Autonomous Crack Detection using Deep Learning on Synthetic Thermogram Datasets

Chinmay Pimpalkhare
200100115
Department of Mechanical Engineering
Indian Institute of Technology Bombay
Name of Guide: Prof. Dnyanesh Pawaskar
BTP Final Report

I. Introduction

A. Need for Crack Detection

Failure in materials can occur due to the propagation of local cracks, which have the potential to cause widespread issues throughout the material. This susceptibility to crack propagation is often exacerbated by factors such as manufacturing errors or defects. A notable consequence of crack propagation is the increased vulnerability of the material to corrosion.

Corrosion, in the context of cracked materials, becomes a significant concern as cracks provide entry points for corrosive agents. This makes the material more prone to degradation, reducing its structural integrity over time. The global impact of failures attributed to corrosion is substantial, with an estimated cost of US$2.5 trillion, equivalent to 2.5% of the global GDP [1].

A case exemplifying the consequences of crack-related failures is the recall of 44,000 Toyota vehicles in 2020 due to potential engine cracks [2]. In this instance, a manufacturing error led to the development of cracks in the engines, necessitating a large-scale recall. This situation highlights the importance of effective detection methods, as timely identification of such issues could have facilitated preventive measures, minimizing the risk of widespread failures and associated economic impacts. Improved quality control and detection mechanisms are crucial elements in mitigating the risks associated with crack propagation in materials, particularly in critical components like automotive engines.

B. Current Methodologies

In the realm of Non-Destructive Testing (NDT), specifically using techniques like Thermography for crack detection, the current practice involves capturing thermographic images during the testing process. Following this, a human expert meticulously examines the thermography report to identify and analyze any indications of cracks or structural anomalies.

However, the existing methodology poses notable challenges. Manual inspection is a laborious and time-consuming task that is susceptible to human error. Moreover, it necessitates the expertise of skilled specialists, contributing to the overall complexity and cost of the process.

To address these issues, a promising solution is to develop an algorithm capable of automating the crack detection process, eliminating the need for expert operators. This algorithmic approach aims to streamline the inspection process, reduce the time required for analysis, and enhance the accuracy of crack detection.

Despite the potential benefits, the development of such algorithms faces its own set of challenges. One significant obstacle is the requirement for substantial amounts of data for effective training. Acquiring experimental data, however, is a resource-intensive endeavor in terms of time, energy, and financial investment.

An innovative idea to overcome this challenge involves the development of models trained on synthetic data. By creating a simulated environment that mirrors real-world conditions, these models can learn to detect cracks under various scenarios. The ultimate goal is to design a model capable of transitioning seamlessly from synthetic data to real-life data, thereby making crack detection more efficient, cost-effective, and widely applicable across diverse testing environments.

II. Proposed Method

A. Bird's Eye View

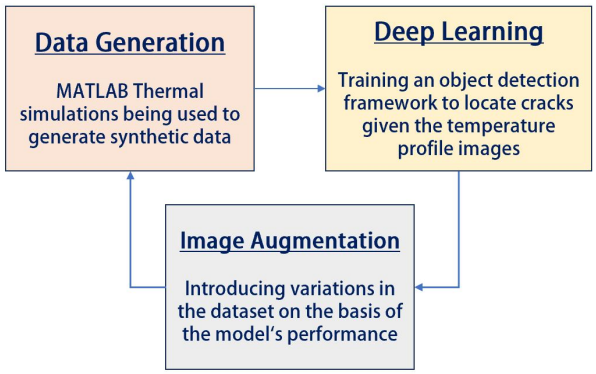

Fig. 1. Overview of our solution

*The report makes use of GPT-3.5 to convert data from presentation form to text in the report

Deep Learning is chosen for its distinct advantages in various applications. Its autonomous and accurate nature allows for the development of methods that can operate without heavy reliance on human intervention. Deep Learning models, once trained, showcase the ability to generalize well to new and unseen data, making them robust in handling diverse scenarios.

The preference for Synthetic Data in this context stems from the challenges associated with collecting Experimental Data. Gathering real-world data for training purposes can be a difficult and resource-intensive task. In contrast, Synthetic Data provides a more accessible and flexible alternative. It can be easily generated, offering researchers and practitioners increased opportunities for experimentation. This ease of data generation facilitates a more iterative and exploratory approach to model development, allowing for the refinement of algorithms in a controlled and adaptable environment.

