



AKADEMIN FÖR TEKNIK OCH MILJÖ
Avdelningen för datavetenskap och samhällsbyggnad

Improved U-Net architecture for Crack Detection in Sand Moulds

Husain Ahmed

Hozan Bajo

28 Maj 2023

Degree project, Basic level (Professional degree), 15 HE
Computer science
Study Programme in Computer Engineering
Supervisor: Mohammad Aslani
Examiner: Goran Milutinovic

Abstract

The detection of cracks in sand moulds has long been a challenge for both safety and maintenance purposes. Traditional image processing techniques have been employed to identify and quantify these defects but have often proven to be inefficient, labour-intensive, and time-consuming. To address this issue, we sought to develop a more effective approach using deep learning techniques, specifically semantic segmentation. We initially examined three different architectures—U-Net, SegNet, and DeepCrack—to evaluate their performance in crack detection. Through testing and comparison, U-Net emerged as the most suitable choice for our project. To further enhance the model's accuracy, we combined U-Net with VGG-19, VGG-16, and ResNet architectures. However, these combinations did not yield the expected improvements in performance. Consequently, we introduced a new layer to the U-Net architecture, which significantly increased its accuracy and F1 score, making it more efficient for crack detection. Throughout the project, we conducted extensive comparisons between models to better understand the effects of various techniques such as batch normalization and dropout. To evaluate and compare the performance of the different models, we employed the loss function, accuracy, Adam optimizer, and F1 score as evaluation metrics. Some tables and figures explain the differences between models by using image comparison and evaluation metrics comparison; to show which model is better than the other. The conducted evaluations revealed that the U-Net architecture, when enhanced with an extra layer, proved superior to other models, demonstrating the highest scores and accuracy. This architecture has shown itself to be the most effective model for crack detection, thereby laying the foundation for a more cost-efficient and trustworthy approach to detecting and monitoring structural deficiencies.

Keywords: U-Net Architecture, Semantic Segmentation, Convolutional Neural Networks, Crack Detection.

Sammanfattning

Att upptäcka sprickor i sandformar har länge varit en utmaning för både säkerhets- och underhållsändamål. Traditionella bildbehandlingstekniker har använts för att identifiera och kvantifiera dessa defekter men har ofta visat sig vara ineffektiva, arbetskrävande och tidskrävande. För att ta ut detta problem försökte vi utveckla ett mer effektivt tillvägagångssätt med hjälp av tekniker för djupinlärning, särskilt semantisk segmentering. Vi undersökte inledningsvis tre olika arkitekturer – U-Net, SegNet och DeepCrack – för att utvärdera deras prestanda vid sprickdetektering. Genom testning och jämförelse framstod U-Net som det mest lämpliga valet för vårt projekt. För att ytterligare förbättra modellens noggrannhet kombinerade vi U-Net med arkitekturerna VGG-19, VGG-16 och ResNet. Dessa kombinationer gav dock inte de förväntade förbättringarna i prestanda. Följaktligen introducerade vi ett nytt lager till U-Net-arkitekturen, vilket avsevärt ökade dess noggrannhet och F1-poäng, vilket gjorde den mer effektiv för sprickdetektering. Under hela projektet genomförde vi omfattande jämförelser mellan modeller för att bättre förstå effekterna av olika tekniker såsom batchnormalisering och drop out. För att utvärdera och jämföra de olika modellernas prestanda använde vi förlustfunktionen, noggrannheten, Adam-optimeraren och F1-Score som utvärderingsmått. Det finns tabeller och figurer som förklarar skillnaderna mellan modeller genom att använda bildjämförelse och jämförelse av utvärderingsmått; för att visa vilken modell som är bättre än den andra. De genomförda utvärderingarna avslöjade att U-Net-arkitekturen, när den förstärktes med ett extra lager, visade sig vara överlägsen andra modeller och visade de högsta poängen och noggrannheten. Denna speciella arkitektur har visat sig vara den mest effektiva modellen för sprickdetektering, och har därigenom lagt grunden för en mer kostnadseffektiv och pålitlig metod för att upptäcka och övervaka strukturella brister.

Nyckelord: U-Net Architecture, Semantics Segmentizing, Convolutional Neural Networks, Crack Detection.

Acknowledgements

We would like to express my heartfelt gratitude to my supervisor, Mohammad Aslani, for his unwavering support and guidance throughout this project. His dedication to teaching us the fundamentals and intricacies of machine learning and thesis writing has been instrumental in our success.

We would also like to extend our appreciation to Syntronic for providing the thesis project idea and generously supplying the DMK U33 Camera. We are especially grateful to Fredrik, the Hardware Development Department Manager, and Mats, the Software Development Department Manager, for their continuous support and assistance in shaping our project. We also wish to thank Linda for helping us navigate the process of starting our thesis project at Syntronic.

Special thanks go to the students from the previous year at the University of Gävle (HIG) who provided invaluable advice on managing our thesis project. Authors also appreciate the support and camaraderie of our classmates working on projects at Syntronic this year. The authors are also grateful to Syntronic for providing us with a conducive working environment, including office space, cameras, tools, computers, software, and technical assistance in every possible way.

Lastly, we would like to express my appreciation to all the teachers at HIG, particularly the head of the Computer Science Department, Åke Wallin, and our supportive examiner Goran Milutinovic. Your guidance and expertise have been invaluable in our journey towards completing this project. Thank you all for your contributions and support in making this project a reality.

List of Abbreviations

CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
DL	Deep Learning
ANN	Artificial Neural Network
SS	Semantic Segmentation
DIC	Digital Image Correlation
RMSD	Root Mean Square Deviation
CCDM	Cross-Correlation Distance Matrix
NDE	Non-Destructive Evaluation
ML	Machine Learning
FHT	Fast Haar Transform
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
VGG	Visual Geometry Group
BN	Batch Normalization
LoG	Laplace of Gaussian
SGD	Stochastic gradient descent

Table of Contents

1 Introduction	8
1.1 Background	8
1.2 Problem statement	8
1.3 Related work	9
1.4 Research objectives and questions	11
1.5 Research contributions	12
1.5.1 Extra-Layer New Improved U-Net	12
1.5.2 New designed dataset	14
2 Theoretical background.	17
2.1 Convolutional Neural Networks (CNN)	17
2.1.1 How CNN works?	17
2.1.2 Convolutional layers	17
2.1.3 Activation layer	18
2.1.4 Pooling layers	19
2.1.5 Fully connected layers	19
2.2 Semantic segmentation	19
2.3 Models' evaluation metrics	20
2.3.1 Accuracy	20
2.3.2 Loss Function	21
2.3.3 Precision	22
2.3.4 Recall	23
2.3.5 F1-Score	23
2.4 Optimizer (ADAM)	24
3 Method	26
3.1 Preparations and implementation of work	26
3.2 Preparation of the dataset	27
3.3 Loading data	28
3.4 Preprocessing the dataset	28
3.5 Using traditional image processing techniques (Canny edge detection)	28

