DEEP LEARNING ON LIVE STREAM UAV VIDEOS FOR VISUAL INSPECTION OF STRUCTURAL AUDIT

[Abstract: 3](#_Toc40212202)

[Introduction: 3](#_Toc40212203)

[Literature review: 3](#_Toc40212204)

[Gap analysis: 4](#_Toc40212205)

[Problem statement: 4](#_Toc40212206)

[Proposed Solution: 5](#_Toc40212207)

[Results and Analysis: 13](#_Toc40212208)

[Future Scope: 15](#_Toc40212209)

[Conclusion: 15](#_Toc40212210)

[Refrences: 15](#_Toc40212211)

# Abstract:

**Keywords:** neural network, crack, dampness, algae, fungi, paint peel-off, unmanned aerial vehicle, drone, structural audit, live stream

# Introduction:

Construction of different infrastructures is a necessity for human life, as it brings value to the land, fulfill human needs and also develop the area around it. As the infrastructures gets old, the risk of living/working in the building increases. We can see cracks, paint peel off and dampness on the walls of old infrastructures. The infrastructure should be examined after 30 years. The process is called structural audit. For structural audit the surveyor has to go to the location and do visual inspection of the infrastructure, different chemical tests, etc. While performing visual inspection for detection of cracks, dampness, paint peel off, the process totally depends on the surveyor’s experience and knowledge. Sometimes there may be human errors. With the help of machine learning and UAVs we can reduce the human errors caused during inspection and monitor almost the whole structure.

Many systems are developed for crack detection using image processing but it takes lot of time and resources. There are many systems developed using deep learning for detecting cracks on pavements and bridges. But only cracks are not the cause for damaging the infrastructure. For visual inspection a well-trained surveyor is required. But with the help of drone technology and deep learning a person with drone operating skills can have the job done. With UAVs, capturing images of higher grounds will be much easier and developing a neural network model for detection of the defects on captured images of walls will help us reduce human error rate and speed up the process. This process will be automated and reduce human work.

# Literature review:

Image processing is been used in detection of cracks. For enhancing the noisy image MESG algorithm is used. SOBEL filter is used for obtaining binary image [8]. CANNY edge detection is used for feature extraction [7]. HMRF-EM algorithm is used to increase the accuracy to label the data [5]. While detecting crack on a large objects motion blur occurs, to remove motion blur Wiener filter is used [4]. Binarization is used to find the width of the crack by changing user parameters for increasing the accuracy of detecting cracks [2].

For semantic segmentation combination of FCN (Full Convolutional Network) and Conv-LSTM (Convolutional Long Short Term Memory) is preferred over CNN to increase the accuracy [9]. For object detection, R-FCN (Region-based Full Convolutional Network) is used as it has pixel level classification design which can be easily applied to any general detection task [10]. Comparison between YOLO (You Only Look Once), R-CNN and SSD (Single Shot multibox Detector) on the basis of AUC, mAP and processing time results that R-CNN performs better in terms of AUC and mAP [11].

Deep Reinforcement learning (DRL), reinforcement learning (RL), Markov decision processes (MDP), partially observable Markov decision processes (POMDP) helps in navigation in complex environment [22]. YOLOv3 is more accurate than YOLOv2 but YOLOv3 takes longer training time [23]. Multilayer perceptron (MLP) is used for image classification and Otsu’s Thresholding Algorithm is used to find hypothesis between 2 classes [25]. Structural audit is an inspection of building whose age is above 30 years [26]. Principal Component Analysis (PCA) is used to calculate distribution of non-overlapping grids on an image [36]. Genetic Algorithm is used for optimization of CNN [37].

# Gap analysis:

[1-8, 14-15, 25, 27, 33] have stated digital image processing technique for crack detection. Instead of using image processing, using deep learning techniques can automatically learn complex features that need not be hardcoded and requires less computation once the model is trained. [1-2, 5-8, 10-12, 14, 16-17, 25-32, 34-35] papers’ methods require manual capturing of images or automatic capturing by fixing the camera in one location. Using Unmanned Aerial Vehicle (UAV) for capturing input data would be useful to capture dynamic data from human inaccessible areas. [1-8, 20-35] papers are working on image data. We can extend these work to video data. [1-8, 26-31, 33-35] papers are detecting only cracks in their work. But there are other structural defects in the infrastructure that can be detected visually including dampness of wall and peeling off of paint. [9-24] papers are working on other fields of image and video segmentation. We can extend their work in the field of structural audit.