B. Data Generation Pipeline

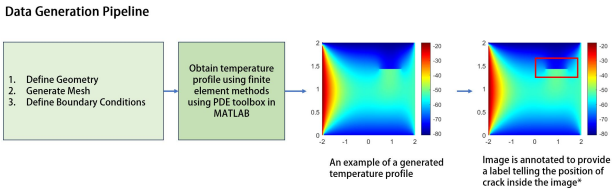

Fig. 2. The Data Generation Pipeline

The process begins by defining the geometry of the system, followed by the generation of a mesh that discretizes the domain. Boundary conditions are then specified to establish the constraints of the simulation. Subsequently, the temperature profile is obtained through the application of finite element methods, utilizing the PDE toolbox in MATLAB.

Annotations for crack detection are carried out using the CVAT.io annotation tool, enabling the precise marking of crack locations. The annotated information, specifying the location of cracks, is then stored in a text file, serving as input for the training of the algorithm. Additionally, boundary conditions provided during the annotation process are also stored.

C. Deep Learning Pipeline

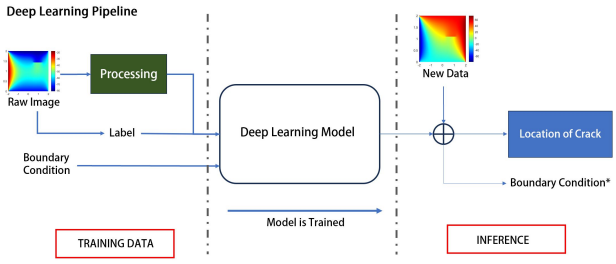

Fig. 3. The Deep Learning Pipeline

III. Survey of Literature

A. Relevant Papers

Various studies have addressed the critical task of crack detection using diverse methodologies and datasets. Yang, Wang et al. (2019) conducted experiments employing varying heat flux and thermal imagery, utilizing a Faster Region Proposal based Convolutional Neural Network for Object Detection. Similarly, Jaeger, Schmid et al. (2022) focused on turbine blade data, employing multiple deep learning models to classify infrared thermal images of turbine blades with cracks. In contrast, Mohan, Poobal (2018) provided a comprehensive review of crack detection methods, predominantly emphasizing vanilla image processing techniques without delving into deep learning.

Tian, Wang et al. (2021) explored a novel approach by implementing Generative Adversarial Networks (GANs) and Principal Component Analysis (PCA) for feature extraction in crack detection, emphasizing their feasibility for image augmentation with experimental data. Alexander, Hoskere et al. (2022) adopted the RFTNet, a Semantic Segmentation Network, specifically tailored for the fusion of RGB and thermal images in automated deep learning crack detection for civil architecture. Meanwhile, Chandra, AlMansoor et al. (2022) delved into the analysis of infrared thermal images of complex pavement defect conditions, incorporating seasonal effects using a vanilla Convolutional Neural Network.

Kovacs et al. (2020) explored deep learning approaches for thermographic imaging, employing synthetic data on virtual waves for Non-Destructive Testing. Fang et al. (2021) contributed to the field by using both synthetic and experimental data, employing a deep learning algorithm with pulsed thermography for automatic defects segmentation and identification, particularly focusing on simulations with CFRP materials.

B. Possible Improvements

Practical challenges in the domain of Non-Destructive Testing (NDT) primarily revolve around the difficulty in accessing datasets. This limitation arises due to various agreements, protocols, and constraints associated with NDT data. The predominant type of datasets utilized in this field comprises experimental data, supplemented by a limited amount of synthetic samples. However, a notable gap exists, as datasets with a substantial fraction of synthetic data are not widely available.

Algorithmic insights gleaned from current research reveal a prevalent use of Convolutional Neural Network (CNN)-based algorithms, with notable examples like Faster R-CNN employed for detection tasks. Additionally, Generative Adversarial Networks (GANs) have found application in the generation of synthetic data to augment existing datasets.

To address existing challenges and enhance the field, possible improvements involve tackling the scarcity of diverse synthetic data. Experimentation with new model architectures and exploration of alternative loss functions could also contribute

to the advancement of Non-Destructive Testing algorithms. These insights collectively highlight the ongoing efforts to overcome practical challenges, leverage existing datasets, and explore algorithmic innovations in the pursuit of more effective and robust crack detection methodologies.

