CORROSION DETECTION

Université Bishop's University

By

- Dhiraj Wani 002253001

- Vishwas Hasija 00225112

Introduction

Corrosion inspection is one paramount operation that must be performed periodically by public road administrations or kindred entities. Inspections are often carried out manually, sometimes in hazardous conditions. The main reason for corrosion inspection is to maintain and nourish the lifecycle of civil infrastructures such as bridges, pipes. Furthermore, such a process may be very sumptuous and time consuming but eventually can be considered fruitful as it can help extend life cycle of existing infrastructures. Researchers have put an abundance of effort into endeavoring to optimize such costly processes by utilizing robots capable of carrying out automatic bridge maintenance, reducing the need for human operators. However such a solution is still very extravagant to develop and carry out. As of late private sector companies have commenced giving infrastructure assessment housing utilizing drones with tall determination cameras. These are able to perform and review civil infrastructures in numerous unfavorable conditions, such as with a bridge collapse, and/or review of the underside of lifted bridges, pipelines. The recordings and pictures procured with this strategy are to begin with put away and after that subsequently checked on physically by bridge organization engineers, who choose which activities are required. Indeed in spite of the fact that this sort of robotization gives clear focal points, it is still exceptionally time consuming, since a physical individual must sit and observe hours and hours of obtained video and images.

Challenges

Not as it were are man-hours an issue, Computer Science & Data Innovation for foundation resource directors, so is human subjectivity. Infrastructure administrators are these days requesting strategies to examine pixel-based datasets without the need for human intercession and interpretation. The conclusion result wanted is to impartially conclude in the event that their resources display a fault or not. As of now, this conclusion changes concurring to the individual doing the picture elucidation and analysis. Comes about are hence conflicting, since the presence of a fault or not is deciphered differently depending on the person. Where one person sees a fault, another may not. Developing an objective fault acknowledgment framework would include esteem to existing datasets by providing a dependable pattern for framework resource directors. The manual handle of assessment is somewhat disposed by having automated robotic arms or cameras taking pictures of components from different points, and after that having an reviewer go through these pictures to decide a rusted component, that needs repair. Indeed this prepare can be very tedious and exorbitant, as engineers has to go through numerous such pictures for hours together, to decide the condition of the component. Whereas conventional computer vision methods have been utilized with restricted victory, within the past, in recognizing erosion from pictures, the appearance of Deep Learning has opened up an entire unused possibility, which may lead to exact location of erosion with small or no manual intercession. Machine learning approach would be considered a replacement for existing manual interference.

Approach

Numerous methodologies are already existing and many developers and computer vision professionals have developed different systems based on different needs which can be considered remarkable. Considering those different systems and using those approaches as a foundation for our involvement in this we have chosen to execute one version of classic computer vision (based on red component) and one deep learning model and perform a comparison test between the two distinctive approaches. Numerous diverse systems and libraries are accessible for both the classic computer vision procedures and the Deep Learning approach.

Tools Used

We have used python as the programming language in our project. The reason behind using python is because it is capable of executing program using different libraries as per the need from the thousands of available libraries. The second reason can be considered is because already python is open source language there is a large community world-wide helping and providing answers to many known errors, in short large help community is available over the internet which makes programmer's life peaceful. We have use spyder as the IDE while developing the program as it is easy to install and import libraries within the same framework and also it is an open source application. Considering the test images we have medium sized dataset, we have approximately 160 images including the images with rust area as well as non-rust images. We have randomly picked this image from over internet. Noticable thing is the more the dataset the more accurate output we get. But since we are developing it on our local machine it is not possible to use large dataset since it is time consuming process. But yes if we go ahead with a live server with more

