By Sourav Thakur, E-mail : Souravthakur1000@outlook.com

Crack Detection Using Machine Learning And Computer Vision

Abstract

Routine inspection and maintenance of metal infrastructure must be performed to ensure structural integrity and prevent structural failures which may cause damage to surrounding infrastructure, environmental pollution and potential loss-of-life. Typically, regular visual inspection is conducted to identify various defects due to environmental exposure during the service life of the structure (such as cracks, loss of material, rusting of metal bindings, etc). Visual inspection can provide a wealth of information that may lead to positive identification of the cause of observed distress. However, its effectiveness depends on the knowledge and experience of the investigator and is prone to human error. Additionally inspection of large structures such as dams, bridges and tall buildings can be prohibitively risky and difficult due to hard-to-reach facets. Through this paper we illustrate the use of Artificial Intelligence (AI) techniques to automate the inspection process and for identification of defects (surface cracks) efficiently. The surface crack detection solution described below is designed for deployment on embedded platforms and uses deep learning algorithms to detect and classify structural cracks on metal surfaces.

Introduction

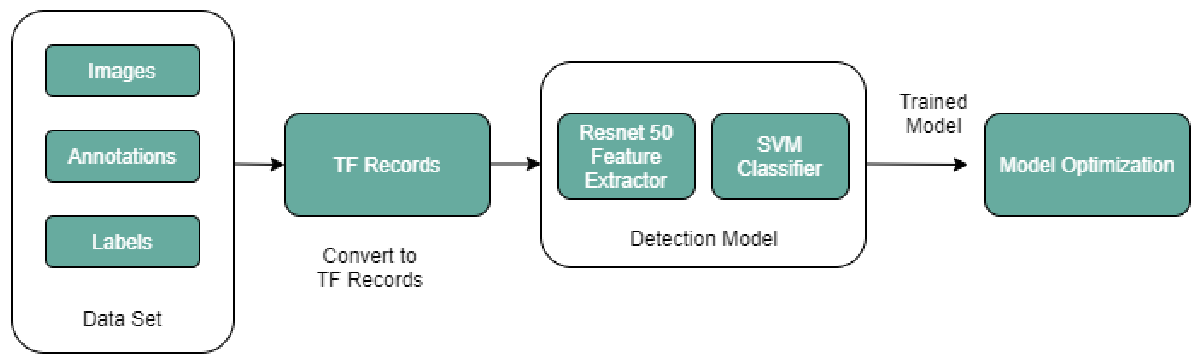
An alternative approach to manual inspection is an automatic crack detection system that works by leveraging methods from computer vision (e.g., gradient thresholding and edge detection).This can lead to automated inspection systems which cater to a select set of surface types and features. Here, we explore the potential of a deep learning-based crack detection system that can be used on multiple surface and crack types. Convolutional Neural Networks (CNNs) are a class of deep-learning neural networks that are frequently used in image classification and segmentation (pixel-wise classification) tasks. They are known for adaptability, particularly in cases where the neural network topology has many layers, which has been shown to allow them to learn low-level features (e.g., lines, edges and angles) and high-level features (e.g., curved surfaces and textures). Our approach to automatic crack detection leverages a topology belonging to this class CNN with ResNet as a backend feature extractor, enabling it to detect and draw a bounding box around any crack identified in an image. To train our model, we randomly partitioned the dataset into 95% for training and 5% for evaluation. To compensate for the limited size of available dataset, a CNN Resnet model (pre-trained - a large-scale object detection, segmentation, and captioning dataset) has been re-trained on all layers with our training dataset.

Surface Crack Detection Process

Our approach to surface crack detection has five stages:

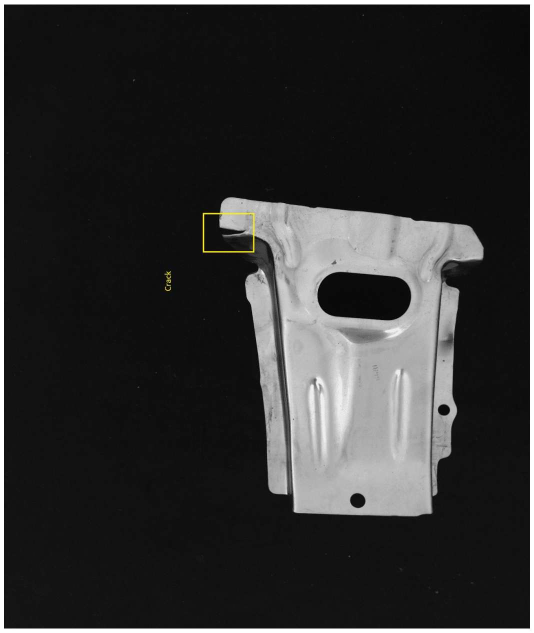
- Annotating and labelling the images
- Converting images into the format used by TensorFlow