# Problem statement:

The proposed system should be able to capture live video stream using UAV, which will be processed for detection of cracks, dampness and paint peel-off using neural network as a part of visual inspection while doing structural audit of infrastructures.

# Proposed Solution:

**1. Dataset**

This paper is using a custom dataset, as the dataset for all structural defects has not been created before. Our dataset consists of 4 classes i.e cracked walls, damp walls, walls with paint peel off, and walls with no defects. We captured the images from our mobile phones in the VJTI campus, Mumbai. The images are of varying dimension as 3 students were capturing images in different orientations from different mobile phones. These captured images contained various wall defects and other noises in the same photo. Hence we manually cropped the samples for each class from these images. Each cropped sample were resized to 256\*256 pixels. We are using RGB images as many important features of our classes can be easily captured if we use 3 channels of image instead of grayscale images.

**2. Preprocessing:**

Figure 1 shows the block diagram of the preprocessing steps that we used.

1. Each of the 4 classes contain unequal amount of cropped images. Pick random 625 samples for each class from the cropped samples. Each class has an equal amount of samples of 625 images which makes the dataset not skewed to any class.

2. Our 4 classes are not orientation dependent, i.e. even after flipping, rotating orthogonally or transposing on any diagonal, the class would not change. For e.g. crack if rotated by 90deg would still remain a crack. Thus we apply 7 non-distorting transformations (it does not add any synthetic pixels to the images) to the images. Those 7 non-distorting transformations are as follows:

i. Rotate image by 90 degrees

ii. Rotate image by 180 degrees

iii. Rotate image by 270 degrees

iv. Flip image horizontally

v. Flip image vertically

vi. Transpose image

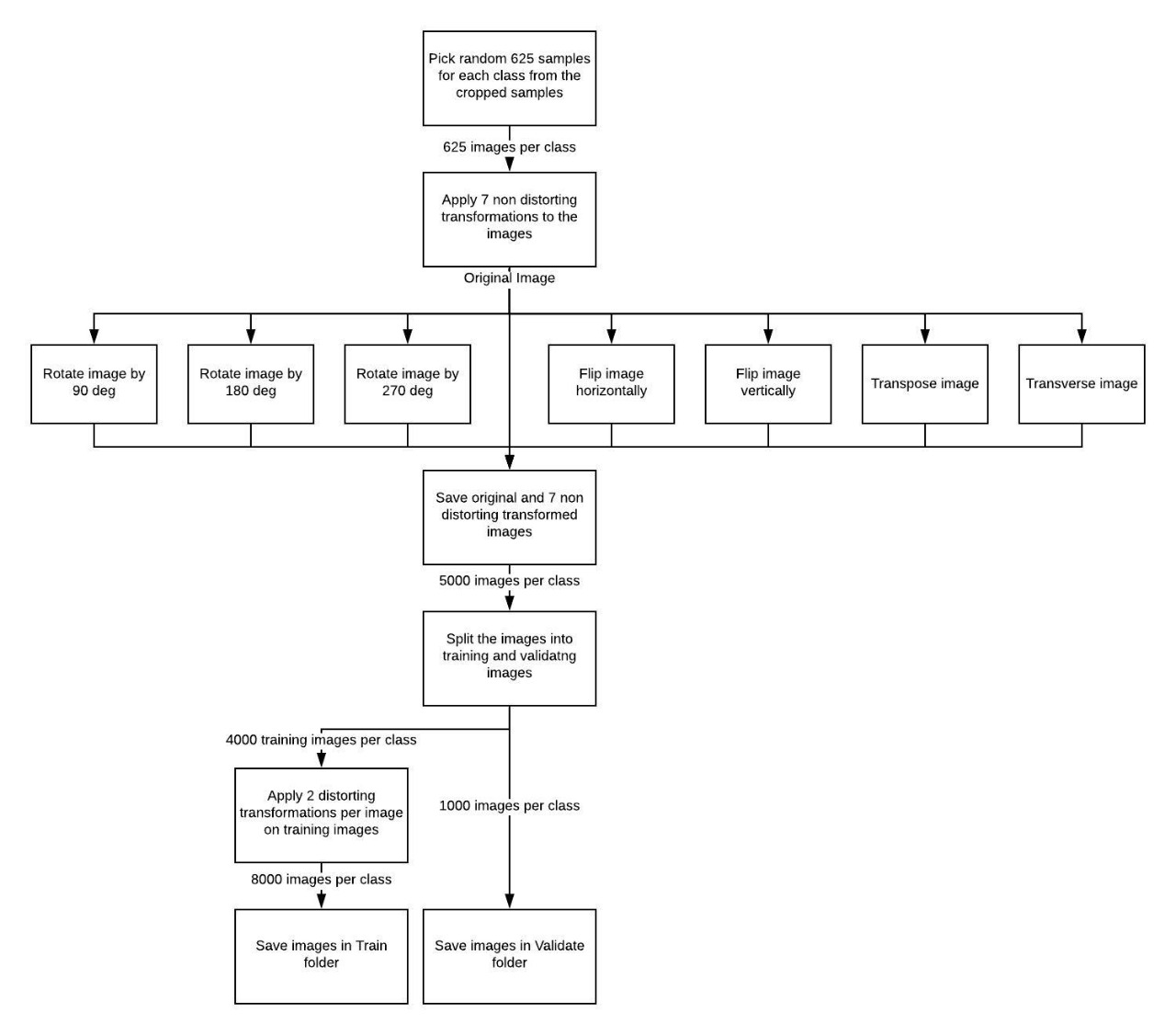
vii. Transverse image

3. Save those original images and their 7 non distorted transformation images. So we scale up the dataset size by a factor of 8 resulting to a total of 8\*625 = 5000 images per class.

4. Split that images into training as well as validating images. Here we are using an 80:20 ratio. So 4000 images are used for training purpose while 1000 images are used for validation.

5. In the process of training, we are applying distorting transformations so that our ML model does not overfit and generalizes on unseen output. We are generating 2 random distorting transformations per image. The total training images per class then results to 4000\*2 = 8000 images per class.

6. Save both training images and validate images in train folder and validate folders respectively.

****

**Fig 1:** Block Diagram for Preprocessing Steps

**3. Model**

As the structural audit domain is not explored much before, we experimented on a variety of model architectures including Transfer Learning, Neural Networks, and Convolutional Neural Network.

**3.1 Transfer Learning**

Training a model from scratch with a small dataset and getting very good accuracy is not always the case. So we tried to increase the size of our dataset by using available similar datasets on the Internet. This is one approach; other approach is to use someone else’s model which is already trained on a very large dataset. This can be achieved by transfer learning. Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. We tried using transfer learning for our problem, the pre-trained model we used is ResNet-50. This model is trained on CIFAR-10 dataset. The dataset is made of animals, vehicles, fruits, insects, man-made objects, etc. And our dataset consists of crack, paint peel-off, fungus and plain wall. The requirement for transfer learning to give good results is that our problem should be similar to the problem on which the model is trained. Our problem is not similar to the ResNet-50. We trained for 30 epochs by using the ‘conv5\_block3\_2\_relu’ layer output as our convolution layer and with one dense layer. When we trained the model we got a training accuracy of 98% but validation accuracy of 25%. Due to this behavior the model predicted every image as paint peel-off.

**3.2 Neural Network**

As the transfer learning was not giving good accuracy, we had to make our own model architecture from scratch. We started with Neural Networks. We tried different architectures by changing the number of hidden layers and number of nodes in those hidden layers. As we are doing the final predictions on video frames, complicated models with more hidden layers and nodes will increase the time required for forward propogation through model, hence we kept these numbers reasonably low. Even with Neural Network, we got a maximum of 60% training accuracy and 63% validation accuracy after training the model for 10 epochs. This hinted us that the model is underfitting and we need to add more complex features in the model.

**3.3 Convolutional Neural Network (CNN)**

After trying Neural Network, we came to know that we needed more complex features for the model to learn. Hence we tried CNN as there are many pattern features of our data that can be captured using CNN models.

Convolutional Neural Networks are a type of Deep Learning technique in which the model tries to capture different features, shapes, edges, etc. in the input image. A typical architecture of a CNN includes convolution layers at the beginning and fully connected dense layers at the end. The convolutional layers extract features from the input, and the dense layers combine these features to learn to recognize the classes in the dataset.

Like other Deep Learning techniques, CNN also involves Forward Propagation and Backward Propagation.