IV. Data Generation

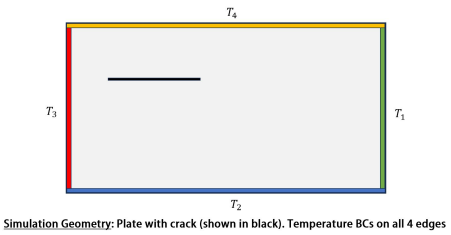

Fig. 4. Simulation Template

The data features for crack detection experiments involve maintaining constant plate dimensions and crack dimensions. The parameter $f = \frac{\text{crack length}}{\text{plate length}}$ is introduced as a key factor. Notably, at $f = 0.1$, the crack is easily visible to the human eye, while at $f = 0.001$, the crack is not immediately apparent, requiring a careful observation of the thermal profile for detection. Further, at $f = 0.0001$, the crack becomes nearly imperceptible to the human eye. Consequently, $f = 0.001$ was determined as the chosen value for further usage.

The variation in crack location along the x-axis is incorporated, and the dataset accounts for cracks at different angles through automated data augmentation. Edge temperatures are initialized randomly to introduce additional variability in the dataset. The experiments focus on steady-state temperature profiles, and the plate material is modeled as steel, utilizing corresponding material parameters.

For the purpose of these experiments, a crack is defined by the absence of material, and the systematic manipulation of these features allows for the exploration of diverse scenarios in crack detection while maintaining a consistent and controlled experimental framework.

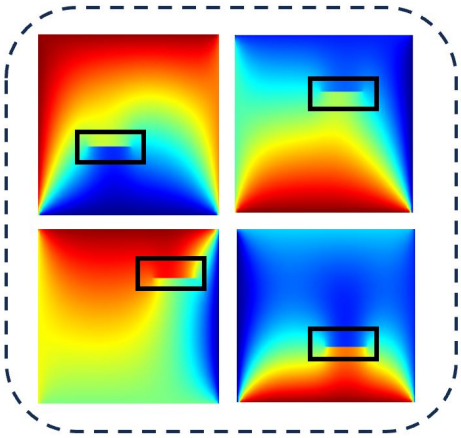

Fig. 5. Easy Examples

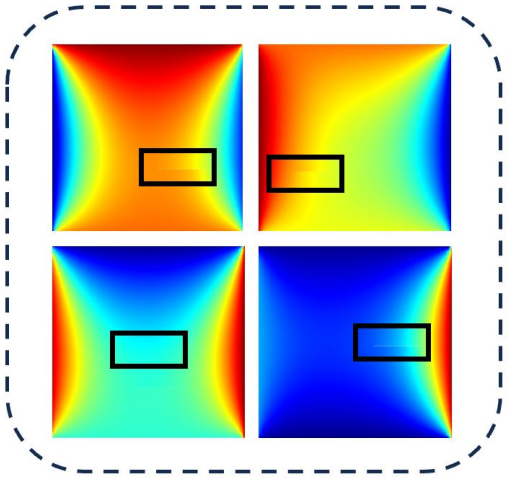

Fig. 6. Hard Examples

Easier examples are characterized by conspicuous variations in the thermal profile due to the presence of a crack. These instances are readily discernible, making it relatively straightforward for both automated systems and the human eye to identify the crack's location. On the other hand, harder examples pose a greater challenge, as the cracks are much more subtle and difficult to spot, even for the human eye. These instances demand heightened sensitivity and advanced detection techniques to accurately identify and locate the cracks within the thermal profile.

A. Benchmarking

In the process of validating the solutions, benchmarking was carried out by comparing our results with an analytical solution derived from the paper titled "Thermal Stresses In Plates with Circular Holes" by K.S. Rao et al., published in Nuclear Engineering and Design in 1971. Given that this paper focuses on a circular hole, we specifically compared our simulation methodology with the analytical solution for the same geometric configuration. Once the benchmarking process confirmed the validity of our method, simulations were extended to encompass the crack geometry, ensuring the reliability and accuracy of our approach in modeling thermal stresses.

