Surface Crack Detection

Pavan Choudhari
Northeastern University, 360 Huntington Ave, Boston, Massachusetts, 02215
choudhari.p@northeastern.edu

Abstract

In recent years, research and applications in artificial intelligence have gained tremendous support from all around the world, and it has helped solve many problems like pose estimation, object detection, image classification, natural language understanding, etc. The office of consumer affairs and business regulation is responsible for the safety of our homes in Boston. The officers are responsible for performing physical checks and assessments to attest to the safety of the house. The COVID-19 pandemic has spread havoc around the world, and it is becoming increasingly difficult for the strained city resources to be directed towards house inspections. Therefore, this paper focuses on using concepts of Machine Learning and Deep Learning to analyze and classify images into the crack and non-crack categories to help expedite the safety assessment process for homes in Boston. This problem is difficult to solve using classical image processing algorithms due to the background noise, texture, blocking objects, and building anomalies in the images. The problem at hand is a binary classification task and within the scope of this study, a comparative analysis of the models will help future researchers avoid pitfalls in building an accurate and precise classification model.

Introduction

The city of Boston has spent huge sums of money in the past to reconstruct houses that have their base crumbling due to the poor-quality concrete used in construction of the houses [1,2]. The establishment of the city dates back 15th century, and concrete was introduced in the building process in the 18th century. The material concrete is second most used material after water, and has many advantages like, low maintenance, it does not rust, rot, or burn, wind resistant, water resistant, non-combustible, and provides effective soundproofing. But, the downside in using this material in homes is that over time it will start to develop cracks due to chemical contamination, efflorescence, and environmental degradation [3]. These cracks are detrimental to the inhabitants of the building and for the property value. Therefore, the Massachusetts government has an office that provides the facility of home inspection to determine the safety of an infrastructure. The inspectors physically examine the building for cracks and reviews the readily accessible exposed portions of the structure of the home, including the roof, the attic, walls, ceilings, floors, windows, doors, basement, and foundation. This is an arduous and time-consuming process. The advent of Artificial Intelligence has helped computers gain human like capabilities. They have been successfully used to classify objects, estimate pose, detect objects, natural language capabilities, etc. Over the years, many datasets have been put together to help researchers provide performance baselines for the Artificial Intelligence algorithms. These datasets also focus on providing large collection of images to solve a particular computer centric difficult task. The MNIST dataset is a collection of handwritten digits which provides a baseline for image classification models. The CIFAR-10 dataset is a collection of images of 10 objects which provides a baseline for object detection models. The image classification task is a difficult task, because of the background noise in the images, uneven texture, uneven light exposure, blocking objects and non-endangering cracks. Fig 1 shows the problematic images. Despite the problems, the models and research methods used in solving the MNIST dataset, is particularly helpful for this research as it will provide techniques that will help automate and expedite the process of crack detection. As, deep learning models like Convolution Neural Network (CNN) has shown promising results on the MNIST problem, it is expected that it will outperform all the machine learning models on our research dataset.

Figure 1. Problematic Images in crack detection. 1) Asphalt filled crack surface. 2) Shadow obstructing the crack. 3) Worn out paint surface.

Background

The problem that we are trying to solve is an image classification task. Traditional image processing methods proposed in building autonomous models have relied on the image analysis expert to extract features from images that will help easily classify the images [4]. In these studies, experts and researchers use image filters and morphology methods to transform the images into something that is easier for the algorithms to classify. The Local Binary Pattern technique is one such traditional method used to detect cracks in images. It identifies the textures in images by converting the images to grayscale and then applying rotation invariant operators to detect the difference in textures [5]. Another similar approach is to use localized search to determine if the area is smooth or rough. This will give more information in determining the local structure [6]. One particular filter, the Gabor filter, can perform pavement crack detection with a 95% precision [7].

Vijayrani et al [8], have used Sobel and Canny edge detection methods for image analysis. This method outlines the edges in the image, which can help reduce distortion and background noise. Prah et al [9], have provided conclusive evidence in their paper, that the Sobel and Canny algorithm are efficient for different types of surface. The results state that the Canny filter worked well on asphaltic concrete, and the Sobel filter works well on Portland cement concrete. These extracted features are then fed to machine learning models for the classification task. Some good examples would be the research work in Adhikari et al [10] classification of pavement cracks, and Wu et al [4] for sewer pipes crack detection.

