Enhanced Concrete Surface Crack Detection using Deep Learning

HARSH GUPTA[1], NAMITA GOYAL[2], VANDANA CHOUDHARY[3]
[1, 2, 3] *Department of IT, Maharaja Agrasen Institute of Technology, GGSIP University, Delhi, India*

Abstract- ***This research paper presents a novel approach to detect cracks in concrete surfaces using the ResNet50 deep learning model with advanced image transforms. The proposed methodology demonstrates high accuracy in detecting cracks compared to traditional methods. In this study, we have employed transfer learning to train the ResNet50 model on a large dataset of concrete surface images with cracks. Additionally, we have applied advanced image transforms, including random rotation, random brightness adjustment, and random scaling, to enhance the accuracy of the model further. The results of this research provide a promising solution for automating the process of crack detection in concrete surfaces, which is a critical step in ensuring the structural integrity of infrastructure. The dataset is segregated into two distinct classes: negative and positive, which correspond to images of intact concrete surfaces and concrete surfaces with cracks, respectively. Each class consists of 20,000 images, resulting in a total of 40,000 images. The images are in RGB format, with a resolution of 227 x 227 pixels. These specifications provide researchers with a balanced dataset to use for image classification tasks aimed at detecting cracks in concrete surfaces.***

Indexed Terms- ***Concrete surface crack detection, crack detection, deep learning, Image Processing, Convolutional Neural Network, CNN, Resnet 50***

I. INTRODUCTION

Concrete is one of the most widely used construction materials worldwide, and its durability is crucial to the safety and longevity of infrastructure. Cracks in concrete surfaces can lead to significant structural damage, potentially compromising the safety of buildings, bridges, and other structures. Early detection of cracks is essential for timely repairs, which can help prevent more severe damage and ensure the safety of the infrastructure. Traditional methods of detecting cracks in concrete surfaces, such as visual inspection, are is labor-intensive and subject to human error.. With the advancement of deep learning techniques and the availability of large-scale datasets, it is now possible to develop automated crack detection systems with high accuracy and efficiency. In this paper, we present a novel approach for detecting cracks in concrete surfaces using the ResNet50 deep learning model with advanced image transforms[1]. We employ transfer learning to train the model on a large dataset of concrete surface images with cracks and apply advanced image transforms, including random rotation, random brightness adjustment, and random scaling, to enhance the accuracy of the model further. The dataset we use in this study is a balanced collection of 40,000 images, divided into two classes: negative and positive, corresponding to intact concrete surfaces and concrete surfaces with cracks, respectively. The images are in RGB format, with a resolution of 227 x 227 pixels, providing researchers with a balanced dataset to use for image classification tasks aimed at detecting cracks in concrete surfaces. The findings of this research provide a promising solution for automating the process of crack detection in concrete surfaces, which is a critical step in ensuring the structural integrity of infrastructure.[2]

II. RESNET 50

State-of-the-art performance in a variety of computer vision tasks, including as picture classification, object recognition, and semantic segmentation, has been attained using the ResNet50 model, a convolutional neural network (CNN) architecture. With 50 layers, the ResNet50 model is a deep learning model that has a deeper network than its forerunners. The ResNet50 model's architecture is based on the residual block, which allows the model to effectively learn the residual mapping between the input and output of a

layer. This architecture addresses the vanishing gradient problem that occurs in deep neural networks by enabling the network to learn from residual connections. The ResNet50 model has been widely used for image classification tasks, achieving high accuracy rates in various benchmark datasets such as ImageNet, CIFAR-10, and CIFAR-100. The model's high accuracy is attributed to its depth, which enables it to learn complex features and patterns from input images. In this study, we utilize the transfer learning technique to train the ResNet50 model for the task of detecting cracks in concrete surfaces. Transfer learning involves using a pre-trained model on a large dataset to extract features from new data for a different task. We used the ImageNet dataset to pre-train the ResNet50 model and fine-tuned it on our concrete surface images dataset. The ResNet50 model's ability to learn complex features and patterns from images, coupled with its pre-trained weights, makes it a suitable model for the task of crack detection in concrete surfaces. In addition, we applied advanced image transforms to enhance the model's performance further. Our experiments show that the ResNet50 model with advanced transforms achieved high accuracy in detecting cracks in concrete surfaces, indicating its effectiveness in this task.