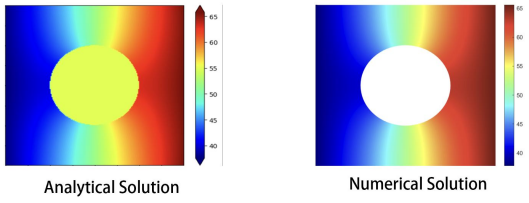

Fig. 7. Benchmarking

A thorough comparison for a circular hole geometry was conducted, revealing a high level of agreement between the analytical and numerical solutions. The visual examination

indicates a close match between the two, with magnitudes exhibiting a striking similarity. It's important to note that the green circle in the analytical solution serves as a mask, signifying regions where values were not computed.

V. Deep Learning

A. Feature Extraction

In the context of neural network detection, our objective is to identify cracks in a given image. The detection process involves analyzing low-level features initially, which are then integrated to identify higher-level features. The neural network focuses on detecting these features at a smaller scale and subsequently combines them using weighted operations. The goal is to calculate the probability of a crack's existence within a specified range of pixels, ultimately pinpointing the crack's location.

Crucially, temperature fluctuations result in rapid changes in pixel values, particularly in the proximity of a crack. This effect is more pronounced when the crack is perpendicular to the direction of maximum opposite edge temperature difference. Additionally, abrupt alterations in the direction of thermal contour lines occur as they become perpendicular to the crack.

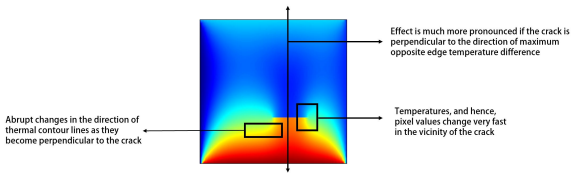

Fig. 8. Local Features from Neural Networks

B. Convolution Neural Networks

Convolution is a fundamental operation in neural networks that plays a crucial role in extracting features from input data. In the context of image processing, convolution involves systematically scanning the input image with a small filter or kernel. The filter contains learnable parameters that act as weights, and it is applied to local regions of the input through a sliding window. As the filter moves across the input, it performs element-wise multiplications and accumulates the results to produce a feature map. This process enables the network to capture spatial hierarchies of features, identifying patterns and structures at different scales. Convolutional layers are adept at recognizing low-level features, such as edges and textures, and progressively combining them to discern more complex and abstract features in deeper layers. This hierarchical feature extraction makes convolutional neural networks effective for tasks like image recognition and object detection. The use of shared weights in convolutional layers also contributes to parameter efficiency, allowing the network to learn and generalize patterns effectively. Overall, convolutional operations play a pivotal role in enhancing the capability of neural networks to understand and interpret complex visual information.

C. Edge Detection

Edge detection in neural networks involves the identification of boundaries or sharp transitions in input data, often employed in image processing tasks. One common method for edge detection is through convolutional operations using specific filters designed to highlight changes in intensity. Mathematically, this can be expressed using the convolution operation:

$$\text{Edge Map} = \text{Convolution}(I, \text{Filter})$$

Here, I represents the input image, and Filter is the edge detection filter or kernel. This filter is designed to respond strongly to variations in pixel values, emphasizing regions where intensity changes abruptly. The resulting edge map highlights the locations of edges in the input image. Common edge detection filters include the Sobel, Prewitt, and Roberts operators. Neural networks leverage such convolutional layers with learnable parameters to automatically adapt and extract meaningful edges, contributing to tasks like object recognition and segmentation. Edge detection is crucial for understanding the underlying structure of visual data, enabling neural networks to focus on relevant features and patterns.

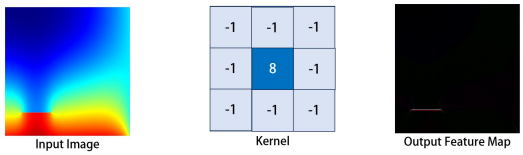

Fig. 9. Edge Detection Kernel

The edge detection kernel applied to the input image generates an output feature map that predominantly appears black, with the exception of the area corresponding to the location of the hole. The kernel is designed to emphasize changes in intensity, and in this context, it effectively highlights the boundaries of the hole.

D. YOLO Architecture

The YOLO (You Only Look Once) architecture stands out as a groundbreaking approach in the field of object detection within computer vision. YOLO revolutionizes the object detection paradigm by enabling real-time processing with remarkable accuracy. Unlike traditional region-based convolutional neural networks that rely on multiple passes over an image, YOLO processes the entire image in a single forward pass. This efficiency arises from dividing the input image into a grid and predicting bounding boxes and class probabilities directly within each grid cell. YOLO's architecture features multiple output layers responsible for detecting objects at different scales, ensuring the robust handling of objects of various sizes within the same image. Furthermore, YOLO seamlessly integrates with deep learning frameworks, facilitating ease of use and deployment. YOLO has undergone several versions, with each iteration refining its architecture for

improved performance and speed, making it a cornerstone in the development of real-time object detection systems.