These methods as outlined in various studies need the help of researchers and experts to experiment with different parameter values to extract features that aid in the classification tasks. But these methods are not fully autonomous. The study by Zhang et al [11], improves on these filters by applying machine learning algorithms, such as K Nearest Neighbors (KNN), Support Vector Machines (SVM), radial basis function (RBF), and Extreme Learning Machine (ELM) on classifying cracks in subway tunnels.

The Fig. 2 shows Machine Learning Research Interest in crack detection over a decade, and this shows waning interest as compared to Deep Learning research which has boomed in the last 5 years [12]. The study by Zhang et al [13] in 2016 using Convolution Neural Network set the groundwork for future deep learning research in this area. He proposed using fully connected layers, with a 6-layer deep neural network, for classifying cracks on roads. The researchers used 600,000 images to train the model and 200,000 to test the model and the resulting F1 score of 0.8965. Given, the complexity of the calculations and the size of the dataset, the attained score is very good.

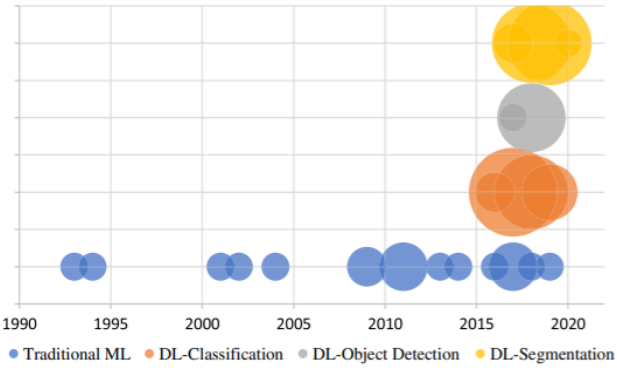

Figure 2. Shows the research interest in crack classification from 1990 to 2020. Traditional Machine Learning has continued interest over the years, but has decreased in size, after 2016 when Deep Learning was introduced to tackle this problem.

Similar, to our problem, Cha et al [14], used a 4-layer convolution neural network to assess cracks on building surfaces with varying contrasts and illuminations. This study is more diverse than the aforementioned method because it takes into account the more diverse conditions. They trained a model on 40,000 images, but on a 54 full resolution image.

It is important to note here that the more the resolution of the image, the higher background diversity it will capture. They outlined that it is important to use more than 10,000 images to train a base model. They scored a mean accuracy of 0.9683.

Related Work

The research done in this paper is based on the works done on the SDNET-2018 dataset [15]. The SDNET-2018 dataset contains 56,000 images of cracked and non-cracked concrete images of bridge decks, walls, and pavements. The paper introduces a sophisticated model to classify images, but we will build a base CNN model for the scope of this research. Tien-Thinh Le [16], in their recent work on our dataset, shows a 10-layer CNN model with a highest accuracy of 99.7% and a F1 score of 99.5%. The model has too many parameters to train and needs a significant amount of RAM and GPU power to train their model. This paper proposes an improved version of this model with a 5-layer CNN architecture achieving a similar accuracy of 99.6 - 99.8%.

The paper also researches on the underrated Machine Learning models for crack detection. From Fig 2 analysis it is evident that in the recent years machine learning models are losing popularity over deep learning models, but from my analysis Extra Tree Classifier achieves a 99.16% accuracy, with an F1 score of 99.2%. This comparatively is a good performance with low training times and does not need a GPU to train the algorithm. Due to its less complexity and faster training times of a few seconds it has the potential to be used in mobile and web applications. The paper goes on to experiment with the idea of using different image filters before applying the Machine Learning models. There are no research papers that experiment this method for this surface crack dataset, and the results will help future researchers either implement this ideology into their research or avoid pitfalls.

Other methods to solve this problem are more sophisticated deep learning models like AlexNet2018 and VGG-16 which are more robust to noise and perform exceedingly well on these tasks. They are left out of the scope of this research, because of the time constraint and limited resources.

Project Description

Dataset: This research paper works on the Surface Crack Detection dataset [17]. The dataset has a total of 40000 images labelled as cracked and non-cracked. There are 20000 cracked surface images and 20000 non-cracked surface images. The pixel dimension is 227x227 with RGB channels. The dataset is generated from 458 high-resolution images, images of the walls and floors of several concrete images around the METU campus. The images have an added complexity of contrasting illuminations and textures. No data augmentation in terms of random rotation or flipping is applied. An example of the dataset is shown in fig. 3.