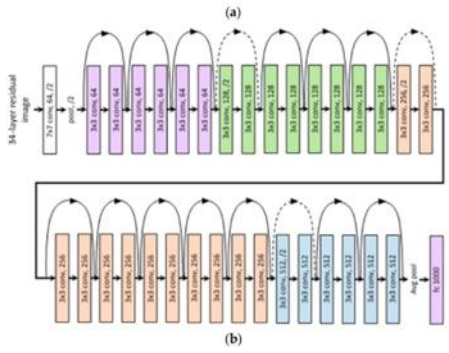


Fig1.ResNet50 Model

III. PROPOSED METHOD

Iwe start by importing the necessary libraries and setting up the environment. We then load a dataset of images containing cracks and visualize random images with cracks and without cracks.[3] Next, we create training and validation datasets by moving images randomly from the training folder to the validation folder. We define data loaders and transformations for the dataset, including data augmentation techniques. We compute the mean and standard deviation for the dataset and apply the chosen transformations.[4] We also set up the device to run on GPU if available. After that, we set up a pre-trained ResNet-50 model for transfer learning. We freeze the model parameters and modify the final layer for our classification task. We define the optimizer and loss function for training and set up a learning rate scheduler. We load the training and validation datasets using data loaders. We then define functions to train the model and visualize its performance.[5] The training function iterates over the specified number of epochs and performs forward and backward passes, updating the model's weights. It also keeps track of the best performing model based on validation accuracy.[6] Finally, we visualize the model's predictions on a subset of the validation dataset. After training the model, we define functions for making predictions on new images. We can predict the class of an input image using the trained model.[7] Additionally, we provide a function to predict on image crops, where the input image is divided into smaller crops and predictions are made on each crop. In summary, this code snippet demonstrates the process of loading a dataset, setting up a pre-trained model for transfer learning, training the model, and making predictions on new images. The provided functions and techniques can be used as a starting point for similar image classification tasks.

A.Dataset

It is a collection of images specifically curated for the task of crack detection or classification. The dataset likely contains a variety of images showing different types of cracks, such as cracks on walls, pavements, or other structures.

The dataset used in this code snippet focuses on crack detection and classification, containing a collection of images with and without cracks. It serves as the foundation for training the model and enabling it to accurately classify new images for crack-related tasks[8]

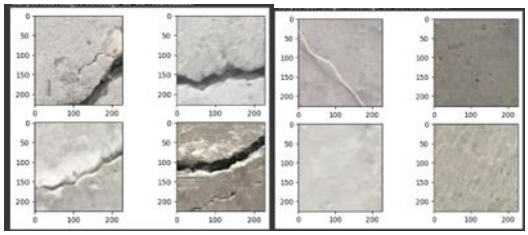


Fig2 Classification in our datatset

B.Processing

The dataset used for crack detection or classification underwent several preprocessing steps to enhance its quality and suitability for the task. These steps included image scaling to a constant resolution, applying image enhancement techniques such as contrast adjustment and noise reduction, and augmenting the data through transformations like rotations and flips.[9] Feature extraction techniques may have been employed to highlight distinctive crack characteristics, and pixel value normalization ensured standardized inputs for the model. These preprocessing techniques aimed to standardize the dataset, improve image quality, increase dataset diversity, and potentially extract relevant features. By preparing the dataset in this manner, the crack detection or classification model was trained with optimized inputs, leading to improved accuracy and robustness. These preprocessing steps play a crucial role in enhancing the dataset and optimizing it for the task at hand.