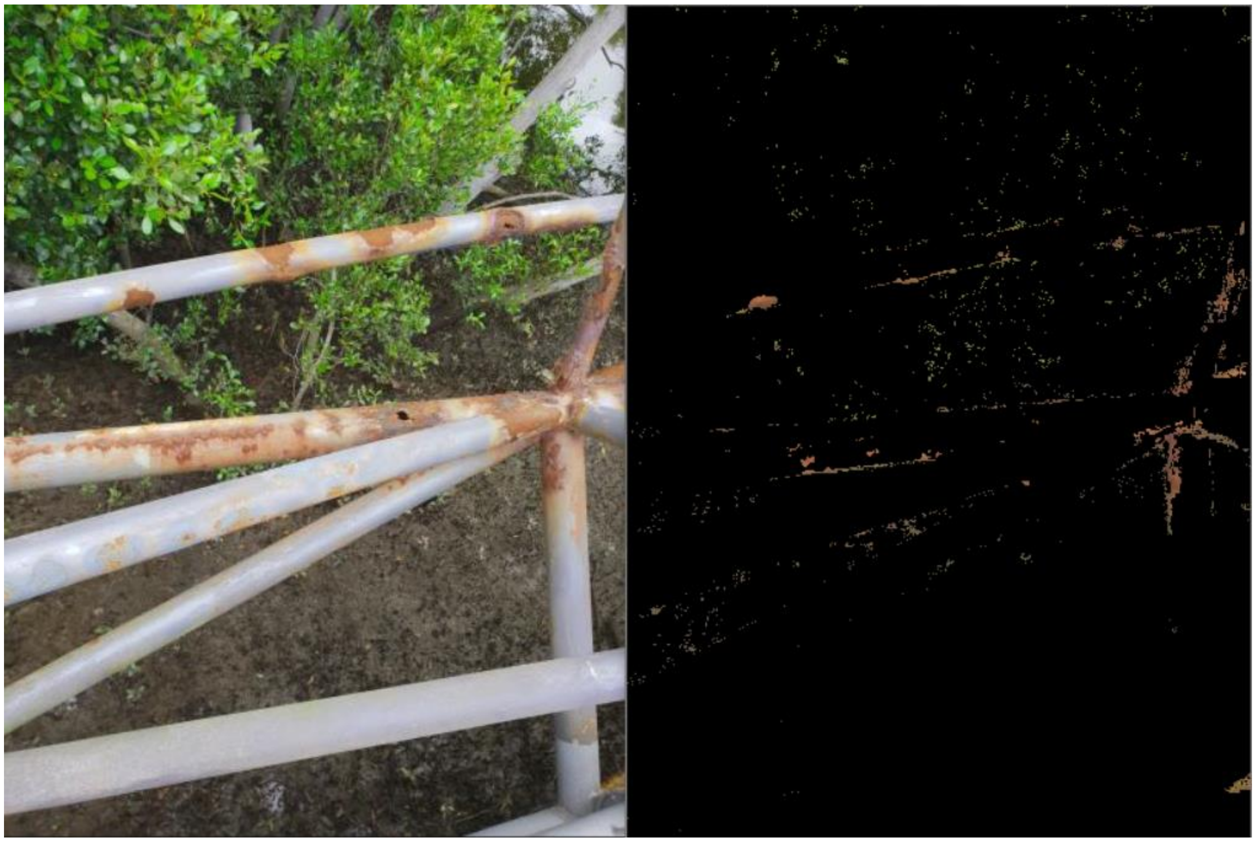
RAM size we can use large dataset. Our dataset is efficient considering the results we need to achieve.

Models Used

As discussed above we have used two different approaches i.e. opencv which is computer vision model and other is deep learning model.