**3.3.1 Forward Propagation**

Convolutional layer uses a kernel to convolve the entire input image. For each convolving step, the kernel weights are elementwise multiplied to the part of the input image that the current convolution window is at and the entire products are summed up. This can be mathematically represented as follows

Here represents the input image and represents the kernel matrix. The resulting matrix is of the size where and are the number of rows and number of columns in the input image and is the size of the kernel.

Max pooling is a sample-based discretization process. This includes a pooling kernel which convolves upon the output of the convolutional layer and calculates the maximum of all the numbers present in the pooling window. Max pooling is also characterizd by the stride which indicates the and i.e. the number of pixels to move the pooling window in each of the x and y directions. Max pooling helps to reduce the computational cost as the size of the input of the next layer is reduced and also it prevents the model to be over fitted.

After passing the input from all convolutional and pooling layers, the input is flattened to a 1D array and then it is passed to the fully connected dense layers. The dense layers applies linear or non linear transformations based on the activation function used. The linear transformation can be represented as

Here represents the matrix of the weights associated with that layer. represents the input of that layer and is the bias matrix. Non linear transformation (softmax which is mostly used for multi labelled classifications) is applied to this linear transformation output which can be defined by the following formula

**3.3.2 Back Propagation**

During the training, the weights of the model are randomly initialized and thus the forward propagation gives erronous answers. Based on this errors, we find the weights in the model that are causing the highest error and update the weights according to the contribution that they are doing in the error. This process is called the backward propagation. Mathematically this same concept can be represented as the rate of change of error with respect to the weights which can be calculated by following formula

Here represents the error in the next layer (or output error if the current layer is the last layer), represents the output value that the current layer with activation produced while forward propagation, represents the output of the layer before applying activation function on it. is the weight of the current layer. For the last layer the error is just the squared difference between actual and predicted output.

This when derivated with respect to to output , we get

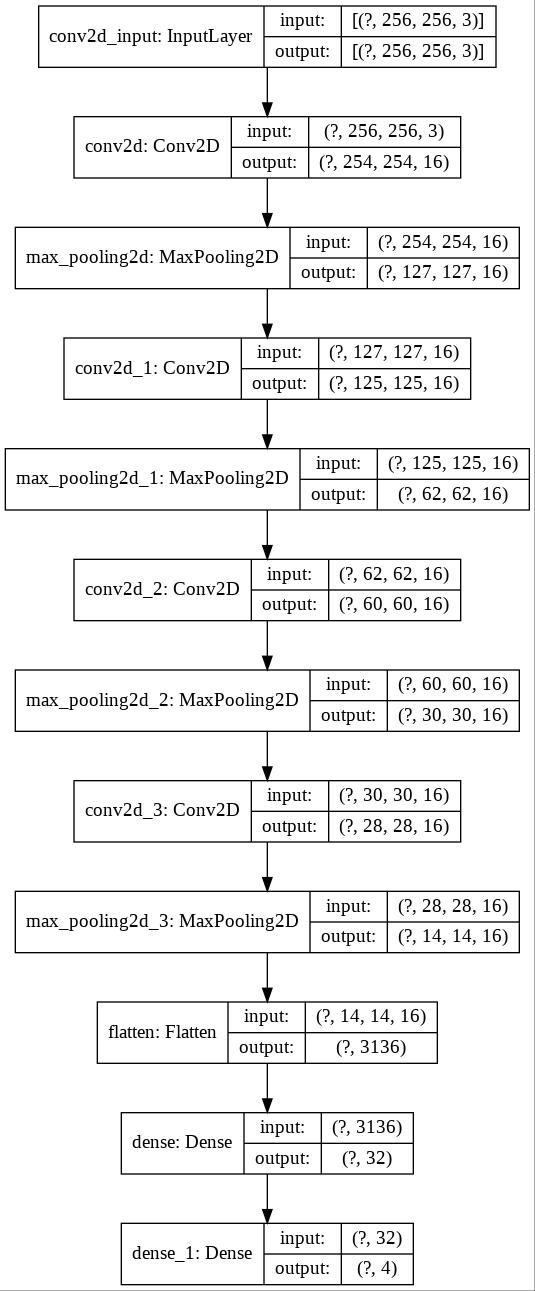
Now we need to calculate the derivative of the output with respect to the linear transformation output which is just the derivative of the softmax function. Next we need to calculate the derivative of with respect to the weights

Thus by calculating all the above partial derivatives, we can use chain rule and calculate the partial derivative of error with respect to the weights. This derivative term is used to update the weights of the fully connected layers in the gradient descent algorithm as follows

The backpropagation of the convolutional layers is similar but instead of weights, the convolutional layers consist of filter matrix.

