.1 Exploring Accelerometer vibration Dataset (DS-1):

The data we are using here is time-series data from vibration (accelerometer) sensor data. This data is generated with an IOT sensor simulator. So for processing purposes we are going to use a really handy tool called the Lorenz attractor model.

This is a very simple, but still very interesting, physical model. Lorenz was one of the pioneers of chaos theory and he was able to show that a very simple model that consists of just three equations and four model parameters can create a chaotic system that is highly sensitive to initial conditions and that also oscillates between multiple semi-stable states where state transitions are very hard to predict. We have generated test data by using a physical Lorenz Attractor model also because it is capable of generating a three-dimensional data stream. We have used the generated data in this demo to detect anomalies, predicting when a bearing is about to break.



(The Accelerometer sensor on a bearing records vibrations on each of the three geometrical axes x, y, and z). The dataset is available in .mat format as it is extracted from lorenz attractor model. Files with the . mat extensions are files that are in the binary data container format that the MATLAB program uses. The extension was developed by Mathworks and MAT files are categorized as data files that include variables, functions, arrays and other information.

3.2 Exploring Cloud Vs Non-cloud Dataset (DS-2):

The cloud Vs Non cloud anomaly data consist of a total 1600 jpg images. Under the Non-cloud section there are total 1500 jpg images and in the cloud section there are 100 jpg images. The images are RGB type and are having shape as (38,384). As the image is of RGB pattern so finally we can say that each image has 3 color channels; So that image size for feeding the network is (384,384,3).

Example of cloud Images -

Example of Non-Cloud Images -

We can clearly see that there is influence of cloud in images under section of Cloud folder. And in case of noncloud folder there is no influence of clouds in satellite imagery that have been provided. But when we check complete folder of non-cloud images we will see that there are some cloud images too. So we are going to train our model using this non-cloud image folder and later we will test the model with cloud folder images.

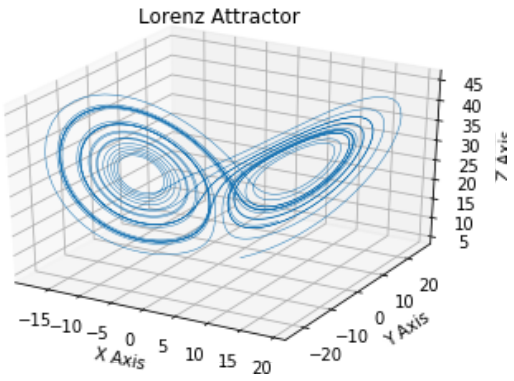
(Example of Cloud influenced image in noncloud section)

That's all about exploration of both the datasets. Now we will discuss the different methods that I have used to train our models.

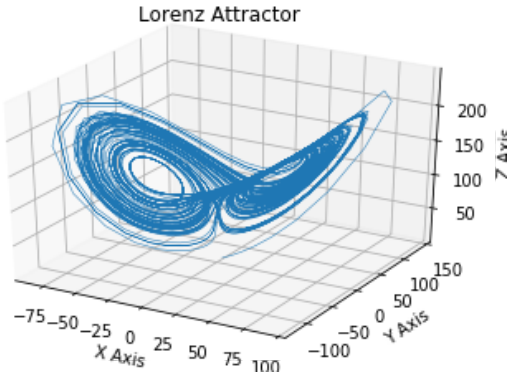
4. Approaches To tackle our Problems:

4.1 Using Feed Forward Network for Anomaly Detection:

Here I am re-using the unsupervised anomaly detection algorithm but turn it into a simpler feed forward neural network for supervised classification. We are training the neural network from healthy and broken samples and at a later stage hook it up to a message queue for real-time anomaly detection.Later we grab the files necessary for training. Those are sampled from the Lorenz attractor model implemented in NodeRED. Those are two serialized pickle numpy arrays. Since this data is sampled from the Lorenz Attractor Model, we will plot it with a phase lot to get the typical 2-eyed plot. First for the healthy data and later for unhealthy data. The plot for Healthy data looks like follows-