1. Classic Computer Vision Technique:

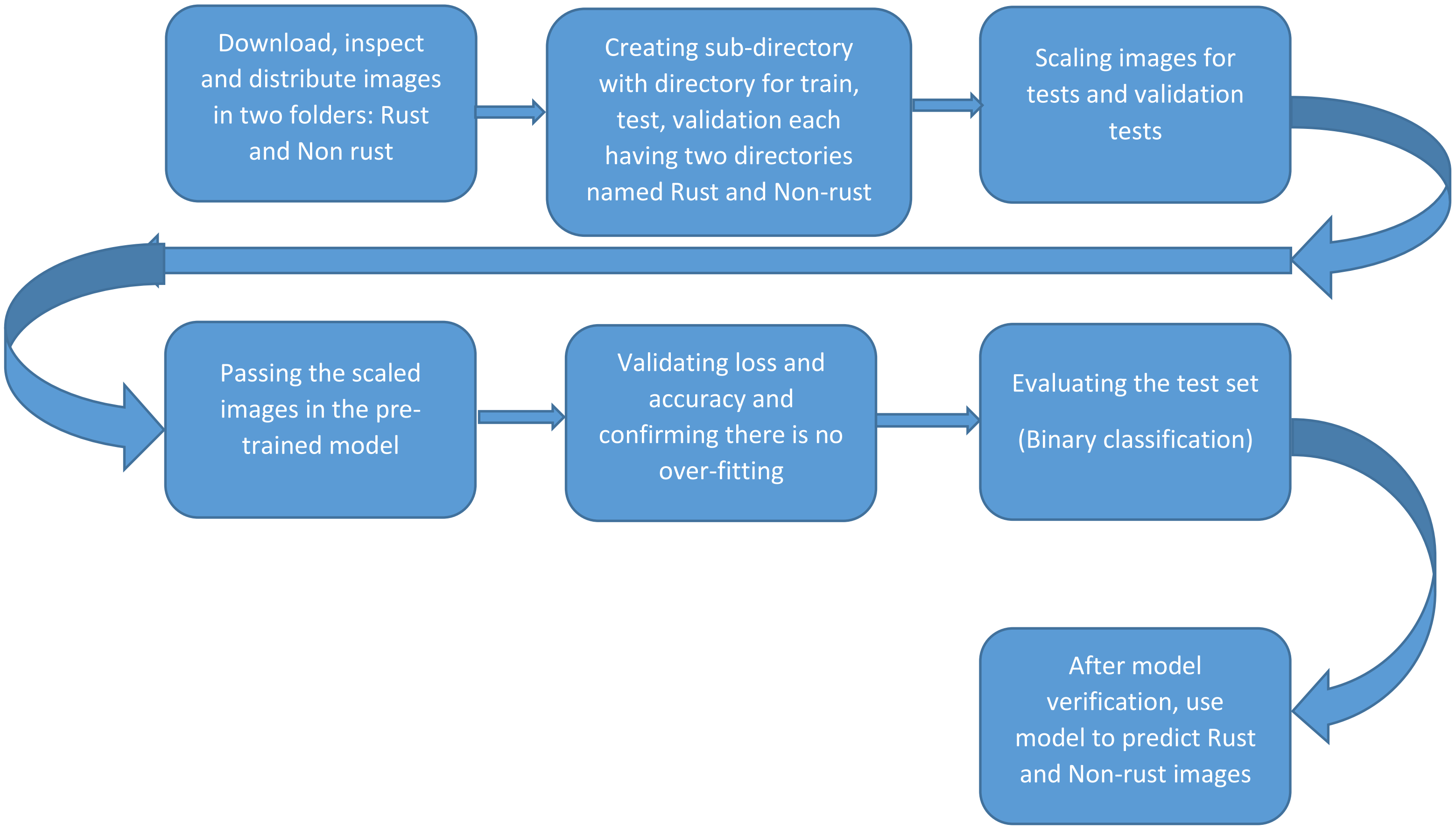
For nearly two decades, engineers in computer vision have depended on OpenCV libraries to develop their programs. With a client community of more than 47 thousand individuals and evaluated number of downloads surpassing 7 million, this set of >2500 calculations and valuable capacities can be considered standard libraries for picture and video applications. The library has two interfaces, C++ and Python. In any case, since Python-OpenCV is fair a wrapper around C++ functions (which perform the genuine computation seriously code), the output loss in execution by utilizing Python interface is very less. For these reasons we chose to create our to begin with classifier using this set of tools. The classifier was generally fundamental. Since an eroded area (rust) has no clear shape, we chosen to focus on the colours, and in specific the red component. After essential filtering, we changed the picture colour space from RGB to HSV, in addition to this we have split the image in two masks, filtered it and then added it back. In addition, since not all the red component was useful for the rust discovery, we attempted to experimentally limit down the component in arrange to discover the best interim that was not result in as many false positives. After broad testing we found the best interim in rust location, to be 0-11 and 175-180. Moreover we smoothed component to the extend 50-255. This cover was at that point changed over into dark and white and the white pixels were checked. Each picture having more than 0.3% of white pixels was at long last classified as “rust”, whereas having less than 0.3% of white pixels shown a “non-rust” detection. The output was quite near to the expectation though noise wasn’t gone completely which lead us to move towards the different approach. The following image is the left part with the input image and the right part as output image,



The output image is capable of detecting the rusted part, whereas it is also detecting some noise what we guess from the input image is that it is identifying red or orange colored parts/area from the tree which if you look carefully exists in the image. Though the output is clear about the rusted part we have to move towards different approach since it is also detecting noise which is not the aim of this project and hence deep learning model was preferred by us with the determination and aim to ignore or avoid the unexpected rust detected area from the input image. Yes, the variation is different according to the input image, sometimes we get accurate output with less noise but sometimes it shows more noise instead of detected rust area. We think this approach will work with the deep learning model.