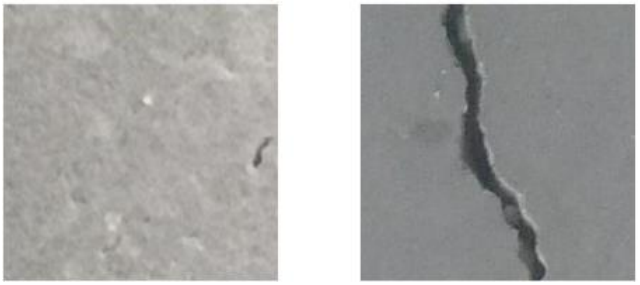

Figure 3. 1) Non-cracked image surface. 2) Cracked Image surface.

Data Preprocessing: The data needs to be pre-processed before applying any analysis and Machine Learning or Deep Learning model. The RGB channel of the images do not add any significant information, and only add to the complexity of the model. Therefore, it is necessary to convert the images to grayscale. To apply any machine learning model, it is important to scale down the size of the image. The model perceives each pixel of the image as a feature, therefore if we use the original image, we will have 51529 features. This is a very large number of feature set for the Machine Learning models, therefore we reduce the image to 50x50 pixel dimension, giving a 90% reduction in the feature space. Fig 4 shows the scaling of the image.

Figure 4. 1) 227x227 original Image. 2) 50x50 scale image.

Each 2-D image is changed into a 1-D vector. Finally, before applying any model the dataset is split into training and test set. The training set has 28000 number of samples, and the test set has 12000 number of samples forming a 70-30 split ratio.

Machine Learning models: This is an overview of the different machine learning models that the paper will

implement. The base model is a Logistic Regression model. For a given dataset D(X,Y), where X is the input matrix and Y is the target matrix, the objective of a logistic regression model is to find a relationship to predict the target. The linear relationship is given as,

$$a = w_0 + w_1x_1 + w_2x_2 + ...w_nx_n$$

Eq 1.

The model will try to learn the weights of the equation, by minimizing the cost function,

$$cost(w) = (-1/m)\sum_{i=1}^{i=m} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

Eq 2.

The Eq. 2 is the logistic loss. The predicted value in Eq. 2 is given by a link function,

$$\hat{yi} = 1/(1 + e^{-a})$$

Eq 3.

We train the model by minimizing the cost function, that is by updating the weights as,

$$w_i = w_j - (\alpha * dw_j)$$

Eq 4.

$$dw_j = \sum_{i=1}^{i=n} (\hat{y} - y)x_j^i$$

Eq 5.

As the model trains the loss function converges, and the resulting model is a trained model. To check the efficiency of the model we define the success metrics and measure its performance on test set [18].

Decision Tree Classifier

A decision tree is a decision support tool that uses tree-like model to make decisions based on reducing heterogeneity in the data. The algorithm recursively decides on a splitting node, which is chosen by the attribute that contributes to the maximum information gain. The algorithm stops when the data after the split is homogenous, or if the tree is pruned to reduce overfitting. The information captured or entropy is given by the equation,

$$Info(D) = -\sum^{m} p_i \log_2(p_i)$$

Eq 6.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Eq 7.

The Eq 6, is the entropy formula, and Eq 7, gives the entropy of the attribute. The information gain is given as the amount of reduction in entropy or impurity from the parent to the child, based on the splitting attribute. The Eq. 8 shows the information gain mathematical equation.

$$Gain(A) = Info(D) - Info_A(D)$$

Eq 8.

There are many other ways to measure the impurity, for example the Gini index is another method that captures the goodness of a split. To calculate the Gini index, use the Eq 9,

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

Eq 9.

The algorithm is given in figure 5,

INPUT: S, **where** S = *set of classified instances*
OUTPUT: *Decision Tree*
Require: $S \neq \varnothing$, $num\_attributes > 0$

```
procedure BUILDTREE
    repeat
        maxGain ← 0
        splitA ← null
        e ← Entropy(Attributes)
        for all Attributes a in S do
            gain ← InformationGain(a, e)
            if gain > maxGain then
                maxGain ← gain
                splitA ← a
            end if
        end for
        Partition(S, splitA)
    until all partitions processed
end procedure
```

Figure 5. Decision Tree Algorithm.