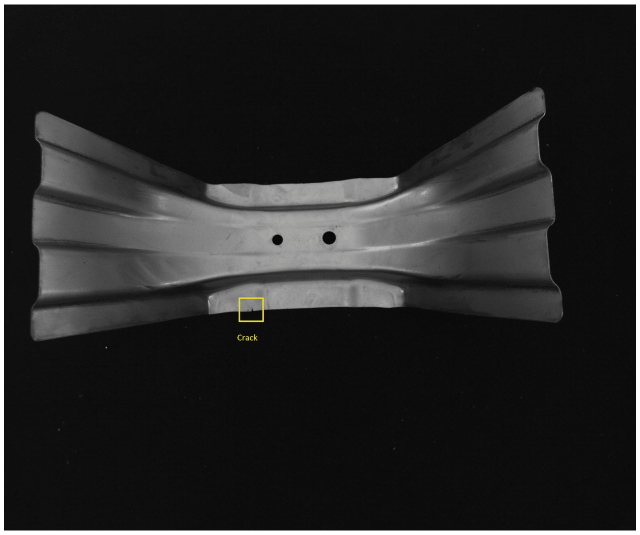
• Re-training the pre-trained model with the new data
• Computing the evaluation metrics
• Optimizing the model training



Workflow of the Crack Detection Process

Annotations and Labelling





Images were annotated by drawing bounding boxes around any cracks which were visible in the images. A bounding box is a small rectangle enclosing the whole crack.

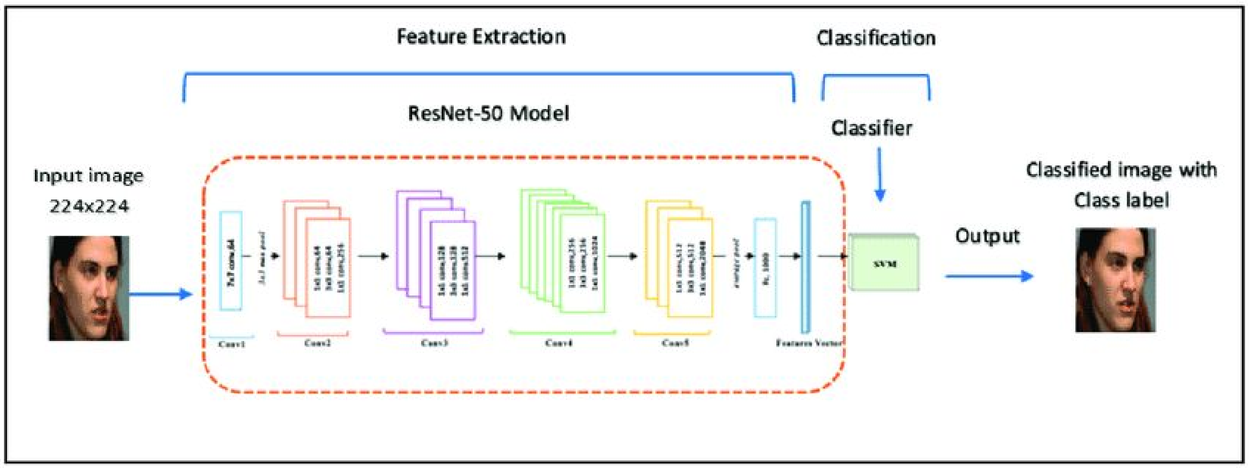
Data Preparation

We will convert the images into the format used by tensorflow for feeding it to the resnet.

Feature Extraction

Model Selection

For this problem, let's build a Convolution Neural Network (CNN) in Pytorch. Since we have a limited number of images, we will use a pretrained network as a starting point and use image augmentations to further improve accuracy. Image augmentations allow us to do transformations like — vertical and horizontal flip, rotation and brightness changes significantly increasing the sample and helping the model generalize.



Shuffle and Split input data into Train and Val

The data downloaded will have 2 folders one for Positive and one for Negative. We need to split this into train and val. The code snippet below will create new folders for train and val and randomly shuffle 85% of the data into train and rest into val.

Apply Transformations

Pytorch makes it easy to apply data transformations which can augment training data and help the model generalize. The transformations I chose were random rotation, random horizontal and vertical flip as well as random color jitter. Also, each channel is divided by 255 and then normalized. This helps with the neural network training.