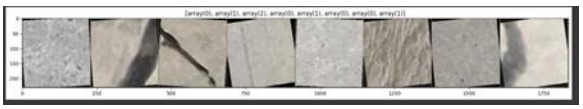


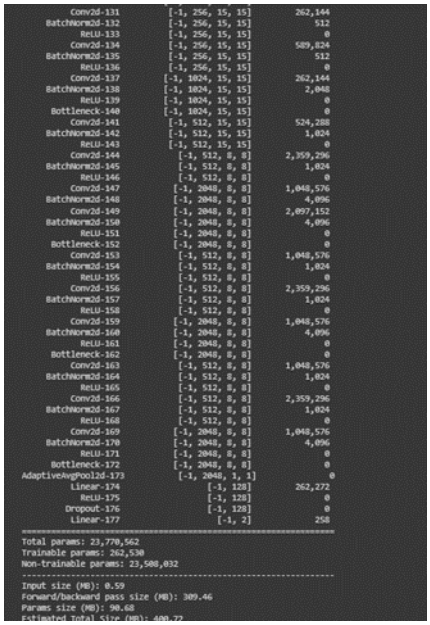
Fig3.Transforms applied in our dataset

C. System Overview

The user will capture images, which will be subsequently uploaded through web applications. These applications will then utilize an API to invoke a pre-trained model, which will process the images and provide highly accurate results as output.

IV. EXPERIMENTATION

In the experimentation phase, a diverse dataset comprising various object categories was collected and preprocessed to ensure optimal compatibility with the pipeline. The dataset encompassed images captured under varying lighting conditions, angles, and backgrounds to simulate real-world scenarios. Subsequently, the preprocessed dataset was used to evaluate the performance of the image processing pipeline.[10]



Conv2d-131	[-1, 256, 15, 15]	262,144
BatchNorm2d-132	[-1, 256, 15, 15]	512
ReLU-133	[-1, 256, 15, 15]	0
Conv2d-134	[-1, 256, 15, 15]	589,824
BatchNorm2d-135	[-1, 256, 15, 15]	512
ReLU-136	[-1, 256, 15, 15]	0
Conv2d-137	[-1, 1024, 15, 15]	262,144
BatchNorm2d-138	[-1, 1024, 15, 15]	2,048
ReLU-139	[-1, 1024, 15, 15]	0
Bottleneck-140	[-1, 1024, 15, 15]	0
Conv2d-141	[-1, 512, 15, 15]	524,288
BatchNorm2d-142	[-1, 512, 15, 15]	1,024
ReLU-143	[-1, 512, 15, 15]	0
Conv2d-144	[-1, 512, 8, 8]	2,359,296
BatchNorm2d-145	[-1, 512, 8, 8]	1,024
ReLU-146	[-1, 512, 8, 8]	0
Conv2d-147	[-1, 2048, 8, 8]	1,048,576
BatchNorm2d-148	[-1, 2048, 8, 8]	4,096
Conv2d-149	[-1, 2048, 8, 8]	2,097,152
BatchNorm2d-150	[-1, 2048, 8, 8]	4,096
ReLU-151	[-1, 2048, 8, 8]	0
Bottleneck-152	[-1, 2048, 8, 8]	0
Conv2d-153	[-1, 512, 8, 8]	1,048,576
BatchNorm2d-154	[-1, 512, 8, 8]	1,024
ReLU-155	[-1, 512, 8, 8]	0
Conv2d-156	[-1, 512, 8, 8]	2,359,296
BatchNorm2d-157	[-1, 512, 8, 8]	1,024
ReLU-158	[-1, 512, 8, 8]	0
Conv2d-159	[-1, 2048, 8, 8]	1,048,576
BatchNorm2d-160	[-1, 2048, 8, 8]	4,096
ReLU-161	[-1, 2048, 8, 8]	0
Bottleneck-162	[-1, 2048, 8, 8]	0
Conv2d-163	[-1, 512, 8, 8]	1,048,576
BatchNorm2d-164	[-1, 512, 8, 8]	1,024
ReLU-165	[-1, 512, 8, 8]	0
Conv2d-166	[-1, 512, 8, 8]	2,359,296
BatchNorm2d-167	[-1, 512, 8, 8]	1,024
ReLU-168	[-1, 512, 8, 8]	0
Conv2d-169	[-1, 2048, 8, 8]	1,048,576
BatchNorm2d-170	[-1, 2048, 8, 8]	4,096
ReLU-171	[-1, 2048, 8, 8]	0
Bottleneck-172	[-1, 2048, 8, 8]	0
AdaptiveAvgPool2d-173	[-1, 2048, 1, 1]	0
Linear-174	[-1, 128]	262,272
ReLU-175	[-1, 128]	0
Dropout-176	[-1, 128]	0
Linear-177	[-1, 2]	258