3.6 Using Basic CNN models. _____	29
3.7 Using Semantic Segmentation Models _____	29
3.7.1 <i>U-Net</i> _____	29
3.7.2 <i>SegNet</i> _____	30
3.7.3 <i>DeepCrack</i> _____	31
3.8 Improving U-Net architecture. _____	32
3.8.1 <i>VGG-16 with U-NET</i> _____	32
3.8.2 <i>VGG-19 with U-NET</i> _____	32
3.8.3 <i>ResNet with U-NET</i> _____	33
3.9 Extra Layer technique _____	33
4 Result and discussion _____	34
4.1 Canny Edge detection method result _____	35
4.2 Semantic Segmentation comparison _____	35
4.3 Combined pre-trained models with U-Net comparison. _____	37
4.4 New Improved Extra Layer U-Net Vs Original U-Net _____	38
4.5 Authors dataset(Ex) Vs Original on Extra layer U-Net _____	39
4.6 F1 score plots comparison _____	40
5 Conclusion and future work _____	42
6 References _____	43

1 Introduction

1.1 Background

More than 60 years ago, in 1959, a computer began to be trained to play chess better than humans. It was the first step in machine learning since the technology has been developed and used until now in several different fields, such as image classification and crack detection. It was the beginning of the use of machine learning in engineering and several tasks in companies and industries. Today, it is possible to create systems that perform better than average people in several areas. Such systems may become economically feasible to use for real problems [1].

However, in crack detection terms and before machine learning was involved in the process to detect cracks in buildings, pavements and sand moulds, numerous techniques have been devised leveraging image processing techniques. Some of those techniques were, edge detection, Hough transform, image segmentation, identification and detection of feature points, and digital image correlation (DIC) method for feature identification and detection [2].

Later, a type of machine learning called Convolutional Neural Networks (CNN) was developed in 1989 to help computers identify different objects in images. Through object classification, computers are trained to recognize different objects in images. That gives computers the ability to find and isolate specific objects from the background of images. Two of those techniques have been developed and they are segmentation and object detection [3].

1.2 Problem statement

In the foundry industry, sand moulds are widely used to manufacture castings due to their low cost and easy availability. However, cracks have the potential to appear during the manufacturing process, and if left undetected, can lead to a product's failure, thus causing significant economic losses and hazards. Therefore, efficient, and accurate crack detection in sand moulds is crucial to ensure the quality and reliability of castings and saves money, and time. In the past, manual methods were used to detect and locate cracks in sand moulds, which required a lot of work and could lead to misjudgements and problems.

With the rapid development of technology, opportunities to improve and streamline this process have arisen. It has become increasingly interesting to explore automated methods to detect cracks, reduce the human workload and increase the accuracy of inspections. This development has taken place in several stages over time. Traditionally, crack detection in sand moulds has been performed manually by trained personnel, which is time-consuming, labour-intensive, and prone to errors. Moreover, obtaining consistent and accurate results is difficult, especially when dealing with complex patterns and shapes. Therefore, there is a growing demand for automated crack detection systems that can provide fast, reliable, and accurate results.

1.3 Related work

Hashimoto and Yamaguchi [4] developed a percolation-based image segmentation algorithm to segment images containing cracks. The algorithm uses an initial and maximum window size to define a percolation threshold, which is then updated to include pixels with lower brightness values in the percolated region until it meets a certain criterion. The study showed that this algorithm performs better than other methods such as thresholding and edge detection. However, there are still limitations in the algorithm that can be improved.

First, it is so sensitive to the threshold selection, that the performance of the percolation method is highly dependent on the choice of threshold parameter, which affects the quality of the extracted cracks. If the threshold is set too high, some cracks may be missed, while if it is set too low, there may be false positives, leading to the detection of non-crack features as cracks [4].

Second, it has limited effectiveness for complex crack patterns. The percolation method may not be effective in detecting cracks with complex patterns or those that are not continuous, such as those caused by impact or fatigue. In these cases, other methods such as machine learning or advanced image processing techniques, may be needed [4].

Image processing-based methods involve texture analysis, edge detection, and morphological image operators. Abdul-Qader[5] showed a comparative analysis of concrete bridge images, a set of 50 Gray-scale images with a resolution of 640x480 pixels was used. The images were obtained from bridge deck surfaces without background interference, with half being of intact components and the other half of deteriorating concrete with cracks. The images were processed using different algorithms through a custom MATLAB code developed for image reading, transformation, and crack isolation. The four algorithms that were compared in his study were: wavelet transform, Fourier transform, Sobel filter, and Canny filter, to analyse the images. Based on Abdul-Qader's findings, the wavelet transform was identified as a more dependable approach compared to the other techniques.

However, these methods do not consider the essential characteristics of cracks, such as their connectivity. While *Fast Haar Transform* (FHT) which was the best method for Abdul-Qader can provide a fast and efficient way to extract features from images, it may not be as effective in capturing detailed information about cracks, particularly if the cracks are small or have some complex patterns and shapes. Additionally, the accuracy of FHT can also be affected by noise in the image, leading to false detections or missed cracks in some cases [5].

Jahangir & Ashfahani [6] presented a wavelet-based damage identification approach for beam-like structures. To verify the efficiency and practicability of the proposed method, three damage scenarios with cracks of varying depths have been studied. The mode shapes and modal strains of the beam were analysed through wavelet transform and the results were compared. As occurs with traditional modal-based methods, the new approach needs a baseline as a reference which is wavelet coefficients of healthy structure. The quantification of damage was based on the traditional *root mean square deviation* (RMSD) and *correlation coefficient deviation metric* (CCDM) indices. The CCDM results obtained with the proposed method are better than the

ones obtained with RMSD. The wavelet analysis algorithm can be broken down into the following steps: The first step consists of obtaining mode shapes and modal strains of the undamaged and damaged beam. In the second step, the wavelet transforms of acquired mode shapes and modal strains are conducted. The third step is about Calculating RDMS and CCDM metrics of wavelet coefficients and damage localization based on these metrics is performed, while the fourth final step is to find out new indices or measurements that will lead finally to detecting the damage or the crack.

However, there are some shortcomings in this technique. Firstly, the method can be susceptible to noise in the input signal, which may lead to false positives by amplifying minor variations in the data that may be due to noise rather than actual cracks. This can make distinguishing between real cracks and noise difficult, especially in images or videos with high noise levels. Secondly, wavelet analysis requires careful tuning of its parameters, such as the choice of a wavelet function, the number of decomposition levels, and the thresholding method used to detect cracks. Finding the right set of parameters can be challenging and time-consuming, requiring expert knowledge and experience. Additionally, the optimal parameter values may vary depending on the specific application or type of image being analysed, making it difficult to generalize the method to different scenarios [6].