YOLOv5, an evolution of the You Only Look Once (YOLO) series, represents a significant advancement in real-time object detection. Introduced as an open-source project, YOLOv5 builds upon the strengths of its predecessors while incorporating novel architectural enhancements. One notable feature is the shift to a more streamlined and scalable design, adopting a single, larger neural network.

VI. Implementation

A. Parameters

Entity	Details
Size of training dataset	80 images (280 x 280 px)*
Size of testing dataset	20 images (280 x 280 px)*
Optimizer	Stochastic Gradient Descent (SGD)
Metrics	Precision, Recall, mAP50, mAP50-95
Number of Epochs	250
Training Image Resize	512 px
Batch size	8
Training Time per Epoch	2.22 seconds (for dataset with size 80)
Inference Time	0.154 seconds (for 20 images)

*The reason why we have used such a small number of data points is that the model was able to perform well even with a small dataset. In further experiments, we shall use augmentation to increase the dataset size.

B. Evaluation Metrics

1. **Precision:** Precision is a metric that measures the accuracy of positive predictions made by a model, representing the ratio of true positive predictions to the total number of positive predictions. It is calculated as $\frac{TP}{TP+FP}$, where TP is the number of true positives and FP is the number of false positives.

2. **Recall:** Recall, also known as sensitivity or true positive rate, gauges the model's ability to capture all relevant instances of a class. It is computed as $\frac{TP}{TP+FN}$, where TP is true positives and FN is false negatives.

3. **IoU (Intersection over Union):** IoU quantifies the spatial overlap between the predicted and ground truth bounding boxes. Defined as $\frac{\text{Area of Intersection}}{\text{Area of Union}}$, IoU provides a measure of localization accuracy.

4. **mAP (mean Average Precision):** mAP is a comprehensive metric that evaluates the precision-recall curve across various confidence thresholds, yielding an average precision value. It is commonly used for object detection tasks.

5. **F1 Score:** F1 score is the harmonic mean of precision and recall, offering a balanced assessment of a model's performance. It is expressed as $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

6. **Confidence:** Confidence in object detection refers to the model's certainty in its predictions. It is often associated with the probability assigned to a predicted class, and higher confidence values indicate greater certainty.

C. Loss Functions

1. **Object Loss:** In the YOLO algorithm, object loss is a crucial component of the loss function, responsible for penalizing errors in predicting whether an object is present in a grid cell. It is defined as the binary cross-entropy loss between the predicted objectness score and the ground truth object label, encouraging the model to accurately discern the presence of objects.

2. **Box Loss:** YOLO incorporates a box loss term to optimize the accuracy of bounding box predictions. This loss penalizes the deviation between predicted and ground truth bounding box coordinates, ensuring precise localization of objects. The box loss term typically involves the mean squared error between predicted and true box parameters.

3. **Class Loss:** Class loss in YOLO addresses the accuracy of predicting the class of an object within a grid cell. It quantifies the dissimilarity between predicted class probabilities and true class labels using the categorical cross-entropy loss. This encourages the model to correctly identify and classify objects across different categories.

D. Metric Curves

1. **Precision vs Recall Curve:** In the YOLO algorithm, a precision vs recall curve illustrates the trade-off between precision and recall at varying confidence thresholds. This curve visually represents how adjusting the confidence threshold impacts the precision and recall values, offering insights into the model's performance across different confidence levels.

2. **F1 vs Confidence Curve:** YOLO often employs an F1 vs confidence curve to analyze the harmonic mean of precision and recall concerning confidence thresholds. This curve showcases how the F1 score, which balances precision and recall, varies as the confidence threshold changes. It helps identify an optimal confidence level that maximizes both precision and recall.