The decision tree is prone to overfitting but is the simplest machine learning model that provides human interpretability of the working of the model.

Adaboost

The Adaboost algorithm is a boosting algorithm that set weights for both the classifiers and data sample in such a way that emphasis is given on the wrongly classified samples. The base classifier can be any model, but mostly single depth decision tree, also known as, stump is used. The main idea is to combine weak classifiers into a single strong classifier that will always perform better than a single deep decision tree. The algorithm is shown in Figure 6,

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 6. Adaboost Algorithm.

The algorithm combines each stump $h_t(x)$ and it is multiplied by a factor alpha which is the power of the decision stump. The lower the error in classification the higher the alpha value, and the better that classifier. Getting a higher error rate, will increase the weights of the misclassified samples, and decrease the weights of the correctly classified samples, and the new training set is derived from this weighted sample set. At the end, all the basic classifiers are combined to form the model.

Random Forest Classifier

Random Forest Classifier is one of the most successful ensembles method, which is based on the same concept of the decision tree but takes a slightly different approach. It is popularly used for classifying data of high dimensions and skewed problems [19]. The random forest classifier tries to solve the problem of overfitting in the decision tree. A slight change in the dataset can give a tremendously different decision tree, and this stems from the idea, that if there is an error close to the root, that error propagates through the entire tree, giving drastically different results. The algorithm works by training a decision tree on a sample set of size N, drawn from the original set with replacement. At each decision split, only $\sqrt{m}$ or $\sqrt{m}/2$ feature subset are used and the best (information gain) is selected as the splitting node. The trees are aggregated together to form the final classification decision for a new instance, and to test the performance of the model, 1/3 of the data is set aside, and this is called the out-of-bag set. The prediction error of the bagged ensemble is estimated by computing predictions for each tree on its out-of-bag observations; averaging these predictions over the entire ensemble for each observation and then comparing the predicted out-of-bag response with the true value at this observation. Bagging works by reducing the variance of an unbiased base learner, such as a decision tree. This technique tends to improve the predictive power of the ensemble, as the random selection of features reduces the correlation between trees in the ensemble.

Extra Tree Classifier

The extra tree classifier is very similar to the random forest classifier, but it differs on the following points:

1) The extra tree classifier does not train on bootstrap samples but uses the entire training dataset.
2) The extra tree classifier chooses $\sqrt{m}$ or $\sqrt{m}/2$ feature subset, but decides the decision splitting criteria at random.

Therefore, the extra tree classifier has lower variance than random forest classifier, and converges faster.

The study does not dive deep into researching different and better Deep Learning models for the dataset. It only intends to construct a better CNN model than the one proposed by Tien-Thinh Le [16].

Deep Learning model

Convolution Neural Network

The convolution neural network in contrast to the machine learning methods does the feature extraction automatically. The convolution layer is responsible for applying a set of kernels on top of the image to give a set of 2-D matrices. These kernels are updates as the neural network trains, as they are the weights of the model. The Eq. 10 is for a single output matrix,

$$A_j = f(\sum_{i=1}^{N} I_i * K_{i,j} + B_j)$$

Eq 10.

Next, the output matrix is passed to a pooling layer. Here, it reduces the dimension of the matrix. The most common algorithms for pooling is max-pooling and average-pooling. Finally, ReLU activation function is used to mimic the firing of neuron in the brain. The output of the max pooling is passed to the ReLU activation function, which is given as the equation 11,

$$f(x) = max(0, x)$$

Eq 11.

The training is complete with the final step of using back propagation. Instead of updating weights after each forward pass, it is better to update the weights by using batches. The update formula is shown in Eq 12.,

$$\Delta\omega_i(t+1) = \omega_i(t) - \eta\frac{\partial E}{\partial w_i} + \alpha\Delta\omega_i(t) - \lambda\eta\omega_i$$

Eq 12.

In Eq 12, the first term is the current weight vector, the next term is the learning rate multiplied by the gradient of the error term with respect to the weight vector, and the next term is the momentum term that helps to speed up learning, and last term is the weight decay term to reduce overfitting.