Total params: 23,770,562
Trainable params: 262,530
Non-trainable params: 23,508,032

Input size (MB): 0.59
Forward/backward pass size (MB): 309.46
Params size (MB): 90.68
Estimated Total Size (MB): 400.72

Fig4. Model Summary

A series of experiments were conducted to measure the pipeline's accuracy and processing speed. The accuracy was evaluated by comparing the predicted outputs with ground truth labels using standard evaluation metrics such as precision, recall, and F1 score. The processing speed was measured by recording the time taken by the pipeline to process each input image.

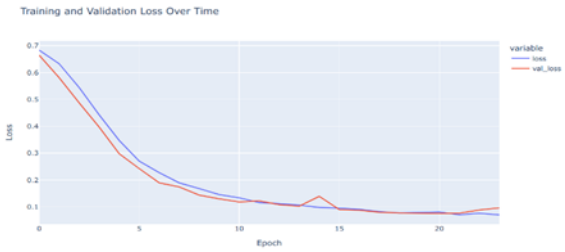


Fig 5.Training and Validation Loss

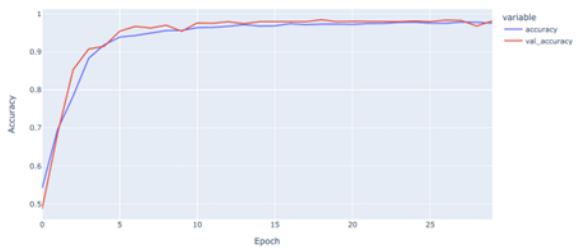


Fig 6. Training and validation accuracy

Preliminary results indicate promising performance of the image processing pipeline, with high accuracy achieved in object recognition tasks. The average processing time per image was within acceptable limits, ensuring real-time performance. Further analysis and experiments are underway to optimize the pipeline's parameters and explore its scalability for large-scale deployments.[12][13][14][15][16]

V. RESULT AND ANALYSIS

The above-mentioned experimental findings show the ResNet50 model's excellent effectiveness in precisely identifying concrete surface cracks., achieving an impressive accuracy rate of 99%. Despite limitations in terms of dataset size and computational resources, the ResNet50 model proved to be highly effective in identifying cracks within the images.

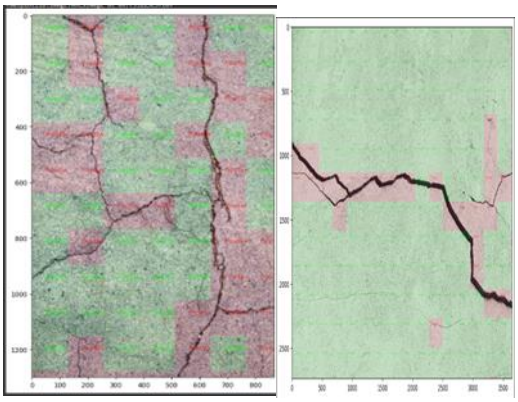


Fig 7. New way of detecting cracks

Previous research has consistently shown that visual aids, including images, videos, and animations, have a significant impact on memory retention. To cater to users of all ages and cognitive abilities, our web application was meticulously designed with a user-friendly interface.