2. Deep Learning Model:

The second approach was based on counterfeit insights, in specific utilizing Deep Learning methods. This approach isn't new. The scientific show of back-propagation was to begin with developed in ‘70s and was one of the primary genuine applications of Deep Learning. This system is particularly suited for picture processing, advertising great speed and extraordinary adaptability. It too offers the opportunity to effortlessly utilize clusters back for demonstrate preparing which may well be valuable within the case of large systems. To begin with, was to gather a great dataset to be used to train the model. We were able to gather around 160 pictures for the “rust” and “non-rust” class. Around 80% of the pictures were utilized for the preparing set, while the rest was utilized for the approval set. In fine tuning, the system took an already prepared arrange and balanced it (continuing the preparing) utilizing the unused information as input. This technique gives a few preferences. To begin with of all, it permits the reuse of already prepared networks, sparing a part of time. Deep Learning strategies have been known to extricate surface based highlights exceptionally successfully. Besides, we presently have a library of pre-trained models accessible as portion of open source stores. The spatial include pecking order learned by the pre-trained show successfully acts as a nonexclusive show, and subsequently its highlights can be utilized for a diverse computer vision issue that might include a totally distinctive classification. Such transportability of learned highlights over distinctive issues could be a key advantage of profound learning and it makes profound learning exceptionally compelling for a small-data situation. We handle this issue as a two-step process. To begin with, we utilize Deep Learning with pre-trained models, to do parallel classification of pictures - those having 'rust' and those with 'no rust'. Surprisingly, this works exceptionally well. Once we recognize the picture as having rust, we create a deep learning show to draw a bounding box around the rust. The following diagram represents the flow of the model and Image classification of rust,


Download, inspect and distribute images in two folders: Rust and Non rust
Creating sub-directory with directory for train, test, validation each having two directories named Rust and Non-rust
Scaling images for tests and validation tests
Passing the scaled images in the pre-trained model
Validating loss and accuracy and confirming there is no over-fitting
Evaluating the test set (Binary classification)
After model verification, use model to predict Rust and Non-rust images

For the primary step of Image classification (rust and norust), we utilize the pre-trained VGG16 that Keras gives out-of-the-box by means of a straightforward API. Since we are applying transfer-learning, let’s solidify the convolutional base from this pre-trained demonstrate and prepare as it were the final completely associated layers. The outcomes about are beautiful astounding! We get a precision of 87%, without any major tinkering with the hyper-parameters or attempting out distinctive pre-trained models.

2.1 Why deep learning

The important feature of deep learning is that without specifying the features explicitly it learns complex features itself. It helps to extract rust features automatically while training the model with rusted components. Deep learning algorithms are known for texture based features. One of the common library ImageNet trained CNN(VGG16) which is readily available is used to extract the rust features effectively. VGG16, Very Deep Convolutional Networks for large Scale Image is developed by Visual Geometry Group. It is trained to detect 1,000 different objects with high accuracy rate. '16' in VGG16 stands for the number of layers involved. VGG16 has five pooling layers with size of 2*2, followed with three fully connected layers and final layers as soft-max layer. We used VGG16 to classify given image as corroded or not corroded.

2.2 Data Preparation

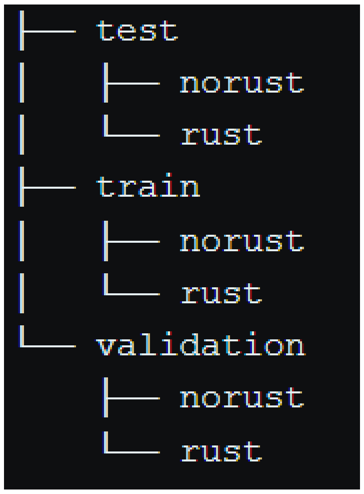
The primary step in any Deep Learning model, as normal, is gathering labeled information and isolating up the data into train, test and validation sets. Here comes Google to offer assistance. We basically look 'rust Images' on google, and download them. It's better done physically, to choose the great determination pictures. The next step is separating up the data into tain set, validation set, and test set. This is often finished by making isolated envelopes for prepare, approval, and test each having sub-folders – rust and norust. Presently we will repeat through the downloaded pictures and duplicate these into the train, validate, and test folders, following a pattern like label.index.file_extension, e.g: rust.0.jpg, rust.1.jpg, norust.0.jpg, norust.1.jpg. Here's the folder split (80-10-10) we use:

train: 70 rust and 60 no-rust pictures

validation: 6 rust and 6 no-rust

test: 6 rust and 9 no-rust pictures

The folder structure looks like the following, under the main folder.



```
├── test
│   ├── norust
│   └── rust
├── train
│   ├── norust
│   └── rust
└── validation
    ├── norust
    └── rust
```

2.3 Training Images

Since we have small number of pictures, data augmentation may be a fundamental method to prepare our Deep Learning model. Keras comes in role here. It gives a breathtaking API (Keras ImageDataGenerator) to produce more pictures by rotating, shifting, zooming the images. All images are scaled by dividing the pixels intensities by 255. Test pictures will only inflate or deflate the accuracy. For illustration, in case the first picture is classified accurately, the coming about 100 pictures from increase of this picture will also be classified accurately. It might be the other way as well!

