**Mobile screen crack detection**

By- chandrashekhar (puresoftware pvt ltd)

**Technology-** pathon3.7,YOLO machime learning,CV2, keras,tensor flow

**SW**:

Python64 bit installation

Anaconda64 bit installation

**Approach1**:

1. Make a folder of scratch and non scratch

2. Read dataset of scratch and non scratch (make a class)

3. Define a model- CNN model (keras with tensor flow)

4. Train dataset (scratch and non scratch)

5. Make a validation data – a new data (predict from model)

6. Edge detection – choose best algorithm)

7. take a codec and circle out crack area

Binary classifier

**Approach2**:

1. labeled dataset (circle out scracth data). scratch and non scratch

2. training data through label -mg

3. model R-C nn

4. Make a validation data – a new data (predict from model)

5. Edge detection – choose best algorithm)

6. take a codec and circle out crack area

self classifier

**Summery:**

Three requirements are the principles for developing the automatic crack detection and classification method.

* **Detection rate**: crack detection and classification approach must guarantee that the vast majority of crack length in the original image is detected in the last output results.
* **Detection accuracy**: crack detection accuracy must be acceptable, which means that the misidentified objects(camera icon,logo etc) should be removed as much as possible.
* **Detection efficiency**: in practical application, cracked images are collected, making the processing of so many images for train and test our data. image processing process must be fast and efficient.

In gray-scale images, cracks present themselves as dark regions with local minimum gray-level components. Morphological image processing operations have an advantage in segmenting relevant structures without complex iterative calculations.

After the crack **segmentation** process, there may still be many misidentified objects that appear as cracks. In attempting to distinguish between cracks and unexpected irrelevant objects, feature extraction becomes the key problem to be solved.

Distance transformation is used in shape segmentation, which use distance to map spatial shape into a probability sequence by distance. A shape descriptor called a **distance histogram** is proposed to perceive the difference between cracks and irregular objects. The standard deviation of the distance histogram is an effective criterion for describing the degree of irregularity of a spatial shape. Along with the standard deviation of the distance histogram, two additional numerical features are used as the basis for classifying the cracks. With a **pattern recognition algorithm** or a **thresholding**.

**Techniques**:

1)signal processing methods in combination with other techniques to detect cracks or other defects in the structure surface. (noise)

2)Four edge-detection algorithms (Fast Haar transform (FHT)

3)**Fast Fourier transform** (FFT)

4)Sobel filtering

5)Canny filtering

**Classification Algorithms**:

To verify the performance feature extraction method, different classification algorithms will used to classify the new observed candidate objects.

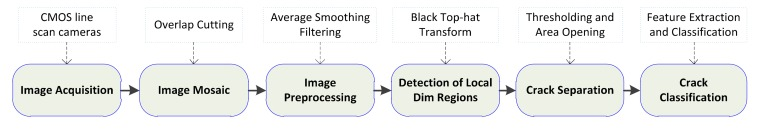
Extreme Learning Machine (ELM)

Radial basis function neural network (RBF)

Support Vector Machine (SVM)

K Nearest Neighbors algorithm (KNN).

**Propsed process:**



**Keras API**:

**ReLu**: rectified linear unit. The rectifier function is an activation function f(x) = Max(0, x) which can be used by neurons just like any other activation function, a node using the rectifier activation function is called a ReLu node.

The main reason that it is used is because of how efficiently it can be computed compared to more conventional activation functions like the **sigmoid** and

**hyperbolic tangent**, without making a significant difference to generalisation accuracy. The rectifier activation function is used instead of a

linear activation function to add non-linearity to the network, otherwise the network would only ever be able to compute a linear function.

training a network when ReLU is used would be faster.

**Dropout**: The reason that randomly ignoring nodes is useful is because it prevents inter-dependencies from emerging between nodes

(I.e. nodes do not learn functions which rely on input values from another node), this allows the network to learn more a more robust relationship.

Implementing dropout has much the same effect as taking the average from a committee of networks, however the cost is significantly less in both time

and storage required.

**Advantage:**

* Sigmoid: not blowing up activation
* Relu : not vanishing gradient
* Relu : More computationally efficient to compute than Sigmoid like functions since Relu just needs to pick max(0,xx) and not perform expensive exponential operations as in Sigmoids
* Relu : In practice, networks with Relu tend to show better convergence performance than sigmoid. ([Krizhevsky et al.](http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf))

**Disadvantage:**

* Sigmoid: tend to vanish gradient (cause there is a mechanism to reduce the gradient as "aa" increase, where "aa" is the input of a sigmoid function. Gradient of Sigmoid: S′(a)=S(a)(1−S(a))S′(a)=S(a)(1−S(a)). When "aa" grows to infinite large , S′(a)=S(a)(1−S(a))=1×(1−1)=0S′(a)=S(a)(1−S(a))=1×(1−1)=0).
* Relu : tend to blow up activation (there is no mechanism to constrain the output of the neuron, as "aa" itself is the output)
* Relu : Dying Relu problem - if too many activations get below zero then most of the units(neurons) in network with Relu will simply output zero, in other words, die and thereby prohibiting learning.(This can be handled, to some extent, by using Leaky-Relu instead.)

**Vanishing Gradients**

the bigger the input (in absolute value) the smaller the gradient of the sigmoid function. But, probably an even more important effect is that the derivative of the sigmoid function is ALWAYS smaller than one. In fact it is at most 0.25!

The down side of this is that if you have many layers, you will multiply these gradients, and the product of many smaller than 1 values goes to zero very quickly.

Since the state of the art of for Deep Learning has shown that more layers helps a lot, then this disadvantage of the Sigmoid function is a game killer. You just can't do Deep Learning with Sigmoid.

On the other hand the gradient of the ReLu function is either 0 for a<0 or 1 for a>0. That means that you can put as many layers as you like, because multiplying the gradients will neither vanish nor explode.