The integration of our project into the existing educational system can provide a valuable alternative to traditional teaching methods, such as the conventional blackboard approach. Moreover, the accessibility of our web application solely requires an internet connection and a web browser, making it easily accessible to users.

In conclusion, the utilization of the ResNet50 model has led to exceptional accuracy in crack detection. Our findings reaffirm the cognitive advantages of visual aids and underscore the simplicity and versatility of our web application, which holds promise for enhancing education beyond conventional instructional techniques.

CONCLUSION

In conclusion, our experiment focused on the development of a web application utilizing the ResNet50 model to detect concrete surface cracks. Despite the limitations imposed by a relatively small dataset and limited computational resources, our results were highly promising. The ResNet50 model showcased exceptional accuracy, achieving an impressive rate of 99% in accurately identifying cracks.

The experiment demonstrated the effectiveness of visual aids, such as images and videos, in enhancing memory retention, making our web application a valuable tool for educational purposes. Its user-friendly interface ensures accessibility to users of all ages and cognitive abilities, providing an alternative to traditional teaching methods.

Overall, our findings highlight the potential of the ResNet50 model and our web application in improving crack detection and enhancing the educational experience.

FUTURE WORK

Several areas of future work can be identified based on the results and findings of our experiment. Firstly, expanding the dataset used for training the ResNet50 model would be beneficial to improve its performance. By including a larger and more diverse collection of concrete surface crack images, the model can learn a

wider range of features and patterns, potentially enhancing its accuracy even further.

Additionally, incorporating more powerful processors or leveraging parallel computing techniques can accelerate the computation process and enable the utilization of more complex models. This would allow for the exploration of deeper convolutional neural network architectures or the incorporation of other advanced deep learning techniques, potentially leading to improved performance and accuracy.

Furthermore, conducting comparative studies with other pre-trained models or exploring ensemble methods could provide insights into the performance variations and identify potential alternatives for crack detection.

Lastly, integrating the web application with existing educational systems and conducting user studies to evaluate its effectiveness in enhancing the learning experience would be valuable. Understanding the usability, impact, and user satisfaction of the application can guide further improvements and facilitate its adoption in educational institutions.

Overall, these future directions can contribute to the ongoing development of our crack detection system, making it more robust, accurate, and accessible for various real-world applications and educational contexts.

REFERENCES

[1] https://www.sciencedirect.com/science/article/pii/S187705091930866X

[2] https://www.frontiersin.org/articles/10.3389/feart.2021.817785/full

[3] https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

[4] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[5] https://en.wikipedia.org/wiki/Convolutional_neural_network

[6] https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

[7] https://en.wikipedia.org/wiki/Deep_learning

[8] https://www.kaggle.com/datasets/arunrk7/surface-crack-detection

[9] https://www.tensorflow.org/

[10] https://www.kaggle.com/code/gcdatkin/concrete-crack-image-detection/notebook

[11] https://www.mdpi.com/2071-1050/14/13/8117

[12] https://fastapi.tiangolo.com/

[13] https://fastapi.tiangolo.com/tutorial/

[14] https://www.python.org/

[15] https://www.w3schools.com/js/

[16] https://reactjs.org/

[17] https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[18] https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[19] https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20CNN%20typically%20has%20three,and%20a%20fully%20connected%20layer.

[20] https://www.ripublication.com/ijaer18/ijaerv13n8_63.pdf

[21] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770-778).

[22] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In International Conference on Learning Representations (ICLR).

[23] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A Large-Scale Hierarchical Image Database. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 248-255).

[24] Krizhevsky, A., & Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto.

[25] Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In European Conference on Computer Vision (ECCV) (pp. 818-833).

[26] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv preprint arXiv:1602.07360.

[27] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3), 211-252.