The following images are from our training dataset and include no rust and rust image respectively.

2.4 Creating Model and Training

Once we have arranged our training data, we will make our Convolutional Neural network model to train these pictures. Initially, we are importing the pre-trained VGG16, with weights from the ImageNettrained model. The include_top = False implies we are not including the last fully connected layers in the model. It’s time to form our Neural network model, utilizing the convolutional base (pre-trained) and include the thick layers on beat for our preparing. The primary portion of the model will act as a highlight extractor and the final layers we have fair included at the top will classify the pictures concurring to our task.

2.5 Model Validation

This is the most important step. Bringing and exposing the model to the images it has not sent before (test images) and evaluating it. We get an accuracy of 86%. The method is very simple. We stack the test picture with target size, as utilized within the show, change over the picture to Numpy cluster representation and utilize this to anticipate the yield course of the picture (likelihood >0.5 implying rust, likelihood <=0.5 meaning no nearness of rust). The result of two test test pictures is shown below,

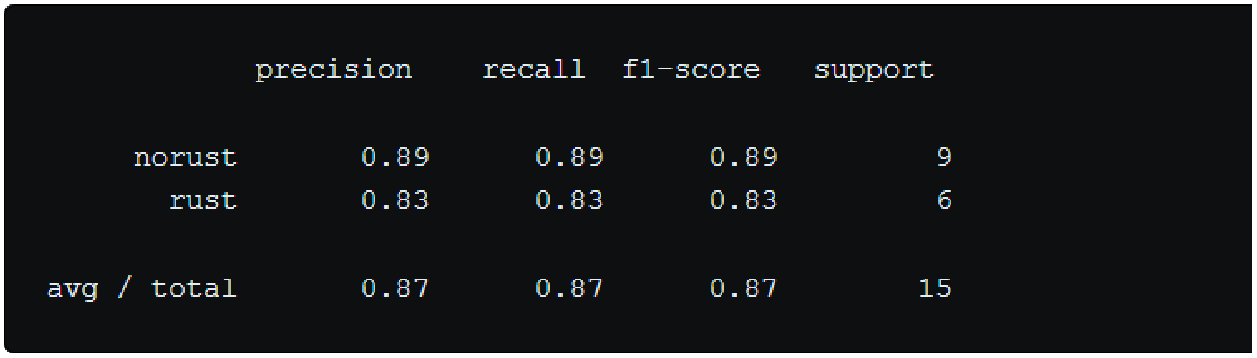
This is a rust image.

This is non-rust image.

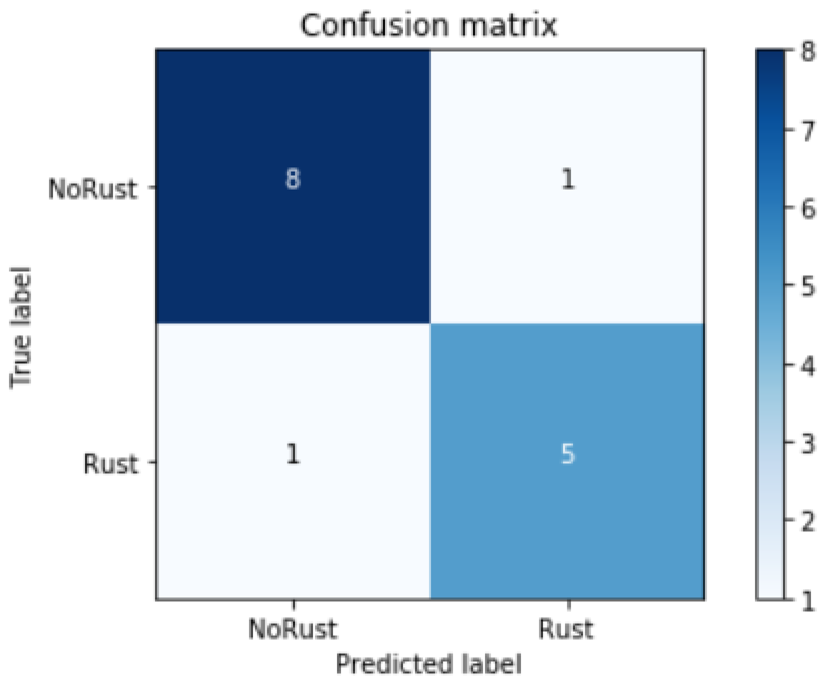
It identifies a red brown painted surface as non rust image.