**3.3.3 Architecture**

For the architecture, we again tried to tune hyperparameters like number of convolutional layers, number of filters in each convolutional layers and number of dense layers. By trial and error we got the architecture, illustrated in Fig 2, which consists of 4 convolutional layers with 16 filters each and a kernel size of 3\*3. We used Max Pooling of 2\*2 after each convolutional layers. We added one hidden Dense layer before the output layer containing 32 nodes. The input layer matches the dimensions of the dataset i.e. (256, 256, 3) where 3 is the number of channels in JPG images. The output layer consists of 4 nodes as we have 4 classes in our dataset.



**Fig 2:** Convolutional Neural Network Architecture

Now as we had the model ready, we used this model to predict classes on frames of videos captured using UAVs.

**4. Video Prediction:**

Figure 5 shows the block diagram showing the steps for ML prediction on video frames.

1. Video captured by the camera is given as an input to the program. As video is a collection of images (frames), the input video file is divided into frames and loaded in the memory.

2. Loop until all the frames are processed by our program.

2.1. For each frame, the dimensions of the image will be rounded down to values which are multiples of 256. This is done because, the dimensions of sliding window are 256×256 pixels and we want to apply predict function on the complete image.

2.2. Create a new result image. The dimensions of the new image are twice the shorter dimension and the same longer dimension of the original frame so that the input frame and resulting model prediction can be shown side by side. We paste the resized image at the top left corner of the result image and the remaining half image is for the model predictions generated by the program.

2.3. Loop until the sliding window mechanism has finished processing all rows.

2.3.1. Initially the window will begin at the left edge of the current row on the image.

2.3.2. Loop until all columns are processed.

2.3.2.1. Crop the image present in the sliding window.

2.3.2.2. Pass this cropped image to the model to predict what class the window belongs to.

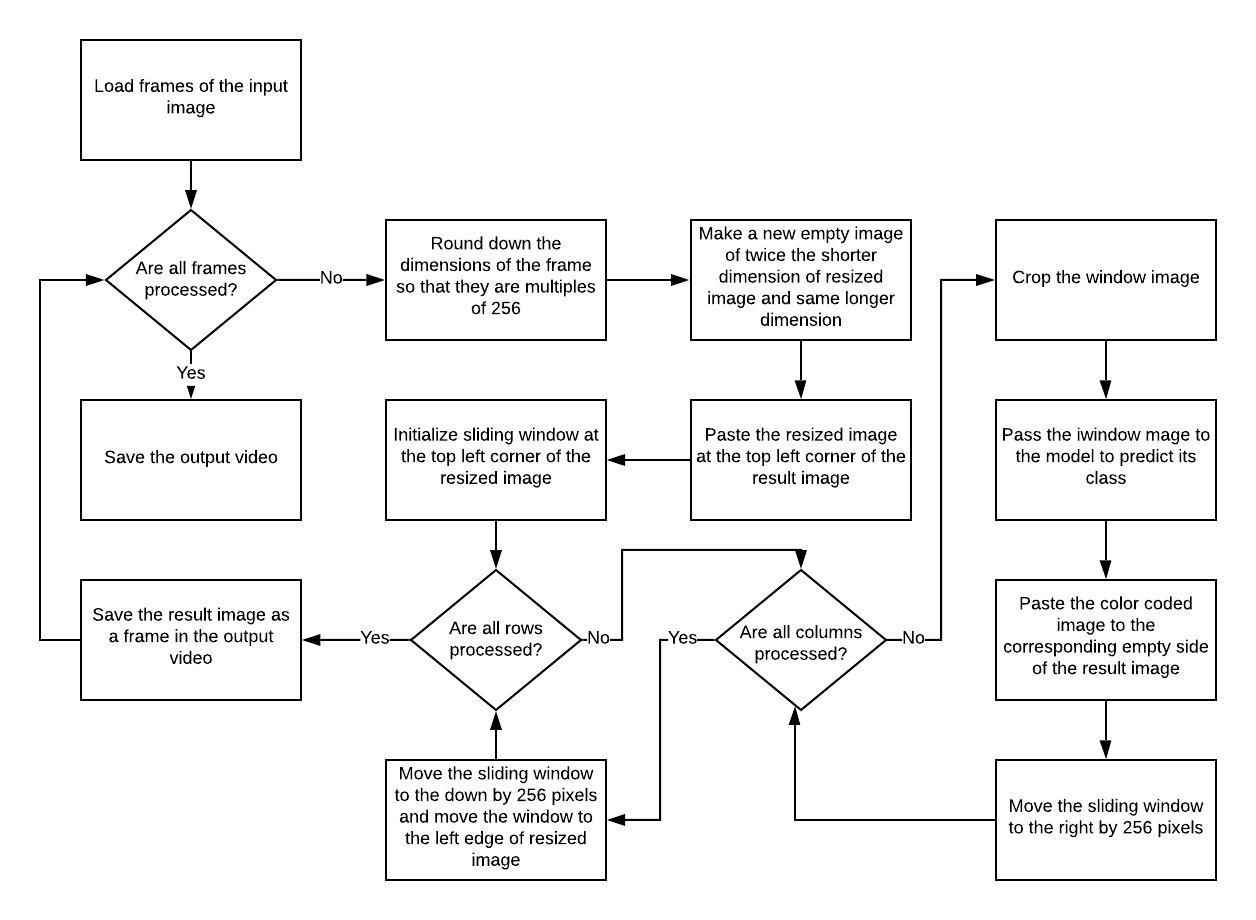
2.3.2.3. Paste the color coded prediction output to the corresponding part of the remaining half of the result image.