E. Metrics Obtained

Metric	Value	Interpretation
Precision	0.996	Value of precision at the maxima of the F1-confidence curve
Recall	0.95	Value of recall at the maxima of the F1-confidence curve
mAP50	0.947	Mean Average Precision when confidence threshold is set to 0.5
mAP50-95	0.507*	mAP averaged across confidence thresholds from 0.5 to 0.95 with step size of 0.05

We are getting very good values of precision, recall and mAP50 while mAP50-95 value is not very optimal

F. Predictions

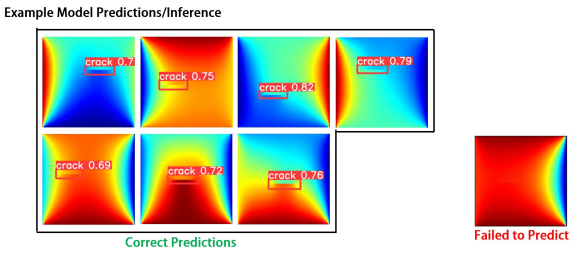

Fig. 10. Predictions

We can thus see that the model is able locate the crack correctly in most cases, but it fails during hard cases, when even a human eye is unable to locate the crack.

VII. Possible Improvements For Object Detection

Evaluation of Real-Life Performance: While the model demonstrates proficiency on synthetic data, assessing its efficacy in real-world scenarios is crucial for practical applications.

Data Diversity Enhancement: To bolster adaptability, incorporating more diverse data is key. Experimentation with various boundary conditions, such as images featuring multiple cracks or those with noise and low resolution, provides a more comprehensive understanding of the model's capabilities.

Optimizing Model Hyperparameters: Fine-tuning the model requires careful consideration of hyperparameters. Systematic exploration is essential to identify the optimal choices, including determining the best optimizer, learning rate, and batch size for improved performance.

Exploration of Model Architectures: Diversifying model architectures enhances the potential for success. Experimenting with alternative object detection models, like the Faster R-CNN, allows for a comprehensive analysis of which architecture aligns best with real-world detection challenges.

VIII. Segmentation

Segmentation is a pivotal technique in computer vision and image processing, aiming to partition an image into meaningful regions. Various types of segmentation methods exist, each tailored to specific applications. Firstly, there's semantic segmentation, which assigns a class label to each pixel in the image, delineating objects or regions based on their semantics. Then, instance segmentation takes a step further by not only classifying each pixel but also distinguishing between individual instances of objects. This method is particularly valuable in scenarios where objects overlap or occur multiple times within an image. However, instance segmentation is computationally intensive due to the need for precise delineation, making it slower than semantic segmentation. Additionally, it requires more extensive training data and often struggles with accurately segmenting objects with intricate shapes or occlusions. The trade-offs lie in the balance between accuracy and efficiency. While instance segmentation offers granular insights into object instances, it demands more computational resources and meticulous tuning. Thus, the choice between segmentation methods hinges on the specific requirements of the task at hand, weighing the precision required against the computational constraints.

A. Fine-Tuning

Fine-tuning involves training a model that was pre-trained on some large dataset on our small dataset. This involves freezing of the initial layers of the model, while updating only the later weights. The earlier layers of any deep learning model generally capture the more generic features such as edges, curves which are not really instance-dependent and are the property of any image, not considering what the theme is actually. However, the later layers start looking at more specific features, for example, facial expression in humans or the ratios of the face and eye of an animal etc. We want the weights of these layers to adapt according to the task that we are performing, and this is done by supplying a limited amount of training data to the pre-trained model.

B. Annotation Process

Annotation of data, especially images is generally a laborious task. In our case, for example, we need to use online tools and manually label each image by drawing a polygonal boundary around every instance of the class which we want the model to detect.

C. Detectron 2

Detectron 2, developed by Meta AI, is an advanced object detection framework known for its robustness and versatility in computer vision tasks. Building upon its predecessor, Detectron, this model boasts improvements in speed, accuracy, and flexibility. Leveraging state-of-the-art deep learning techniques, Detectron 2 provides a comprehensive toolkit for researchers and developers to tackle a wide range of challenges in object detection, instance segmentation, and related domains. Its modular architecture allows for easy customization and integration, making it a preferred choice for both academia and industry applications alike.

D. Mask RCNN

Mask R-CNN represents a paradigm shift in computer vision, seamlessly integrating object detection and instance segmentation. It extends the Faster R-CNN architecture by incorporating a parallel branch dedicated to generating pixel-level segmentation masks. Leveraging a Region Proposal Network (RPN) for object detection and a Mask branch for segmentation, Mask R-CNN achieves state-of-the-art performance in delineating object boundaries at the pixel level. This technical advancement enables precise segmentation of objects within an image, facilitating tasks such as object counting, scene understanding, and image manipulation. With its robustness and accuracy, Mask R-CNN has become a cornerstone in the field of semantic segmentation, empowering researchers and practitioners to tackle complex visual recognition challenges with unprecedented detail and precision.