We test our model on arbitrary pictures and run the forecast model, making beyond any doubt we attempt blend up">to blend up the show with reddish-brown surfaces which are 'no rust' and got beautiful great comes about. We see there are few mistakes as well. Double classification has few measures of precision past simple ‘Accuracy’. These are exactness and review. Exactness is characterized by the extent of anticipated rust pictures which are really rust(true positives) within the set of all anticipated rust pictures. Review is the anticipated rust pictures, which are actually rust(true positives) within the set of all veritable rust pictures. As able to get it Review is much more vital here as we need to distinguish all rust pictures. We want it to be 100%. In the event that the Accuracy is < 100% it implies we are labeling some 'no rust' pictures as 'rust', which is still fine. We get these by running our show on the test information. The Scikit-learn measurements library calculates the twofold classification measurements based on the real name and anticipated name.

2.6 Test Images classification report

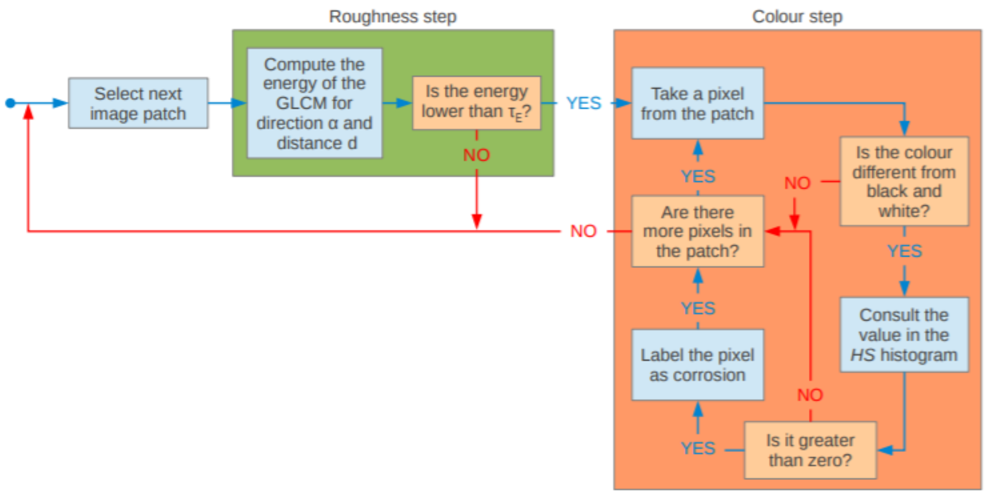


```
             precision    recall  f1-score   support

     norust       0.89      0.89      0.89         9
       rust       0.83      0.83      0.83         6

avg / total       0.87      0.87      0.87        15
```



2.7 Localization and detection of rust

Once we accurately classify a rust picture, we go to the another step of localizing and distinguishing the corroded portion within the picture by drawing a bounding box on the rust. Usually since not all the rust pictures we downloaded are of the frame where a bounding box can be drawn over the rusted portion. The Question location steps require a bit more consideration.


Roughness step
Colour step
Select next image patch
Compute the energy of the GLCM for direction α and distance d
Is the energy lower than τE?
YES
NO
Take a pixel from the patch
Is the colour different from black and white?
Are there more pixels in the patch?
Consult the value in the HS histogram
Label the pixel as corrosion
Is it greater than zero?
YES
NO

2.7.1 GLCM

GLCM Gray level Co-occurrence matrix is used to detect texture First color image is converted into gray scale which facilitate detection of different textures available in the image based on gray – level pattern existing in the texture. Joint probability is calculated of the grey level pairs occurring at two points. With joint probability and number of pixel it is easy to calculate the normalized GLCM. After calculating normalized GLCM, rust particle with textures can be separated creating a mask. This mask is implemented to give only rusted part and rest of the non-rusted past is black.