2.3.2.4. Move the sliding window to the right by 256 pixels.

2.3.3. Once all columns are processed move the sliding window down by 256 pixels.

2.4. Once all rows are processed, the frame is completely processed and the result image can be saved as a frame in the output video.

3. Once all frames are processed then we save the resulting frames by converting them to a single video at a given path.



**Fig 5:** Block diagram for classes prediction on video

# Results and Analysis:

**1. Training time:** Our modeltook 90 seconds per epoch and total time of training was 22.5 minutes for 15 epochs.

**2. Metrics:** After training the model for 15 epochs, we got the following results for accuracy and loss which we plotted in Fig 3 and 4.

|  |  |
| --- | --- |
| C:\Users\admin\Desktop\content\model\plots\accuracy_vs_epochs.jpg | C:\Users\admin\Desktop\content\model\plots\loss_vs_epochs.jpg |
| **Fig 3:** Accuracy vs Epochs | **Fig 4:** Loss vs Epochs |

As we can see from the plots, that the model is slightly overfitting at epoch 14 but the validation accuracy in this case is still 91% which was a great increase from earlier attempts. Further we have mentioned class wise metrics (precision, recall, and f1 score) in Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Crack** | 0.84 | 0.97 | 0.90 | 1000 |
| **Dampness** | 0.99 | 0.85 | 0.92 | 1000 |
| **No Defect** | 0.95 | 0.92 | 0.94 | 1000 |
| **Paint Peel Off** | 0.88 | 0.89 | 0.89 | 1000 |
|  |  |  |  |  |
| **Accuracy** |  |  | 0.91 | 4000 |
| **macro avg** | 0.92 | 0.91 | 0.91 | 4000 |
| **weighted avg** | 0.92 | 0.91 | 0.91 | 4000 |

**Table 1:** Class wise metrics report

**3. Confusion matrix:** We can see in the confusion matrix, shown in Table 2, that the values of true positive are very good. Out of 1000 test images per class approximately 900 images per class are identified correctly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Predicted→ Actual↓** | **Crack** | **Dampness** | **No Defect** | **Paint Peel Off** |
| **Crack** | 973 | 1 | 1 | 25 |
| **Dampness** | 83 | 852 | 26 | 39 |
| **No Defect** | 21 | 0 | 920 | 59 |
| **Pain Peel Off** | 82 | 7 | 18 | 893 |

**Table 2:** Confusion matrix

**4. Output format:** The output format is color coded and shown adjacent to the input video for better visualization. Fig 5 shows some frames of the ourput videos that we generated on test videos.

|  |  |
| --- | --- |
| F:\github\Drones_For_Structural_Audit\images\Screenshot (676).png | F:\github\Drones_For_Structural_Audit\images\Screenshot (677).png |
|  |  |
| F:\github\Drones_For_Structural_Audit\images\Screenshot (678).png | F:\github\Drones_For_Structural_Audit\images\Screenshot (679).png |

**Fig 5.** Output video frames

**5. Prediction time:** When we run the program on a NVIDEA GeForce 940MX Graphic Card powered laptop, prediction took 9 ms per (256\*256) image and 10 such predictions are made per frame for 720p videos leading to 90ms per frame. The program skips frames if the incoming rate of frame is higher but this can be easily solved if high end hardware is used instead.