Let's visualize broken data plot-

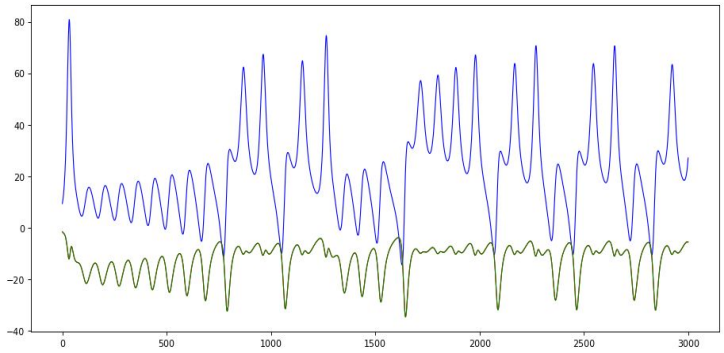


Now we want to use an ordinary feed-forward network. So we need to do some pre-processing of this time series data. A widely-used method in traditional data science and signal processing is called Discrete Fourier Transformation. This algorithm transforms from the time to the frequency domain, or in other words, it returns the frequency spectrum of the signals. The most widely used implementation of the transformation is called FFT, which stands for Fast Fourier Transformation, so we are going to use the same.

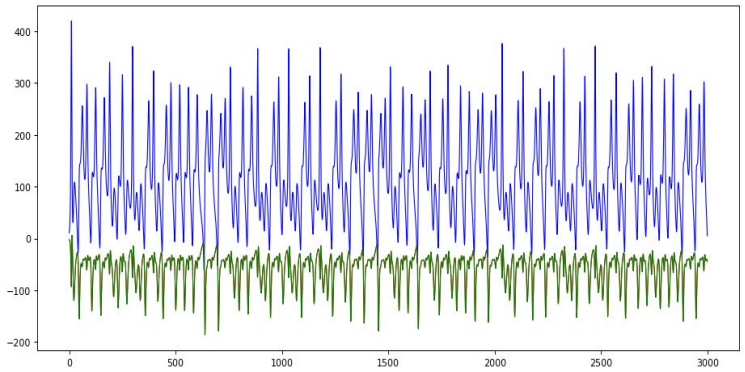
After fourier transformation, we notice that the shape is the same as the input data. So if we have 3000 samples, we get back 3000 spectrum values, or in other words 3000 frequency bands with the intensities. The second thing we notice is that the data type of the array entries does not float anymore, it is complex. So those are not complex numbers, it is just a means for the algorithm to return two different frequency compositions in one go. The real part returns a sine decomposition and the imaginary part a cosine. We will ignore the cosine part in this example since it turns out that the sine part already gives us enough information to implement a good classifier.

But first let's visualize plots of the two arrays to get an idea of a healthy and broken frequency spectrum, How they differ from each other -

Healthy_data_FFT



Unhealthy_data_FFT



We've transformed the data set in a way that our machine learning algorithm (a deep feed forward neural network implemented

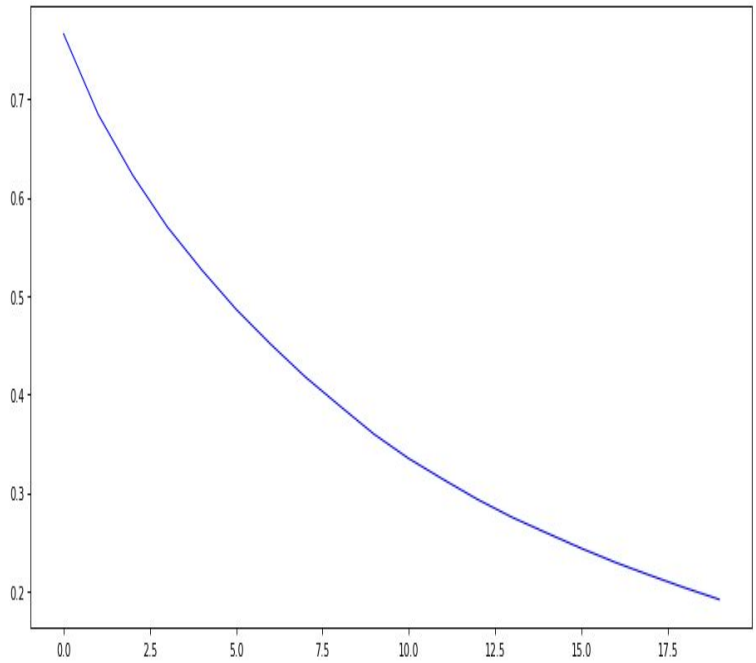
as a binary classifier) works better. So now let's scale the data to a - 0..1

Let's reshape again to have 3 only examples (rows) and 3000 features(columns). We have turned our initial data set which contained 3 columns (dimensions) of 3000 samples. Since we applied FFT on each column we've obtained 3000 spectrum values for each of the 3 three columns. We are now using each column with the 3000 spectrum values as one row (training example) and each of the 3000 spectrum values becomes a column (or feature) in the training data set.