The CNN architecture proposed in this paper is a 5-layer architecture as shown in the figure 6 below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 120, 120, 64)      640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 60, 60, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 28800)             0
_________________________________________________________________
dense (Dense)                (None, 256)               7373056
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 7,486,018
Trainable params: 7,485,506
Non-trainable params: 512
_________________________________________________________________
```

Figure 7. Model Summary

Model Evaluation

Since, we have the ground truth and the predictions from the models, we can define a set of metrics that will evaluate the performance of the models. The positive class is the cracked image surfaces that we are modeling to classify. The following definitions are important to note:

1) True Positive: This is the case when the ground truth is a crack image and prediction is for a crack image.
2) True Negative: This is the case when the ground truth is a non-crack image and prediction is for a non-crack image.
3) False Positive: This is the case when the ground truth is a non-crack image and prediction is for a crack image.
4) False Negative: This is the case when the ground truth is a crack image and prediction is for a non-crack image.

Based on these components' metrics are developed as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Eq 12.

$$Precision = \frac{TP}{TP + FP}$$

Eq 13.

$$Recall = \frac{TP}{TP + FN}$$

Eq 14.

$$F1 = 2 \times \frac{Precision + Recall}{Precision \times Recall}$$

Eq 15.

These are the commonly used metrics to evaluate and compare different models.

Empirical Results

A total of seven models were trained on the surface crack dataset. The dataset was split into 28000 images for training set, and 12000 for the test set. For, the Machine Learning models, an additional step was done to convert the 2-D matrices into vectors and stored into a csv file. The Deep Learning models needed the image to be resized to 120x120 as we designed the convolution input layer this way. The models we train are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier, Gradient Boosting Classifier, Extra Tree Classifier and CNN. These models were trained on Google Colab, with 12GB RAM, and Tesla K80 GPU. The training times of the models are as shown in table 1 below.

Model	Runtime in seconds
Logistic Regression	115.17
Decision Tree Classifier	183.54
Random Forest Classifier	161.5
AdaBoost Classifier	184.21
Gradient Boosting Classifier	1072.73
Extra Tree Classifier	45.35
Convolution Neural Network	300

Table 1. Runtime of different models.

All the Machine Learning models were trained with their default parameters in sklearn python package [20], and the parameters for the best machine learning model are shown in the table 2 below.

Parameters	Value
Split criteria	Gini Index
Minimum Samples Leaf	1
Minimum Samples Split	2
Number of base estimators	100
Random state	7

Table 2. Extra Tree Classifier Model Parameters.

To get an accurate model with minimum chances of overfitting, 5-fold cross validation was performed with the Extra Tree Classifier model. The accuracy of the model increased by a little to 99.24%, and the F1 score was 99.1%. The resulting confusion matrix is show in the figure 8,

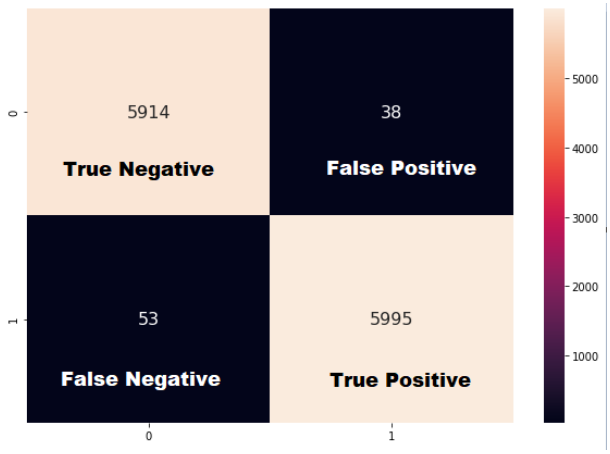

Figure 8. Cross Validation confusion matrix

The ROC plot in figure 9, shows that the model has performed very well on the task.

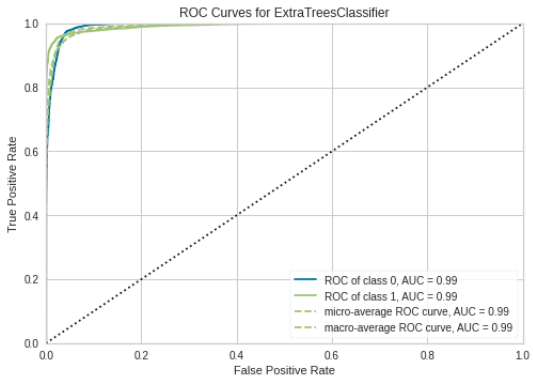