Ronny, Hung & Sheng [7] presented a methodology for crack detection in images captured by mobile robots. The approach involves identifying edge points using the *Laplacian of Gaussian* (LoG) algorithm, which serves as a 2D measure of the second spatial derivative of the image. However, the sensitivity of the Laplacian filter to noise is addressed through a pre-processing step of convolving the image with a Gaussian smoothing filter. By exploiting the associative property of convolution, a hybrid filter is created that can detect cracks in images with high efficacy. The problem in that project regarding the researchers was the accuracy of detecting cracks, and the need to focus on enhancing the crack detection algorithm under different environmental lighting conditions. It was a must for their team to boost the algorithm's resilience under varying conditions. Additionally, the authors intended to leverage the effectiveness of *Non-Destructive Evaluation* (NDE) sensors, such as Impact Echo and Ultrasonic Surface Waves, to gain insight into detecting vertical cracks.

Sun, Liu, & Fang [8] used the *Hough transform* method and combined different image processing techniques to reach better results with based-image processing methods to detect cracks. For image filtering and noise reduction, their team used the neighbourhood average method, the Gaussian filter method, and the median filter method. And in the crack extraction section, they have combined three algorithms: The *Soble operator edge detection* algorithm, the *Hough transform* method, *seed filling* algorithm. While in the last step of the process which is image skeleton extraction, they used an *improved Zhang thinning algorithm* and *Glitch removal* algorithm. The study proved that the accuracy of detecting cracks was improved by using the Hough method and combining many traditional image processing techniques.

Segmentation is used to find and isolate specific parts of an image. Object detection is used to find and describe objects in an image, such as finding cracks in a sand mould or concrete. These

technologies help computers process images and videos in real-time and automatically identify different objects in images and videos. This has been very helpful in many different areas. AlexNet was the first model in CNN to use new techniques to improve object classification, by using deep learning techniques, AlexNet was trained with large data sets to recognize patterns and structures in images and detect cracks. Deep learning has also been applied in image recognition where artificial neural networks (ANNs) have been used to identify objects and patterns in images and improve classification [3].

After the development of the AlexNet model, there have been several other models within CNN. Another study describes the most famous and important models in the field where each model has its unique characteristics and uses. The development in CNN does not stop at the models themselves but also includes new techniques to improve the performance such as transfer learning; this can be used to transfer knowledge from one model to another. In this way, previously trained models can be reused and adapted to new tasks, which can save time and resources in the development process. Choosing the right model for a specific task is an important part of developing a CNN solution and should be done carefully considering the task's requirements and the amount of data available [9].

The development within CNN continues to produce new and effective methods. It includes not only combining CNN's models such as transfer learning and using two models together, but also other techniques to improve performance and results in the task. An exciting application of the technology is in the measurement of crack length in materials after detection, where machine learning with CNN technology and image processing technology together can provide better results for cracks in images and measure their length and make decisions about how dangerous they are to use in the industrial process [10].

1.4 Research objectives and questions

There are two questions to be answered in this thesis:

1. How to detect sand mould cracks efficiently by using deep learning techniques and architectures?
2. Which is the most accurate semantic segmentation architecture in detecting cracks of sand moulds with a higher F1 score?

The primary objectives of this thesis project are twofold: first to develop a robust machine learning model capable of accurately detecting and localizing cracks in sand moulds, cement structures, and pavements, thereby minimizing costs associated with producing new components or delivering faulty products to customers; and second to seamlessly integrate the trained model into an industrial system, enabling real-time monitoring and providing timely alerts to the production line, ensuring the swift identification and remediation of defects, ultimately enhancing overall production efficiency and product quality.

1.5 Research contributions

In the realm of Semantic Segmentation architectures for crack detection, the authors present two significant advancements. Firstly, we proposed a refined U-Net architecture, an enhancement designed to optimize the model's performance. Secondly, the authors curated a novel, meticulously designed dataset, specifically tailored to improve model accuracy and elevate the F1 score in crack detection tasks. These contributions represent substantial strides in the ongoing quest for superior crack detection methodologies.

1.5.1 Extra-Layer New Improved U-Net

The process involves adding an extra layer to the existing U-Net Architecture, alongside the layers with 64, 128, 256, and 512 convolutional filters. This additional layer consists of 32 convolutional filters. Subsequently, the output of this newly introduced layer is concatenated with another layer in the decoder part. However, it is crucial to exercise caution when introducing new layers, as their effectiveness is not guaranteed. It is necessary to carefully consider the trade-offs and potential risks associated with increasing the model's complexity, such as elevated computational costs and the potential for overfitting.

To avoid any overfitting to come after adding the extra layer, the authors made a special dataset and combined it carefully to remove irrelevant or redundant features. Making the model train without overfitting requires also using data augmentation. For that purpose, the produced combined dataset itself is having augmented forms of part of the images within it. The early stopping technique was also used to reach the most accurate and highest F1 score model.