As we are done with preprocessing and understanding our dataset we will configure parameters for our network and so the model. Since this is only a toy example, we don't use a separate corpus for training and testing. We just use the same data for fitting and scoring.Here I have prepared the training data by concatenating a label "0" for the broken and a label "1" for the healthy data. Finally we union the two data sets together. So once we concatenate both training data named "Broken" and "Healthy" we can clearly observe that those bands almost have the same energy. But the neural network algorithm still can work those out which are significantly different. Now understand that as per our model framework we have to provide the first 3000 columns of the array as the 1st parameter and column number 3000 containing the label as 2nd parameter. We are using slicing here to pass data as such.

Now it's time to do the training. You should see the loss trajectory go down, we will also show it with plot later. (Note: We also could use TensorBoard for this but for this simple scenario we skip it.) In some rare cases training doesn't converge simply because random initialization of the weights caused gradient descent to start at a suboptimal spot on the cost hyperplane. Just recreate the model (the cell which contains *model = Sequential()*) and re-run all subsequences to avoid that.

After plotting the losses we will figure out how our model has performed.



We have almost achieved 89% accuracy using feed forward neural networks for anomaly detection in this case.

4.2 Using CNN (Pretrained) for Anomaly Detection:

Convolutional Neural Network or Comp Net is an artificial neural network that is so far been most popularly used for analyzing images. Although image analysis has been the most widespread use of CNN's they can also be used for other data analysis and classification problems as well most generally we can think of a CNN as an artificial neural network that has some type of specialization for being able to pick out or detect patterns and make sense of them.

This pattern detection is what makes CNN so useful for image analysis. So if a CNN is just some form of an artificial neural network what differentiate it from other neural networks? Well it is just a standard multilayer perceptron or MLP. CNN has a hidden layer called convolutional layers and these layers are precisely what make CNN effective. So now the question arises “What are these Convolutional layers?” Convolutional layer receives input and transforms input into some way and then outputs the transformed input to the next layer. The transformation here is called a convolution operation.

When adding a convolutional layer to the model we also have to specify how many filters we want the layer to have. A filter can technically just be thought of as a relatively small matrix for which we decide the number of rows and number of columns that this matrix has and the values within the matrix are initialized with random numbers. Basically we are giving a size of filter for convolving operation here. If we consider a filter of size 3*3 then this filter will run across all the image in 3*3 manners gathering the output for a single cell of our matrix. The convolve operation can be formulated as follows -

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

To find anomaly(Cloud images) we will train our model only with non-anomaly(Non Cloud images) data. So that our model can understand non anomaly data more, than anomaly data. We will train it with pre pre-trained CNN model called VGG19. VGG19 is a huge pre-trained convolutional neural network created by IMAGENET. We are using VGG19 by method of Transfer Learning. In case of Transfer learning the used model had been already trained with different problem statement. This method is really flexible as it allows feature extraction, preprocessing and can be directly integrated into another new model using a pretrained model.Transfer learning has the Advantage of reducing the training time of a neural network model and further it could result in less generalization error. Let’s visualize the VGG19 Model before moving to further sections.

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 384, 384, 3)]	0
block1_conv1 (Conv2D)	(None, 384, 384, 64)	1792
block1_conv2 (Conv2D)	(None, 384, 384, 64)	36928
block1_pool (MaxPooling2D)	(None, 192, 192, 64)	0