**6. Comparison between transfer learning model and our model:** We cannot compare the output of transfer learning with our model because our problem is not similar to the problem of transfer learning model.

**7. Comparison between available models and our model:** Available models are binary classifiers and our model multi-label classifier. Available classifier has accuracy up to 98% and can only classify 2 classes while our model has accuracy of 91% but can classify 4 classes.

# 

# Future Scope:

1. Current dataset consists of four classes. Classes can be increased by identifying more defects and the variety of defect can be increased. Also the model can be used for visual inspection of other infrastructures by increasing the classes.

2.Current dataset has 5000images per class. By doing more survey we can increase the images count and by using a good camera we can increase the image quality.

# Conclusion:

We have generated dataset with 4 classes (crack, dampness, paint peel-off, plain wall) and each class has 5000 images. We tried testing our dataset using ResNet-50 model but validation accuracy was not good. Then we made our CNN model which gave us a validation accuracy of 91% which is less than the existing models but our model can classify more classes as compared to the existing models. Precision for crack, dampness, plain wall and paint peel-off, is 84%, 99%, 95% and 88%. Recall is 97%, 85%, 92%, 89%. F1-score is 90%, 92%, 94%, 89%.

# References:

[1] S. Kang, C. Chun, S. Shim, S. Ryu and J. Baek, "Real Time Image Processing System for Detecting Infrastructure Damage: Crack," 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2019, pp. 1-3.

[2] J. Vora, M. Patel, S. Tanwar and S. Tyagi, "Image Processing Based Analysis of Cracks on Vertical Walls," 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, 2018, pp. 1-5.

[3] R. K. Meghana, S. Apoorva, Mohana and Y. Chitkara, "Inspection, Identification and Repair Monitoring of Cracked Concrete structure –An application of Image processing," 2018 3rd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2018, pp. 1151-1154.

[4] L. Peng and J. Liu, "Detection and analysis of large-scale WT blade surface cracks based on UAV-taken images," in IET Image Processing, vol. 12, no. 11, pp. 2059-2064, 11 2018.

[5] M. Lin, R. Zhou, Q. Yan and X. Xu, "Automatic Pavement Crack Detection Using HMRF-EM Algorithm," 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 2019, pp. 1-5.

[6] L. S. Calderón and J. Bairán, "Crack detection in concrete elements from RGB pictures using modified line detection Kernels," 2017 Intelligent Systems Conference (IntelliSys), London, 2017, pp. 799-805.

[7] W. Li, M. Zhang, Z. Shen, W. Hu and P. Li, "Track Crack Detection Method in Complex Environment," 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2018, pp. 356-359.

[8] X. Liu, Y. Ai and S. Scherer, "Robust image-based crack detection in concrete structure using multi-scale enhancement and visual features," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 2304-2308.

[9] Y. Wang, Y. Lyu, Y. Cao and M. Y. Yang, "Deep Learning for Semantic Segmentation of UAV Videos," *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, Yokohama, Japan, 2019, pp. 2459-2462.

[10] H. Shi, Z. Liu, Y. Fan, X. Wang and T. Huang, "Effective object detection from traffic camera videos," 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, 2017, pp. 1-5.

[11]P. V. Babayan, M. D. Ershov and D. Y. Erokhin, "Neural Network-Based Vehicle and Pedestrian Detection for Video Analysis System," 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1-5.

[12] B. Hou, J. Li, X. Zhang, S. Wang and L. Jiao, "Object Detection and Trcacking Based on Convolutional Neural Networks for High-Resolution Optical Remote Sensing Video," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019, pp. 5433-5436.

[13] K. Mouri, H. Lu, J. K. Tan and H. Kim, "Object Detection on Video Images Based on R-FCN and GrowCut Algorithm," 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT), Busan, 2018, pp. 1-4.

[14] R. C. Joshi, M. Joshi, A. G. Singh and S. Mathur, "Object Detection, Classification and Tracking Methods for Video Surveillance: A Review," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1-7.