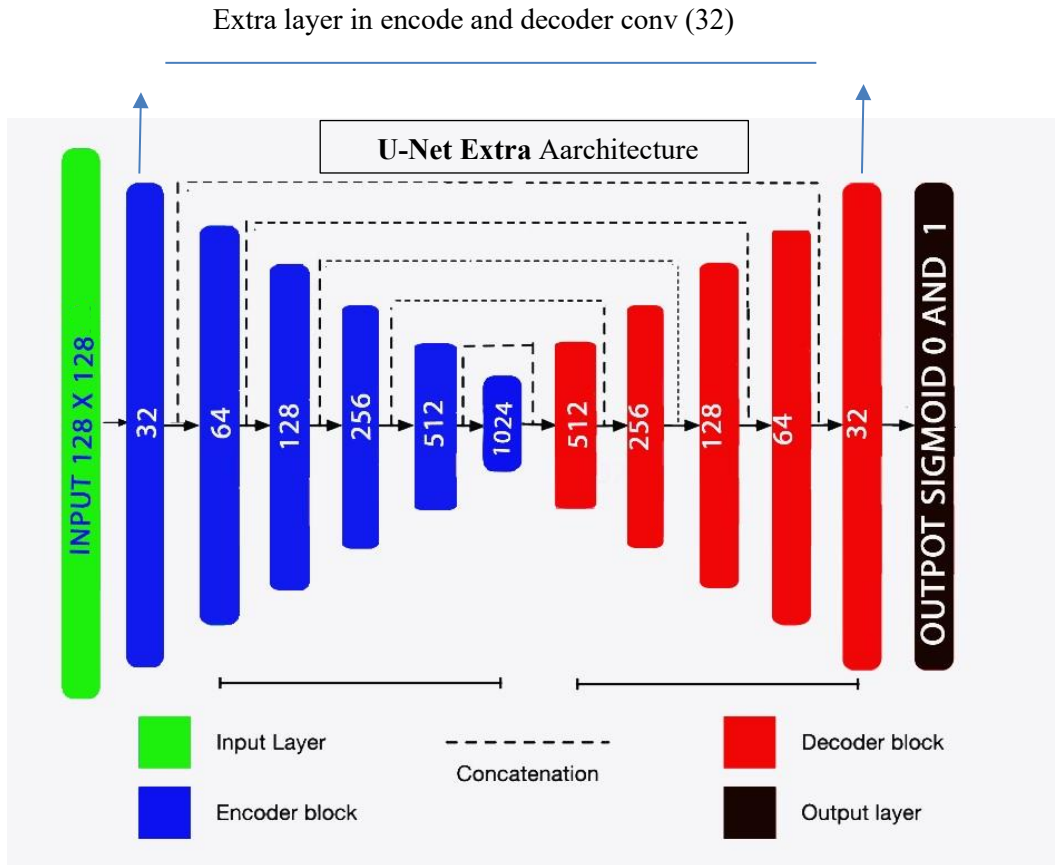


Figure1: U-NETextra layers with 32 filters were added for more features extraction

Adding an extra layer to the U-Net architecture can provide several advantages, as follows:

Improved feature extraction: An extra layer allows the model to learn and extract more complex features from the input images. This can lead to a more detailed representation of the image and a better understanding of the underlying patterns, which is crucial for accurate semantic segmentation.

Enhanced performance metrics: By capturing more intricate features and details, an extra layer can contribute to improvements in performance metrics, such as accuracy, and F1 score. This can result in more precise segmentation and better overall outcomes for various applications.

Increased model ability: While adding an extra layer can provide performance benefits, it also increases the model's ability. This may allow the model to better handle complex real-world scenarios and adapt to a wider range of input data variations. However, it is essential to balance the increased complexity with computational costs and potential overfitting risks.

Hierarchical Feature Learning: In deep learning architectures like U-Net, adding extra layers can enhance the hierarchical feature learning capability of the model. This allows the model

to learn features at different scales and levels of abstraction, which can be beneficial for capturing both local and global patterns in the input images.

Flexibility and Customization: Integrating an extra layer in the U-Net architecture allows for flexibility in adapting the model to specific problem domains. Researchers and practitioners can experiment with various layer configurations to find the optimal structure for a given task or dataset, potentially resulting in better performance tailored to their unique requirements.

1.5.2 New designed dataset

In this study, our second contribution lies in the creation and optimization of two distinct datasets, derived from the original `crack_segmentation_dataset` available on the Kaggle database. The original dataset, comprising 11,298 images with corresponding masks, was primarily modified in two significant ways. The first modification, referred to as the 'simplified version dataset', involved reducing the total image count to 7,226. This was achieved by eliminating four groups of images that were primarily characterized by deep, large cracks, along with their associated labels. In its place, we introduced a new selection of cement crack images, also sourced from Kaggle. The introduction of these new images was aimed at improving the detection of multi-thin cracks in images. The results from subsequent testing confirmed the effectiveness of this modification, as the simplified version dataset demonstrated superior performance in detecting multi-thin cracks, compared to the original dataset. However this was not working for all kind of cracks.

To address the shortcomings of the simplified version dataset, we developed a second dataset. This dataset was formulated based on the original `crack_segmentation_dataset` but with the addition of two more groups: a cement cracks group and the concrete crack dataset -O. The final count of images in this dataset amounted to 12,544. The primary aim of this dataset was to enhance the detection of the cracks. Testing results indicated that this dataset was indeed more proficient at detecting cracks in video captures.

In conclusion, our team has successfully created two novel datasets from the original `crack_segmentation_dataset`. The first dataset excels in the fast training and evaluating models in a fast way with GPU, while the second dataset specializes in detecting cracks in the most accurate way with the chosen model. These contributions highlight the value of dataset diversification and optimization, which can significantly enhance the accuracy and applicability of crack detection methods in varying contexts. (See Table 1, Table 2, and Table 3).

Type	Number of images
CFD	137
Crack Tree	236
Eugen Muller	63
Deep Crack	533
Crack-500	2519
Forest	136
No Crack	1645
Rissible for Florian	4131
Sylvie Chambon	187
Volker	1171
Gaps	540
Total	11298

Table 1:Kaggle (Original Dataset) "crack_segmentation_dataset"

Type	Number of Images
CFD	117
Crack tree	206
Eugen Muller	55
Forest	118
No Crack	1431
Possible For Florian	3842
Volker	1011
New added (Cement cracks)	446
Total	7226

Table 2: Simplified version dataset (Reader can notice that four groups were removed, and one new group have been added)

Type	Number of images
CFD	137
Crack Tree	236
Eugen Muller	63
Deep Crack	533
Crack-500	2519
Forest	136
No Crack	1645
Rissible for Florian	4131
Sylvie Chambon	187
Volker	1171
Gaps	540
Pavements Cracks	446
Concrete Crack dataset-O	800
Total	12544

Table 3: Extra segmentation version dataset (the reader can notice that two new groups of images were added to the original dataset)

2 Theoretical background.

In the realm of machine learning, Convolutional Neural Networks (CNNs) serve as a foundational pillar, particularly in image-based tasks. CNNs underpin the theory of semantic segmentation, a technique aiming to partition an image into semantically meaningful regions. Model performance is evaluated using specific metrics, ensuring the model's reliability and accuracy. Among various optimization algorithms, ADAM has emerged as a versatile and efficient choice, enhancing the learning process. The interplay of these components forms the basis for our project, vital for the advancement of accurate and robust image analysis. A comprehensive exploration of these theories and principles is forthcoming in the subsequent sections.

2.1 Convolutional Neural Networks (CNN)

CNN (Convolutional Neural Network) is a type of neural network (NN) used in machine learning (ML). This technique has improved the way data and images are handled and analysed in many fields, for example using such techniques and their trained models to analyse data using sensors, live video and images in other locations and make quick decisions like the human brain does. CNN helps to recognize patterns in images and is used in changing several fields such as medicine, diseases, and security. These techniques can detect patterns, improving people's quality of life, and providing better financial results for companies and industries [11].

2.1.1 How CNN works?

In the realm of scientific understanding, when working with a Convolutional Neural Network (CNN), the input is the image itself rather than each individual pixel. A CNN considers the image as a unified entity and processes it in the form of a matrix. This matrix comprises multiple channels, with each channel representing a specific colour component (e.g., red, green, blue) or other image properties. During the CNN's operation, the image matrix traverses through different layers, including convolutional, pooling, and fully connected layers. These layers operate on the image, executing operations involving filters, feature detection, and non-linear transformations. Their purpose is to extract and assimilate significant features and patterns from the image as a coherent entity [12].