block2_conv1 (Conv2D) (None, 192, 192, 128) 73856

block2_conv2 (Conv2D) (None, 192, 192, 128) 147584

block2_pool (MaxPooling2D) (None, 96, 96, 128) 0

block3_conv1 (Conv2D) (None, 96, 96, 256) 295168

block3_conv2 (Conv2D) (None, 96, 96, 256) 590080

block3_conv3 (Conv2D) (None, 96, 96, 256) 590080

block3_conv4 (Conv2D) (None, 96, 96, 256) 590080

block3_pool (MaxPooling2D) (None, 48, 48, 256) 0

block4_conv1 (Conv2D) (None, 48, 48, 512) 1180160

block4_conv2 (Conv2D) (None, 48, 48, 512) 2359808

block4_conv3 (Conv2D) (None, 48, 48, 512) 2359808

block4_conv4 (Conv2D) (None, 48, 48, 512) 2359808

block4_pool (MaxPooling2D) (None, 24, 24, 512) 0

block5_conv1 (Conv2D) (None, 24, 24, 512) 2359808

block5_conv2 (Conv2D) (None, 24, 24, 512) 2359808

block5_conv3 (Conv2D) (None, 24, 24, 512) 2359808

block5_conv4 (Conv2D) (None, 24, 24, 512) 2359808

block5_pool (MaxPooling2D) (None, 12, 12, 512) 0

conv2d_transpose (Conv2DTran (None, 24, 24, 8) 36872

conv2d_transpose_1 (Conv2DTr (None, 48, 48, 16) 1168

conv2d_transpose_2 (Conv2DTr (None, 96, 96, 64) 9280

conv2d_transpose_3 (Conv2DTr (None, 192, 192, 256) 147712

_____

_____

conv2d_transpose_4 (Conv2DTr (None, 384, 384, 512) 1180160

_____

_____

conv2d_transpose_5 (Conv2DTr (None, 384, 384, 3) 13827

==========

==========

Total params: 21,413,403
Trainable params: 21,413,403
Non-trainable params: 0

_____

So from the above summary we can see that the VGG19 Model has 2141303 trainable parameters. Which implies that this model is a huge Neural Network.

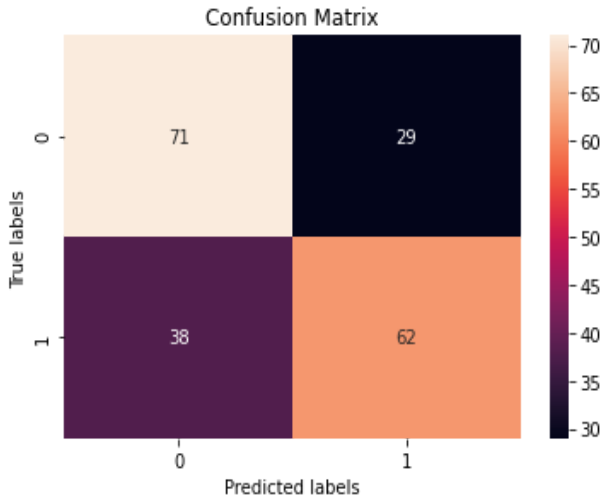
When we train our model with non-anomaly data, Reconstruction_loss will be less for our non-anomaly data when compared to anomaly data. We are also going to use Autoencoders here. Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. Let's dive into autoencoders to know them well :

- Encoder: It is similar to any simple ANN model. In which the model learns how to minimize the input dimensions and compress the input data into an encoded representation.
- Bottleneck: It is nothing but a hidden layer which contains the compressed representation of the input data.
- Decoder: The decoder model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.
- Reconstruction Loss: This is the method that measures how well the decoder is performing and how close the output is to the original input.

We can use AutoEncoders for denoising as well. Here we are passing anomaly data to model as an input and training it to remove the noise from the data.

Now we have discovered the procedure that I am using for this model. So let's discuss the last chapter of this Anomaly detection problem and that is the result. Here I am presenting results in the form of F1_score, Recall and Precision. As well as by using Seaborn I am also providing the Confusion Matrix for the same in image next to scores.

	matrix	Values
0	F1_score	0.649215
1	Recall	0.620000
2	Precision	0.681319



5. Acknowledgement:

This research paper is submitted to the School of Computer Science, Lovely Professional University, Phagwara. This term paper is considered as an assignment for the course Deep Learning Techniques with Tensorflow(INT248) and prepared keeping that in concern. The faculty of respective subject Mr.Sanjay Kumar sir have guided me throughout completion of this term paper by providing exact information of patterns that needed to follow while writing the term paper. So I would like to show my gratitude to Mr. Sanjay Kumar sir. Although if there is any error in this document then it's of my own and should not tarnish the reputation of an esteemed person.

6. References:

1. https://keras.io/api/applications/vgg/#vgg19-function (Documentation)
2. https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/ (TowardsDataScience)
3. https://www.kaggle.com/ashoksrinivas/cloud-anomaly-detection-images (Kaggle Dataset)
4. https://www.youtube.com/watch?v=nTt_ajul8NY (simplyLearn)
5. Material provided by IBM on coursera for course Deep Learning A-Z
6. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (TowardsDataScience)