Figure 9. ROC plot for Extra Tree Classifier

The model performance metric for all the Machine Learning models is shown in table 3.

The Convolution Neural Network has a training accuracy of 99.4%.

The accuracy curve in figure 10, shows that the model is overfitting after 6 epochs.

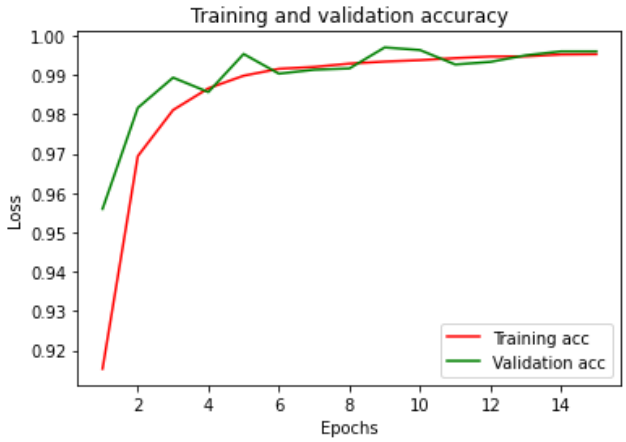

Figure 10. Training accuracy curve.

Model Comparison

To help researchers compare the effects of different filters on crack classification tasks, the table 3 was supplemented.

Table 3 Comparison study of applying different filters

Algorithm		Original	Gaussian Blur	Laplacian filter	Sobel X
Logistic Regression	Accuracy	0.8983	0.8989	0.7614	0.8272
	Precision	0.9363	0.9418	0.8765	0.8861
	Recall	0.8558	0.8513	0.6111	0.7526
	F1 score	0.8942	0.8943	0.7201	0.8139
Decision Tree Classifier	Accuracy	0.9594	0.9576	0.8063	0.8545
	Precision	0.9635	0.9600	0.8098	0.8636
	Recall	0.9554	0.9554	0.8031	0.8435
	F1 score	0.9594	0.9577	0.8064	0.8534
Random Forest Classifier	Accuracy	0.9897	0.9879	0.9117	0.9306
	Precision	0.9876	0.9842	0.8713	0.9051
	Recall	0.9919	0.9919	0.9670	0.9627
	F1 score	0.9897	0.9880	0.9166	0.9330
AdaBoost Classifier	Accuracy	0.9217	0.9084	0.8389	0.8690
	Precision	0.9881	0.9658	0.8770	0.9032
	Recall	0.8543	0.8477	0.7901	0.8279
	F1 score	0.9164	0.9029	0.8313	0.8639
Gradient Boosting Classifier	Accuracy	0.9559	0.9527	0.8808	0.9025
	Precision	0.9890	0.9884	0.9015	0.9328
	Recall	0.9225	0.9165	0.8563	0.8684
	F1 score	0.9546	0.9511	0.8783	0.8995
Extra Tree Classifier	Accuracy	0.9916	0.9918	0.9153	0.9339
	Precision	0.9922	0.9924	0.8839	0.9166
	Recall	0.9910	0.9914	0.9572	0.9554
	F1 score	0.9916	0.9919	0.9191	0.9356

As we can see from the table Gaussian filter with Extra Tree Classifier gives the best result. The gaussian filters tend to blur the noise, hence giving a more robust classification model. A finer tuning of these filters can be done by image processing experts and researchers to get better results.

Conclusion

In this research paper we have seen a comprehensive overview of the applications of classification models on classifying crack and non-crack images. We have seen the effects of applying filters to enhance the performance of the models. Though, the results were not significantly better than on the original images, image processing experts can fine tune the filters or apply more sophisticated filters to get more accurate results. The paper results show that in the group of Machine Learning models, Extra Tree Classifier is the best classifier, with an F1 score of 0.9916. For autonomous feature selection, we extend to focus on the result that a basic 5-layer CNN model can outperform all the Machine Learning models and attain a very good accuracy of 99.4%. But the deep learning models are prone to overfitting, therefore as future work researchers can work on building more robust and accurate deep learning models. The project has taught me many different concepts and approaches to solving classification tasks. Given, more time I would like to research more into using a better deep learning model to classify the images. Also, I would like to understand the effects of using dimensionality reduction techniques to reduces the feature space of the images. Further study would be required to understand the effects of filters on classification tasks, which will help fine tune the Machine Learning models that might potentially outperform the Deep Learning models. My advice to future students of DS 5220 would be to explore the applications of deep learning models, and transfer learning methods on this dataset, and to explore the effects of Gabor filter on this dataset.