2.1.2 Convolutional layers

Convolutional layers are the first layer in a CNN's model. It consists of several convolutional filters (kernels) that are used to extract important features from the images. By applying multiple filters to an image, the model can detect different features such as edges, shapes, and patterns. Starting from the upper left corner it is a 3x3 matrix and it can be 5x5 or 7x7. As the filters move across the image, it multiplies their values by the pixels of the image, and then add all the values

together to give a single output for each step until all the pixels of the image are done, then put them into a feature map that has multiple numbers of matrix your filtering [12].

When convolution operations happen in CNN's model, there are two techniques (padding and stride) that can be used to fill the edges, firstly there is zero padding and second is reflective padding, with zero padding the edges of the image are filled with zeros and with reflective padding copies the pixels at the edges to create a mirror image. The purpose of padding is to maintain the size of the image when using convolution operations and Stride to determine how much the window moves the filters, across the image pixels at each step [12].

2.1.3 Activation layer

2.1.3.1 ReLU

ReLU, short for Rectified Linear Unit, is an activation function commonly used in neural networks. It is a non-linear function that introduces non-linearity into the network, which allows the model to learn more complex relationships between the inputs and outputs. The ReLU function is defined as follows:

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

In other words, if the input x is positive, the function returns the input value x itself; if the input x is negative or zero, the function returns 0. The simplicity of the ReLU function makes it computationally efficient and easy to implement. It has been found to alleviate the vanishing gradient problem, which occurs when training deep neural networks, as it does not saturate for positive input values [13].

2.1.3.2 Sigmoid

The sigmoid function, also known as the logistic function, is an S-shaped curve that maps any real-valued input to a value between 0 and 1. It is commonly used as an activation function in neural networks, especially for binary classification tasks or in the final layer of the network, where the goal is to output probabilities.

The sigmoid function is defined as follows:

$$f(x) = 1 / (1 + \exp(-x))$$

Where x is the input value and $\exp(-x)$ is the exponential function with base “e” raised to the power of $-x$.

The sigmoid function has some attractive properties, such as being smooth and differentiable, with easily computable derivatives. However, it also has some drawbacks, like being susceptible to the vanishing gradient problem when used in deep networks, as the gradients become very small for large input values. In many modern deep learning architectures, other activation functions like ReLU (Rectified Linear Unit) have become more popular due to their better performance in certain scenarios [14].

2.1.4 Pooling layers

After the convolutional layers in CNN, the next layer in the model follows the pooling layer, that type of layer reduces the dimensions of the matrices in the feature maps to improve the performance of the model, there are several types of pooling, but two most common methods are average pooling and max pooling. With average pooling, the average value of the parts in the feature map is calculated, while max pooling selects the largest value from a part of the feature map. The pooling layer's main task is to reduce the demands on the computer's performance [13].

2.1.5 Fully connected layers

The Fully Connected layer is the last in the CNN's model and is used to convert the matrices created from previous layers into a one-dimensional vector. The purpose of this layer is to use a dimension vector to be able to classify input data into different categories. After the Fully Connected Layer usually comes a Softmax layer (or a Binary layer depending on the project) which allows the model to produce a probability over the different categories. This allows the CNN to produce a mode for a given input that describes which category the input belongs to with the highest probability. Softmax can be used to produce a probability distribution over multiple classes, Binary can be used to indicate which of the two classes the input belongs to with the highest probability. So, the choice between Softmax or Binary output depends on the project's specific requirements and number of classes [15].

2.2 Semantic segmentation

Semantic segmentation is a powerful computer vision technique used in many applications, including crack detection in various fields such as civil engineering and infrastructure maintenance. One type of crack detection that can be achieved using semantic segmentation is the identification of sand mould cracks in cast components. Sand mould cracks are a common defect in castings that can lead to component failure and are caused by improper handling, casting defects, and thermal stresses. Semantic segmentation using deep learning models such as U-Net, SegNet, and DeepCrack can help detect these cracks early on and enable timely repairs to be carried out.

U-Net is a deep learning model commonly used for semantic segmentation. It is a convolutional neural network that consists of a contracting path and an expanding path, which enables it to produce accurate segmentation results even with limited training data. SegNet, on the other hand, is a deep encoder-decoder architecture designed for image segmentation. It uses an encoder to extract features from the input image and a decoder to produce the output segmentation map [16].

DeepCrack is another deep learning model specifically designed for crack detection. It uses a U-Net architecture with a modified loss function to improve the accuracy of crack segmentation in images. DeepCrack has shown promising results in detecting cracks in various types of structures, including concrete and asphalt pavements [17].

By utilizing semantic segmentation techniques such as U-Net, SegNet, and DeepCrack, we can accurately identify and label regions of an image that correspond to sand moulds cracks or other types of structural damage. This early detection enables timely repairs to be carried out, which ultimately improves the safety and longevity of structures. The use of deep learning models in semantic segmentation has revolutionized the field of crack detection and is continuing to advance with new developments and applications.

2.3 Models' evaluation metrics

2.3.1 Accuracy

Accuracy is a commonly used performance metric in machine learning that measures the proportion of correctly classified data points out of all data points. In the context of semantic segmentation for crack detection, accuracy can be used to evaluate the ability of a model to correctly classify pixels in an image as either cracked or not cracked [18]. The equation for accuracy is:

Accuracy = $(TP + TN) / (TP + TN + FP + FN)$. Where:

- TP (True Positive) = number of pixels correctly classified as cracked
- TN (True Negative) = number of pixels correctly classified as not cracked
- FP (False Positive) = number of pixels incorrectly classified as cracked
- FN (False Negative) = number of pixels incorrectly classified as not cracked

The algorithm for computing accuracy in semantic segmentation for crack detection is as follows:

1. Input: Predicted mask, Ground truth mask
2. Calculate the number of True Positives (TP) by counting the number of pixels in the predicted mask that are classified as cracked and are also classified as cracked in the ground truth mask.
3. Calculate the number of True Negatives (TN) by counting the number of pixels in the predicted mask that are classified as not cracked and are also classified as not cracked in the ground truth mask.
4. Calculate the number of False Positives (FP) by counting the number of pixels in the predicted mask that are classified as cracked but are not cracked in the ground truth mask.
5. Calculate the number of False Negatives (FN) by counting the number of pixels in the predicted mask that are classified as not cracked but are cracked in the ground truth mask.
6. Compute the accuracy using the above equation.

Generally, Accuracy measures the proportion of correctly classified pixels in an image. The accuracy metric can be used to evaluate the performance of machine learning models and to identify areas where the model can be improved [18].

2.3.2 Loss Function

the loss function is a key component of the training process for machine learning models and is used to measure the difference between the predicted output and the actual output. In the context of semantic segmentation for crack detection, the loss function is used to measure the difference between the predicted mask and the ground truth mask. The most used loss function for semantic segmentation is the cross-entropy loss, which measures the difference between the predicted probability distribution and the actual probability distribution [18].