IX. Improving Data Generation Pipeline

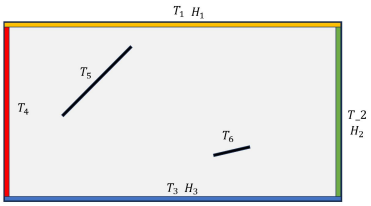

Fig. 11. Improved Simulation Template

The earlier pipeline was creating a lot of redundant images because there were only horizontal cracks. We wish to improve the data generation by applying a greater variety of boundary conditions. We also increase the diversity by randomly selecting the parameters such as the crack's center coordinates, the length of the crack, the width of the crack and the angle made by the crack with each of the plate edges.

Characteristic	Range
Plate Length	4 units
Plate Width	4 units
Crack Width	10^{-i} units $\in \{2, 3, 4\}$
Crack Length	$\in [0.3, 0.7]$ units
Crack Location	anywhere in the plate
Crack Inclination	$\in [0, 2\pi]$

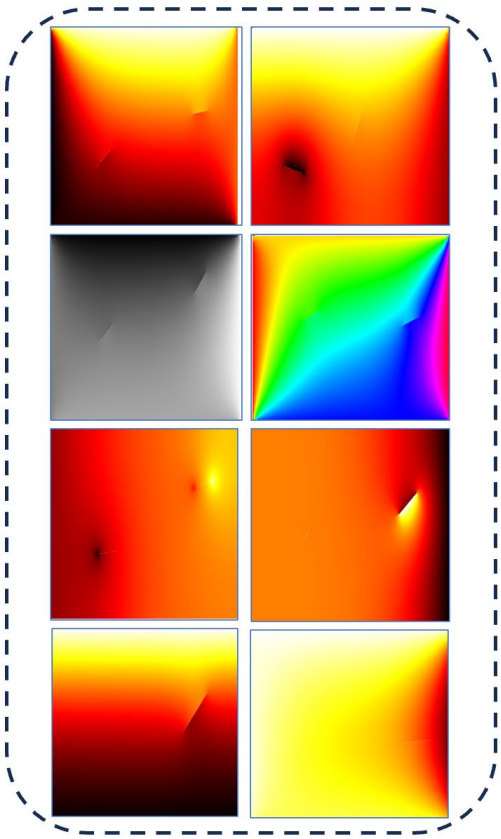

Fig. 12. Examples generated by new pipeline

X. Results

A. Using pre-trained models directly

We observe that if we directly use a pre-trained model without any sort of fine-tuning, the results observed are quite poor. Further, models trained on crack data, but from a different origin, such as pavement data, end up performing poorly when trying to detect the cracks from the image

B. Training with Fine-tuning

We trained a Detectron Model with the following dataset and characteristics

Dataset	Size
Training Dataset	105 images
Validation Dataset	30 images
Testing Dataset	15 images

XI. Inference Results

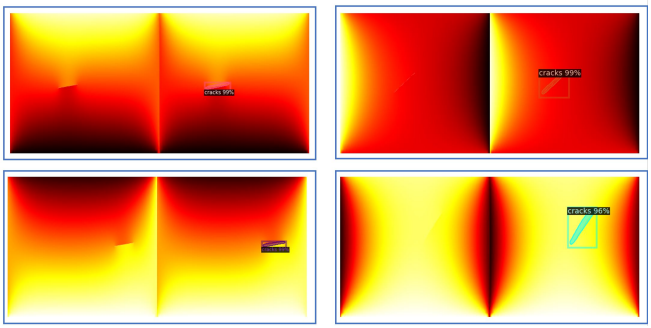

Fig. 13. Inference Results

The inference results are very promising. Here, we just grade the model on basis of whether it was able to detect the crack or not. We observe that in each of the 15/15 cases, the model was able to detect the crack correctly. This is excellent as the model is also succeeding in situations where a human has to squint a lot in order to be able to detect the image. In a couple of cases, detecting the crack was next to impossible for a human. Thus, this shows that finetuning is indeed a very good way to train a model on such a task.