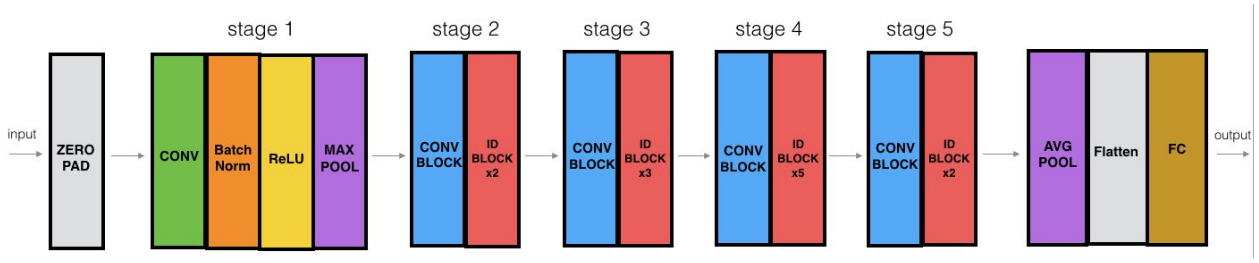
Pretrained Model

We are using a Resnet 50 model pretrained on ImageNet to jump start the model. As shown below the ResNet50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters. We are going to freeze all these weights and 2 more fully connected layers — The first layer has 128 neurons in the output and the second layer has 2 neurons in the output which are the final predictions.

ResNet-50 is a convolutional neural network that is 50 layers deep. We can load a pre trained version of the network trained on more than a million images from the ImageNet database. ResNet short for

Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of the ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks were difficult due to the problem of vanishing gradients.

We need to convert the images into forms that tensorflow can understand i.e tensors.
Then we'll feed the input into the resnet and use resnet 50 network to calculate the model accuracy on the training set and validation set.



We'll use keras to implement the resnet. We used Adam as the optimizer and train the model.

Steps:

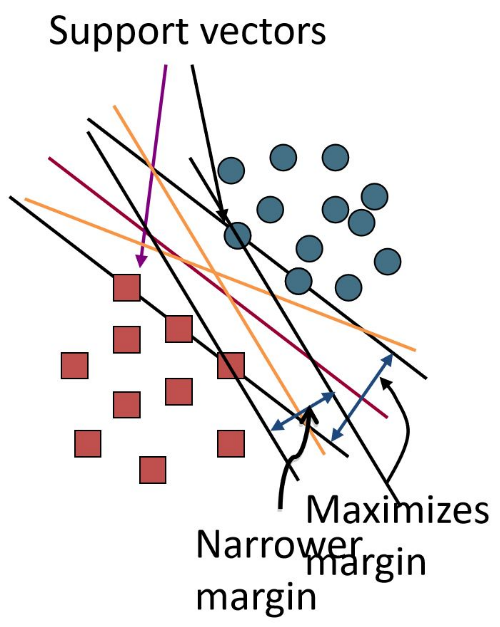
1. Convert data to tensors.
2. Fit the resnet model
3. Optimize the model and use hyper parameter tuning

Classifier

Support Vector Machine:

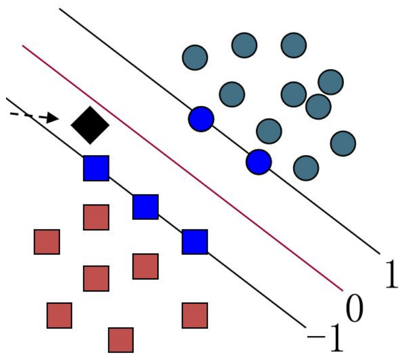
We can use SVM along with ResNet to classify the images. SVM will be used as the last layers of the ResNet and will be trained on the new images.

ResNet layers-----> SVM-----> Output



- SVMs maximize the margin around the separating hyperplane.
- A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, the support vectors.
- Solving SVMs is a quadratic programming problem
- Seen by many as the most successful current text classification method*

Classification with SVMs



- Given a new point x, we can score its projection onto the hyperplane normal:
- I.e., compute score: wTx + b = ΣαiyixiTx + b
- Decide class based on whether < or > 0
- Can set confidence threshold t.

    Score > t: yes
    Score < -t: no
    Else: don’t know

Model Training and Prediction on Real Images
We will use transfer learning to then train the model on the training data set while measuring loss and accuracy on the validation set.

Integrating the Trained CNN into a Smartphone Application
The trained CNN can be used to identify the defect in a new image, and the popularity of smartphones provides an opportunity to mobile public to detect cracks. For this purpose, based on the framework of Core ML, the trained CNN model is integrated into a smartphone application to detect cracks in practice conveniently. During the integrating process, the Xcode an integrated development environment is utilized to create an application with Swift programming language. The generated application named Crack Detector is installed on a phone. Notably, it can predict not only local photos but also a new photo taken of the surface at that time. The result exposes the great performance of the trained CNN. In addition, this smartphone application can draw more attention to crack detection handily.