The cross-entropy loss function is defined as follows:

$$\text{Loss} = - \sum (y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y}))$$

Where:

y is the ground truth label for a given pixel, with a value of either 0 or 1 \hat{y}

is the predicted probability for the pixel, with a value between 0 and 1

The algorithm for computing the cross-entropy loss in semantic segmentation for crack detection is as follows:

Input: Predicted mask, Ground truth mask

Flatten the predicted mask and the ground truth mask into vectors of pixel values.

Calculate the cross-entropy loss for each pixel using the above equation.

Compute the mean cross-entropy loss across all pixels.

In addition to the cross-entropy loss, other loss functions can be used for semantic segmentation, such as the Dice loss and the Jackcard loss.

These loss functions are based on the similarity between the predicted mask and the ground truth mask and are commonly used in medical image segmentation.

The loss Function is used to measure the difference between the predicted output and the actual output. The cross-entropy loss function is commonly used in semantic segmentation for crack detection and measures the difference between the predicted probability distribution and the actual probability distribution. The choice of loss function depends on the specific problem being addressed and the goals of the analysis [18].

2.3.3 Precision

precision is a performance metric used to evaluate the ability of a machine learning model to correctly identify positive instances. In the context of semantic segmentation for crack detection, precision can be used to measure the ability of a model to correctly classify cracked pixels [19].

The equation for precision is: $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$

Where: TP (True Positive) = number of pixels correctly classified as cracked.

FP (False Positive) = number of pixels incorrectly classified as cracked.

The algorithm for computing precision in semantic segmentation for crack detection is as follows:

1. Input: Predicted mask, Ground truth mask
2. Calculate the number of True Positives (TP) by counting the number of pixels in the predicted mask that are classified as cracked and are also classified as cracked in the ground truth mask.
3. Calculate the number of False Positives (FP) by counting the number of pixels in the predicted mask that are classified as cracked but are not cracked in the ground truth mask.
4. Compute the precision using the above equation.

In addition to precision, other performance metrics are commonly used in semantic segmentation, such as accuracy and F1 score. These metrics are used to provide a more comprehensive evaluation of the model's performance. In short, precision measures the ability of the model to correctly classify cracked pixels. The precision metric can be used to evaluate the performance of machine learning models and to identify areas where the model can be improved [19].

2.3.4 Recall

The recall is a performance metric used to evaluate the ability of a machine learning model to correctly identify all positive instances. In the context of semantic segmentation for crack detection, recall can be used to measure the ability of a model to correctly identify all cracked pixels [19].

The equation for the recall is:

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Where:

TP (True Positive) = number of pixels correctly classified as cracked.

FN (False Negative) = number of pixels incorrectly classified as not cracked.

The algorithm for computing recall in semantic segmentation for crack detection is as follows:

1. Input: Predicted mask, Ground truth mask
2. Calculate the number of True Positives (TP) by counting the number of pixels in the predicted mask that are classified as cracked and are also classified as cracked in the ground truth mask.
Calculate the number of False Negatives (FN) by counting the number of pixels in the predicted mask that are classified as not cracked but are cracked in the ground truth mask.
3. Compute the recall using the above equation.

In addition to the recall, other performance metrics are commonly used in semantic segmentation, such as precision and F1 score. These metrics are used to provide a more comprehensive evaluation of the model's performance [19].

2.3.5 F1-Score

The F1 score is a performance metric used to evaluate the balance between precision and recall in a machine-learning model. In the context of semantic segmentation for crack detection, the F1 score can be used to measure the overall performance of a model in identifying cracked pixels [19].

The equation for the F1 score is:

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

The algorithm for computing the F1 score in semantic segmentation for crack detection is as follows:

1. Input: Predicted mask, Ground truth mask.
2. Calculate the number of True Positives (TP) by counting the number of pixels in the predicted mask that are classified as cracked and are also classified as cracked in the ground truth mask.
3. Calculate the number of False Positives (FP) by counting the number of pixels in the predicted mask that are classified as cracked but are not cracked in the ground truth mask.
4. Calculate the number of False Negatives (FN) by counting the number of pixels in the predicted mask that are classified as not cracked but are cracked in the ground truth mask.
5. Compute the precision and recall using the above equations.
6. Compute the F1 score using the above equation.

2.4 Optimizer (ADAM)

Optimizers are a key component of the training process for machine learning models and are used to adjust weights and biases in models during training. In the context of semantic segmentation for crack detection, optimizers can be used to improve the performance of the model in identifying cracked pixels.

One of the most used optimizers in deep learning is the Adam optimizer, which is a variant of stochastic gradient descent (SGD) that computes individual adaptive learning rates for different parameters. The Adam optimizer combines the advantages of two other optimizers, Adagrad and RMSprop, and is known for its fast convergence and robustness to noisy gradients.

The Adam optimizer computes the learning rate for each parameter using a combination of the first and second moments of the gradients and uses a moving average to estimate the momentum and variance of the gradient. The Adam optimizer updates the weights and biases of the model in the direction of the negative gradient, with a learning rate that is adaptive to each parameter [20].

The algorithm for the Adam optimizer in semantic segmentation for crack detection is as follows:

1. Input: Model weights and biases, training data, Learning rate, Batch size, Number of epochs
2. Initialize the model weights and biases with random values.
3. Divide the training data into batches of size Batch size.
4. For each epoch, iterate over each batch of data and compute the gradient of the loss function concerning the model parameters.
5. Compute the first and second moments of the gradient using the exponential moving average.
6. Compute the adaptive learning rate for each parameter using the first and second moments of the gradient.
7. Update the model parameters using the adaptive learning rate and the negative gradient.
8. Repeat steps 4-7 for the specified number of epochs.

The Adam optimizer is a popular optimizer used in deep learning and can be used to improve the performance of machine learning models in semantic segmentation for crack detection.

3 Method

In the field of computer vision, CNNs have become one of the most popular and effective methods for image processing tasks. They can learn to extract features from images in an automated way, which makes them suitable for tasks such as object recognition, segmentation, and classification. Semantic segmentation (SS) is one of the known used techniques in crack detection based on CNN. The method used in this study is based on CNN, SS, Deep Learning (DL), Python, cracks images and labels dataset, and a camera to test samples and trained models. More details will be explained in this section.

3.1 Preparations and implementation of work

In this thesis project, we collaborated with Syntronic, a company that provided essential resources and a conducive environment for the successful completion of the project. Syntronic supplied an industrial-grade camera, the DMK 33Ux290, specifically designed to capture high-quality images with outstanding sensitivity, resolution, and dynamic range, which were integral to the training of our machine-learning models. (See Figure 2).



Figure 2: Preparations of testing trained models in Syntronic room.

We used three laptops for model training, one of which was equipped with an NVIDIA GeForce GTX 1650 Ti graphics card. This powerful GPU greatly accelerated the training process, enabling the team to experiment with multiple models and configurations, ultimately leading to improved performance and faster convergence.