XII. Future Scope

- From our experiments, we observed that Diversity $>$ Quantity!! We need to incorporate a variety of images; the size of the dataset is not much of an issue as the model is able to learn fast.
- We have not yet performed domain adaptation, and this is something which surely remains to be tested.
- Performing some kind of registration and reconstruction to generate three-dimensional data and projecting this data to a lower dimension is also something worth exploring.
- We can also employ software with stronger simulation capabilities to generate more complex data and situations. For example, software such as Ansys, LS-DYNA etc which have much higher modelling capabilities and are

state-of-the-art in their respective fields can be used for more precise generation.

XIII. Image Augmentation

We briefly discuss one method, which can make use of the inherent color properties of an image to help in creating an augmented dataset. The main color properties being

- **Brightness**: The intensity of light in an image, affecting how light or dark it appears overall.
- **Exposure**: The amount of light that reaches the camera sensor, determining how light or dark an image appears.
- **Saturation**: The intensity or purity of colors in an image, influencing how vivid or dull they appear.
- **Contrast**: The difference in brightness between the lightest and darkest parts of an image, affecting its overall clarity and definition.
- **Warmth**: The presence of red and yellow tones in an image, influencing its perceived temperature or mood.

We suggest a method to increase the dataset and also create examples which might be harder for the model to detect. This will improve robustness of the model

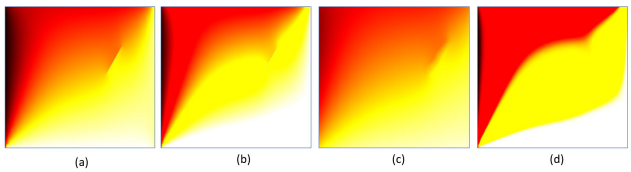

Fig. 14. From left to right ⟶ (a) Original image. (b) Image with high Brightness. (c) Image with parts near the crack blurred to decrease visibility. (d) Image with contrast, brightness and exposure adjusted such that it is harder to locate the crack

References

[1] Jaeger, B.E., Schmid, S., Grosse, C.U. et al. Infrared Thermal Imaging-Based Turbine Blade Crack Classification Using Deep Learning. J Nondestruct Eval 41, 74 (2022). https://doi.org/10.1007/s10921-022-00907-9

[2] Yang, J., Wang, W., Lin, G., Li, Q., Sun, Y., & Sun, Y. (2019). Infrared Thermal Imaging-Based Crack Detection Using Deep Learning. IEEE Access, 7, 182060-182077.

[3] Arun Mohan, Sumathi Poobal, Crack detection using image processing: A critical review and analysis, Alexandria Engineering Journal, Volume 57, Issue 2, 2018, Pages 787-798, ISSN 1110-0168, https://doi.org/10.1016/j.aej.2017.01.020.

[4] Tian, L., Wang, Z., Liu, W. et al. A New GAN-Based Approach to Data Augmentation and Image Segmentation for Crack Detection in Thermal Imaging Tests. Cogn Comput 13, 1263–1273 (2021). https://doi.org/10.1007/s12559-021-09922-w

[5] Alexander, Q.G., Hoskere, V., Narazaki, Y. et al. Fusion of thermal and RGB images for automated deep learning based crack detection in civil infrastructure. AI Civ. Eng. 1, 3 (2022). https://doi.org/10.1007/s43503-022-00002-y

[6] Chandra S, AlMansoor K, Chen C, Shi Y, Seo H. Deep Learning Based Infrared Thermal Image Analysis of Complex Pavement Defect Conditions Considering Seasonal Effect. Sensors. 2022; 22(23):9365. https://doi.org/10.3390/s22239365

[7] Péter Kovács, Bernhard Lehner, Gregor Thummerer, Günther Mayr, Peter Burgholzer, Mario Huemer; Deep learning approaches for thermographic imaging. J. Appl. Phys. 21 October 2020; 128 (15): 155103. https://doi.org/10.1063/5.0020404

[8] Fang Q, Ibarra-Castanedo C, Maldague X. Automatic Defects Segmentation and Identification by Deep Learning Algorithm with Pulsed Thermography: Synthetic and Experimental Data. Big Data and Cognitive Computing. 2021; 5(1):9. https://doi.org/10.3390/bdcc5010009

[9] Matlab Official Documentation: https://in.mathworks.com/help/matlab/

[10] YOLOv5 Official Implementation: https://github.com/ultralytics/yolov5