[15] Q. Wei, S. Lao and L. Bai, "Panorama Stitching, Moving Object Detection and Tracking in UAV Videos," 2017 International Conference on Vision, Image and Signal Processing (ICVISP), Osaka, 2017, pp. 46-50.

[16] B. Tian, L. Li, Y. Qu and L. Yan, "Video Object Detection for Tractability with Deep Learning Method," 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), Shanghai, 2017, pp. 397-401.

[17] M. H. Gasmallah and F. Zulkernine, "Video Predictive Object Detector," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, 2018, pp. 365-371.

[18] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci and N. Antonopoulos, "Video Stream Analysis in Clouds: An Object Detection and Classification Framework for High Performance Video Analytics," in IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 1152-1167, 1 Oct.-Dec. 2019.

[19] C. Wu et al., "UAV Autonomous Target Search Based on Deep Reinforcement Learning in Complex Disaster Scene," in IEEE Access, vol. 7, pp. 117227-117245, 2019.

[20] R. Geraldes et al., "UAV-Based Situational Awareness System Using Deep Learning," in IEEE Access, vol. 7, pp. 122583-122594, 2019.

[21] N. Q. Truong, P. H. Nguyen, S. H. Nam and K. R. Park, "Deep Learning-Based Super-Resolution Reconstruction and Marker Detection for Drone Landing," in IEEE Access, vol. 7, pp. 61639-61655, 2019.

[22] C. Wang, J. Wang, X. Zhang and X. Zhang, "Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning," 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, 2017, pp. 858-862.

[23] S. A. Hassan, T. Rahim and S. Y. Shin, "Real-time UAV Detection based on Deep Learning Network," 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), 2019, pp. 630-632.

[24] A. Zeggada, F. Melgani and Y. Bazi, "A Deep Learning Approach to UAV Image Multilabeling," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 5, pp. 694-698, May 2017.

[25] S. Koshy, B. Radhakrishnan and L. P. Suresh, "Strength analysis of buildings using image processing and SHM principles," 2016 International Conference on Emerging Technological Trends (ICETT), Kollam, 2016, pp. 1-6.

[26] N. A. M. Yusof, M. K. Osman, M. H. M. Noor, A. Ibrahim, N. M. Tahir and N. M. Yusof, "Crack Detection and Classification in Asphalt Pavement Images using Deep Convolution Neural Network," 2018 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2018, pp. 227-232.

[27] S. Liang, X. Jianchun and Z. Xun, "An Extraction and Classification Algorithm for Concrete Cracks Based on Machine Vision," in IEEE Access, vol. 6, pp. 45051-45061, 2018.

[28] L. Guo, R. Li, X. Shen and B. Jiang, "Crack and Noncrack Damage Automatic Classification from Concrete Surface Images using Broad Network Architecture," 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 1966-1971.

[29] L. Zhang, G. Zhou, Y. Han, H. Lin and Y. Wu, "Application of Internet of Things Technology and Convolutional Neural Network Model in Bridge Crack Detection," in IEEE Access, vol. 6, pp. 39442-39451, 2018.

[30] M. David Jenkins, T. A. Carr, M. I. Iglesias, T. Buggy and G. Morison, "A Deep Convolutional Neural Network for Semantic Pixel-Wise Segmentation of Road and Pavement Surface Cracks," 2018 26th European Signal Processing Conference (EUSIPCO), Rome, 2018, pp. 2120-2124.

[31] R. Fan et al., "Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding," 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 474-479.

[32] F. Nagata, K. Tokuno, K. Watanabe and M. K. Habib, "Design Application of Deep Convolutional Neural Network for Vision-Based Defect Inspection," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 2018, pp. 1705-1710.

[33] Y. Chen, T. Mei, X. Wang and F. Li, "A bridge crack image detection and classification method based On climbing robot," 2016 35th Chinese Control Conference (CCC), Chengdu, 2016, pp. 4037-4042.

[34] X. Wang and Z. Hu, "Grid-based pavement crack analysis using deep learning," 2017 4th International Conference on Transportation Information and Safety (ICTIS), Banff, AB, 2017, pp. 917-924

[35] S. Gibb, H. M. La and S. Louis, "A Genetic Algorithm for Convolutional Network Structure Optimization for Concrete Crack Detection," 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, 2018, pp. 1-8.