Syntronic also provided two variable-feature highlights for illumination, ensuring that the samples were consistently and evenly lit during image acquisition. The company further supplied a variety of coloured filters, enabling us to simulate real-world factory environments by adjusting the lighting conditions and capturing samples with differing colours. This approach facilitated the

development of a robust model capable of accurately detecting cracks across diverse manufacturing scenarios, ultimately leading to a more reliable and efficient production process.

3.2 Preparation of the dataset

CNNs have become a popular technique for image recognition and classification tasks. It requires large amounts of data to learn and generalize effectively. To train a CNN, a large dataset is needed. The dataset should be diverse enough to include different variations of the object of interest in different settings. Additionally, the dataset should be split into training, validation, and testing sets to evaluate the performance of the trained model.

In this project, Firstly, in the early process of the thesis, the authors worked on basic CNN models like VGG-16, VGG-19, ResNet, etc. The dataset that has been used was SDNET-2018 and it had almost 5600 images of two groups (Cracks, No Cracks). The SDNET-2018 dataset work as dividing the However, later in semantic segmentation, authors used websites like Kaggle provided "crack_segmentation_dataset", containing more than 11,298 images with their labels (masks). The dataset was split into three sets: 80% for training, 10% for validation, and 10% for testing. By splitting the data into these sets, the CNN's performance can be evaluated accurately, and overfitting can be avoided.

Later in the next stages of the project, the authors created two new datasets based on Kaggle's "crack_segmentation_dataset". See section [1.5.2](#). The "simplified segmentation dataset" dataset was specialised to increase the speed of the training but after using it was found that it is not detecting as much as the original dataset it was the best for using on GPU because of the lack of RAM so the authors use it in testing models. And the second is called "Extra segmentation version dataset". Was the best in detecting cracks of the sand moulds samples test provided in Syntronic labs. The following table shows general information about the four main datasets that the authors used in this project(See Table 4).

Dataset Name	Dataset Creator	Dataset Type	Amount
SDNET 2018	Kaggle website	Classification	5600
Crack_semantic_Segmentation	Kaggle website	Semantic Segmentation	11298
Simplified segmentation dataset	The authors	Semantic Segmentation	7226
Extra segmentation version dataset	The authors	Semantic Segmentation	12544

Table 4: Main datasets (Two datasets Originally from Kaggle websites and the other two are made by authors after modifications to find the best dataset)

The semantic segmentation datasets (images, masks) and the technique of splitting the data into training, validation, and testing sets enhanced the effectiveness of CNNs in recognizing and classifying images. After using a large and diverse dataset and by splitting the data, the CNN models were trained more effectively, and improved their ability to recognize and classify new images.

3.3 Loading data

The function takes in four arguments: the path to the directory containing the images, the path to the directory containing the corresponding masks, and the desired height and width of the resized images. The first step in the data loading process involves reading the images and masks using the OpenCV library. This is done using a list comprehension that iterates over each file in the specified directory. The "cv2.imread" function is used to read the images as colour images (i.e., with three channels for RGB) and the masks as grayscale images (i.e., with a single channel). The "os.path.join" function is used to concatenate the directory path and the file name for each image and mask. The resulting list of images and masks is then sorted using the "sorted" function to ensure that they are in the same order.

Next, the images and masks are resized using the "cv2.resize" function to the specified height and width. This is done to ensure that all the images and masks are of the same size and to reduce the computational complexity of subsequent operations. The resized images and masks are then stored in two separate lists. Then, the function returns the images and masks as NumPy arrays, which are commonly used in deep learning frameworks for training and testing machine learning models.

3.4 Preprocessing the dataset.

The image data is pre-processed to be used in computer vision tasks. The resizing step ensures that all images and masks are of the same size, while the normalization step improves the performance of machine learning models by standardizing the pixel values. The use of NumPy arrays and the reshaping step allows for efficient manipulation and processing of large datasets.

3.5 Using traditional image processing techniques (Canny edge detection)

Canny Edge Detection is a multi-stage algorithm designed to detect a wide range of edges in images while addressing three main issues: low error rate, edge localization, and single response constant. The first step in the process is *Noise Reduction*: The first step in Canny's edge detection method is noise reduction. Images can contain various forms of noise, which can adversely affect the edge detection process. Therefore, the image is smoothed using a Gaussian filter to reduce noise. Second is the *Gradient Calculation*, where the smoothed image is then filtered with a Sobel kernel in both horizontal and vertical directions to get the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From these two images, we can find the edge gradient and direction for each pixel. *Non-Maximum Suppression* is the third step which helps to suppress any pixel value that is not considered to be an edge. This is done by comparing each pixel of the gradient image with its neighbours along the gradient direction. If the pixel has a value that is less than its neighbour, it is set to zero. The next step is *Double Thresholding*; to determine potential edges, two threshold values (minVal and maxVal) are set. Any edges with an intensity gradient more than (maxVal) are considered "sure edges", while those below (minVal) are considered non-edges, and those in between are classified as "weak edges". *Edge Tracking by Hysteresis* is the final step where the weak edges are analyzed for connectivity to

sure edges. If they are connected, they are considered part of the edges. Otherwise, they are discarded. The result of this process is a binary image with "thin edges". The Canny edge detection algorithm is widely used due to its good detection, localization, and minimal response features.

3.6 Using Basic CNN models.

In the present study we use Visual Studio Code as an integrated development environment (IDE) to program and analyse data derived from a dataset of images. For programming, the two major languages under consideration are MATLAB algorithms and Python. However, Python was chosen for its flexibility and extensive libraries. To properly implement these libraries, OpenCV, TensorFlow, and other libraries were employed. To link Python to Visual Studio Code, extensions will be utilized to ensure compatibility with the Python environment. Those libraries helped to reach the purpose of the project which is recognizing the cracked images from non-cracked ones. The goal of this thesis is not to evaluate CNN Models but to find the most accurate model in detecting cracks in the given images to the trained model.

In the realm of Convolutional Neural Networks (CNNs), the selection of appropriate models is critical given the multitude of options available. The accuracy of the models in detecting images was assessed, and the test image filtered to match the trained models. Based on this comparison, the system was determined whether the image contains a crack. While this approach is the current focus of the study, it is subject to change and further development as the project progresses.

3.7 Using Semantic Segmentation Models

U-Net, SegNet, and DeepCrack are all encoder-decoder architectures suitable for binary crack detection tasks. U-Net is known for its skip connections, enabling it to capture fine-grained details effectively. SegNet, on the other hand, is more memory efficient and suitable for large-scale segmentation tasks. DeepCrack is a specialized model designed explicitly for crack detection and segmentation, incorporating batch normalization and ReLU activation functions for improved performance.