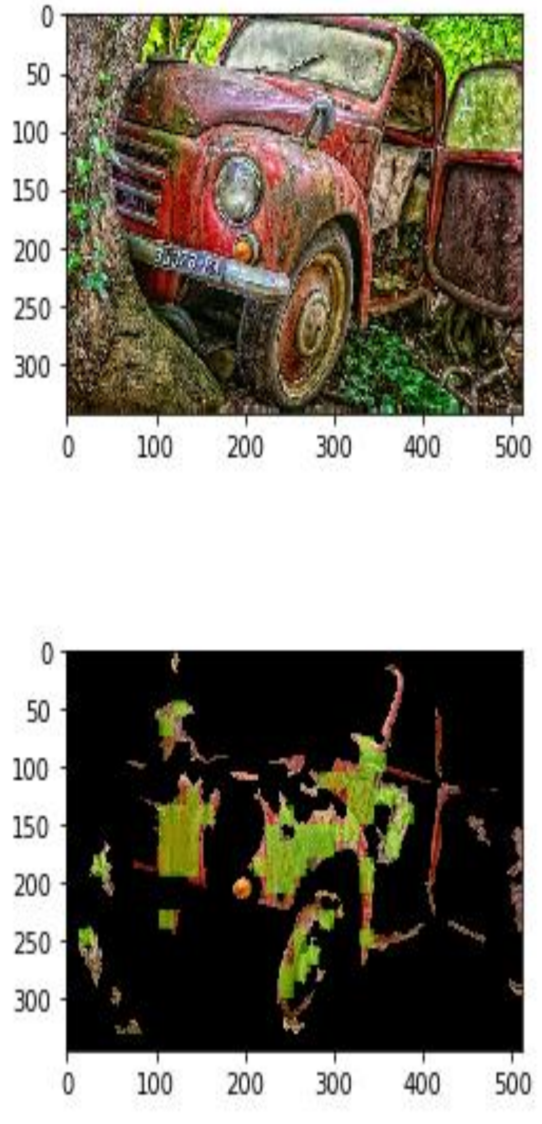
2.7.2 Why GLCM

Deep learning only classify the image in rusted and non-rusted category. But to localize and detect rust particle we used GLCM. Once the image has been classified as rusted image it is GLCM comes into role. If image is not rusted it is classified as “Not rusted” and no further action is required.

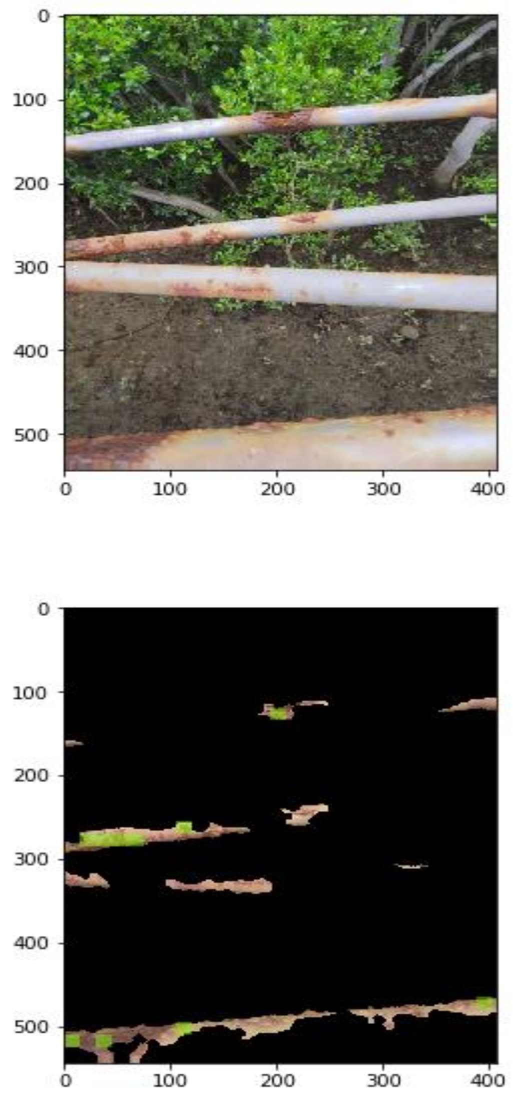
2.8 Results

The following images are the outputs with original image and resulting image combined. We can evaluate it for different images, following are one of them.