References

[1] $350M Needed to Fix 2,000 Homes with Crumbling Foundations." Masslive, 8 Jan. 2020, https://www.masslive.com/news/2020/01/massachusetts-estimates-350m-needed-to-fix-2000-homes-with-crumbling-foundations.html

[2] Bad Concrete in the Foundations of 36,000 Houses. Where Else? (Boston: Refinancing, Buyers) - Massachusetts (MA) - City-Data Forum. https://www.city-data.com/forum/massachusetts/2982151-bad-concrete-foundations-36-000-houses.html. Accessed 28 Apr. 2021.

[3] "Wood vs. Concrete: The Best Choice for Builders and GCs | Giatec Scientific." Giatec Scientific Inc., 23 July 2020, https://www.giatecscientific.com/education/wood-vs-concrete-best-choice-builders-contractors/.

[4] Wu, Wei, Zheng Liu, and Yan He. "Classification of defects with ensemble methods in the automated visual inspection of sewer pipes." Pattern Analysis and Applications 18.2 (2015): 263-276.

[5] Guo, Zhenhua, Lei Zhang, and David Zhang. "A completed modeling of local binary pattern operator for texture classification." IEEE transactions on image processing 19.6 (2010): 1657-1663.

[6] Y. Hu and C. X. Zhao, "A Local Binary Pattern Based Methods for Pavement Crack Detection," Journal of Pattern Recognition Research 1, pp. 140-147, 2010.

[7] M. Salman, S. Mathavan, K. Kamal and M. Rahman, "Pavement crack detection using the Gabor filter," 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013, pp. 2039-2044, doi: 10.1109/ITSC.2013.6728529.

[8] Vijayarani, S., and M. Vinupriya. "Performance analysis of canny and sobel edge detection algorithms in image mining." International Journal of Innovative Research in Computer and Communication Engineering 1.8 (2013): 1760-1767.

[9] A. A. Prah and N. A. Okine, "Evaluating pavement cracks with bi dimensional empirical mode decomposition," EURASIP J. Adv. Signal Process, vol. 1, pp. 861701-1–861701-7, 2008.

[10] Adhikari, R.S., Moselhi, O., Bagchi, A. Image based retrieval of Concrete Crack Properties, Automation in Construction, 39(1), 180-194, 2014.

[11] Zhang, W.; Zhang, Z.; Qi, D.; Liu, Y. Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring. Sensors 2014, 14, 19307-19328. https://doi.org/10.3390/s141019307

[12] Hsieh, Yung-An, and Yichang James Tsai. "Machine learning for crack detection: review and model performance comparison." Journal of Computing in Civil Engineering 34.5 (2020): 04020038.

[13] Zhang, L., F. Yang, Y. D. Zhang, and Y. J. Zhu. 2016. "Road crack detection using deep convolutional neural network." In Proc., IEEE Int. Conf. on Image Processing (ICIP), 3708–3712. New York: IEEE.

[14] Cha, Y.J., Choi, W. and Büyüköztürk O., Deep Learning Based Crack Damage Detection Using Convolutional Neural Networks, Computer-Aided Civil and Infrastructure Engineering, 32(5): 361– 378, 2017.

[15] Dorafshan S, Thomas RJ, Maguire M. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. Data Brief. 2018;21:1664-1668. Published 2018 Nov 6. doi:10.1016/j.dib.2018.11.015.

[16] Le, Tien-Thinh, Van-Hai Nguyen, and Minh Vuong Le. "Development of Deep Learning Model for the Recognition of Cracks on Concrete Surfaces." Applied Computational Intelligence and Soft Computing 2021 (2021).

[17] Özgenel, Çağlar Fırat (2019), "Concrete Crack Images for Classification", Mendeley Data, V2, doi: 10.17632/5y9wdsg2zt.2

[18] Sekhar, Sidharth. "Math Behind Logistic Regression Algorithm." Medium, 30 Aug. 2019, https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c.

[19] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[20] Supervised Learning — Scikit-Learn 0.24.2 Documentation. https://scikit-learn.org/stable/supervised_learning.html. Accessed 29 Apr. 2021.