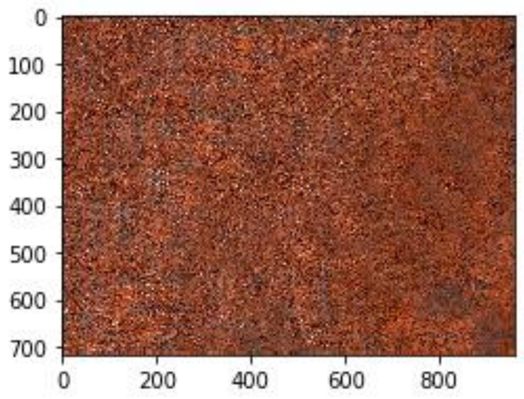
Example 1:



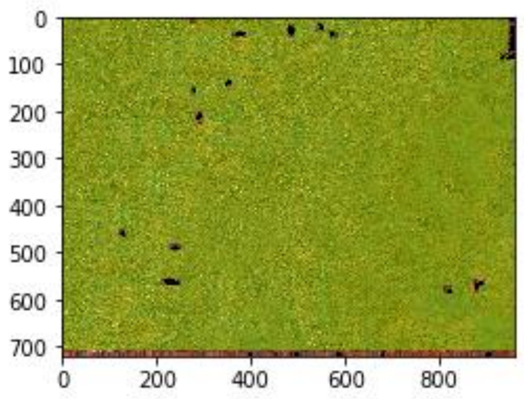
Example 2:

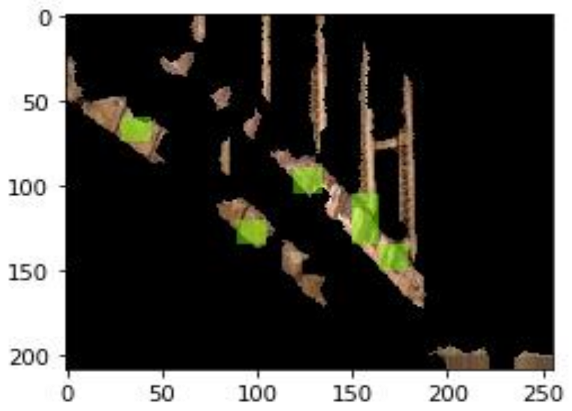


Example 3



Example 4





Implementation

Project contains two program one Corrosion Detection and another is Red_particles. Deep learning is implemented in Corrosion detection. Localizing the rust particle is implemented in Red_particles. Corrosion Detection imports the Red_particles. If image is classified as rusted than only red_particles program will come into role otherwise not.

Conclusion

We displayed a comparison between two diverse models for rust discovery, one based on red component detection utilizing OpenCV library, whereas the second one utilizing Deep Learning models. We trained the model with more than 150 pictures, finding out that the deep learning model performs way better in a genuine case situation. However for a genuine application, it may be advantageous to incorporate both the frameworks, with the OpenCV demonstrate utilized fair for removing the false positives some time recently they are passed to the deep Learning strategy. Too, the OpenCV based calculation may moreover be valuable for the classification of images where the Deep Learning calculation has low certainty/confidence. The result can be seen as sparing in review taken a toll, superior quality items, and discovery of a imperfection at an early stage subsequently reducing cost of re-work. In real generation, the prepared model can be integrated with an IoT framework driving to program isolation of great and inadequate parts. Such AI empowered cleverly Assessment frameworks are going to be a popular in close future and Deep Learning is reaching to play a necessarily part in these. In future work we are looking forward to refine the model and prepare it with an unused and bigger dataset of pictures, which we accept would improve the precision of the Deep Learning model. In this way, we'll do a few field testing using real time video from real infrastructure assessments.

References

1. Deep Convolution Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection by Kasthurirangan Gopalakrishan, S.K Khaitan and Ankit Aggarwal. https://www.researchgate.net/publication/319952138_Deep_Convolutional_Neural_Networks_with_transfer_learning_for_computer_vision-based_data-driven_pavement_distress_detection

2. Using deep learning and Tensor flow object detection API for Corrosion detection and localization. https://blog.floydhub.com/localize-and-detect-corrosion-with-tensorflow-object-detection-api/

3. Corrosion detection using A.I: A comparison of standard computer vision technique and deep learning model by Luca Petricca, Tomas Moss, Gonzola Figueroa and Stian Broen. https://www.researchgate.net/publication/303563962_Corrosion_Detection_Using_AI_A_Comparison_of_Standard_Computer_Vision_Techniques_and_Deep_Learning_Model

4. Image processing based detection of pipe corrosion using texture analysis and metaheuristics by Nhat-Duc Hoang and Van-Duc Tran https://www.hindawi.com/journals/cin/2019/8097213/

5. Corrosion detection for Automated Visual Inspection by Francisco Bonnnin- Pascual and Alberto Ortiz. https://cdn.intechopen.com/pdfs/46235.pdf