We have decided to try those three architectures (U-Net, SegNet, DeepCrack) to see which one is suitable to be enhanced and developed by measuring the accuracy, loss, and F1 score of those architectures and their trained models. Their implementation will be explained as follows:

3.7.1 U-Net

U-Net is a powerful and efficient architecture for binary segmentation tasks, such as crack detection. Its encoder-decoder structure captures both local and global contexts while maintaining the resolution and spatial information of the input image. The encoder part consists of a series of convolutional and pooling layers, progressively reducing the spatial dimensions while extracting hierarchical features. The decoder part, on the other hand, up-samples and

concatenates feature maps from the encoder to recover the resolution and provide precise pixel-wise predictions.

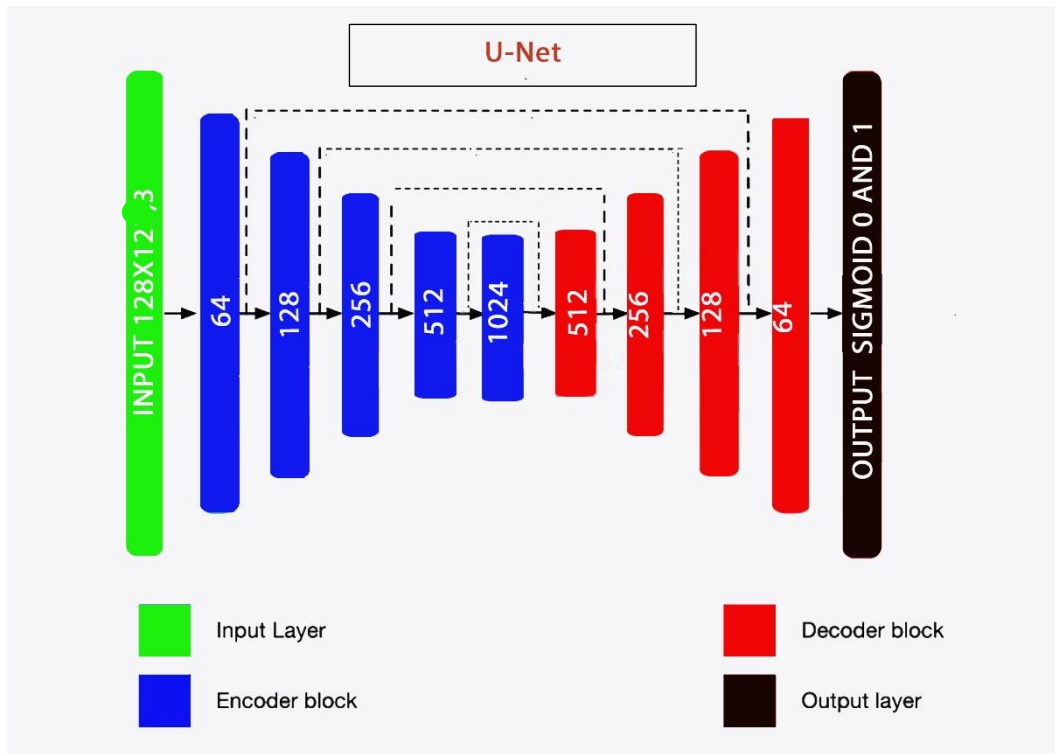


Figure 3: Original U-Net Model architecture encoder & decoder

One of the key advantages of U-Net is its skip connections, the skip connection is the group of algorithms as connections of the same level of feature maps from the encoder to the decoder. These connections ensure that high-resolution features are merged with up-sampled low-resolution features, improving the model's ability to capture fine-grained details such as the edges of cracks. This makes U-Net particularly well-suited for crack detection and binary segmentation tasks, where precise localization of cracks is crucial.

3.7.2 SegNet

SegNet is another architecture designed for semantic segmentation tasks, including binary crack detection. The architecture consists of an encoder-decoder structure, where the encoder follows a structure like the VGG16 model, progressively extracting features through convolutional and max-pooling layers. The decoder in SegNet, however, differs from U-Net as it does not utilize skip connections to merge encoder features with the up-sampled feature maps directly.

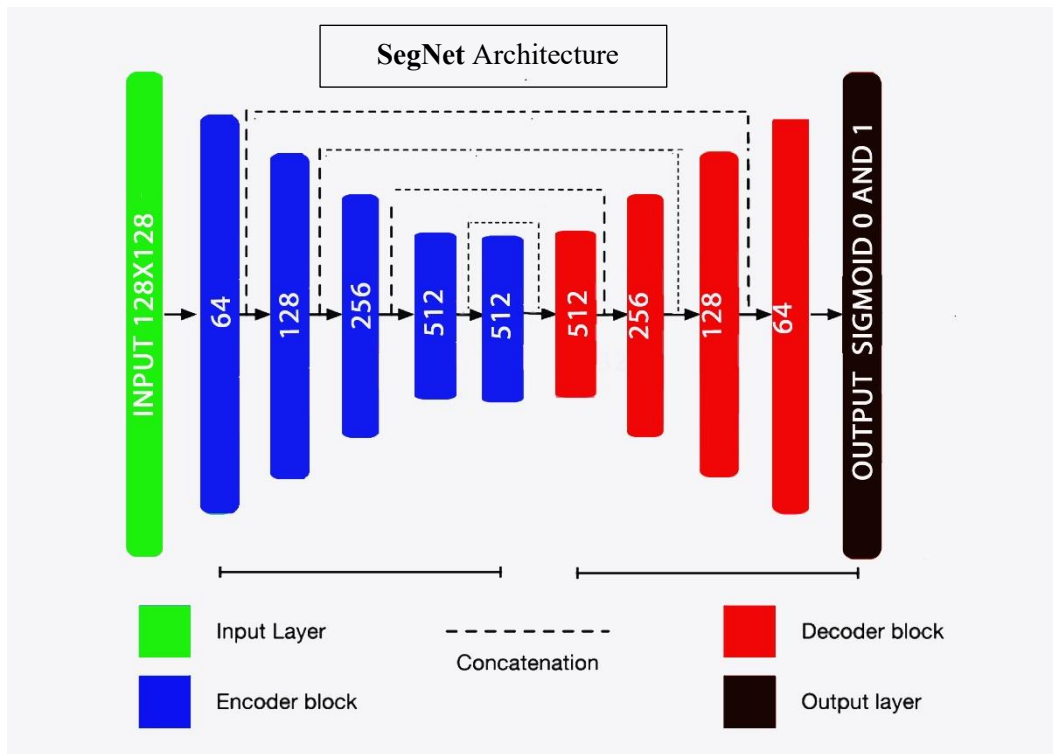


Figure 4: SegNet original model architecture

Instead, SegNet's decoder stores and uses the max-pooling indices from the encoder to up-sample feature maps in the corresponding decoder layers. This approach reduces the number of trainable parameters and memory requirements, making SegNet a more efficient architecture, especially for large-scale segmentation tasks. Although SegNet may not capture fine-grained details as effectively as U-Net due to the lack of skip connections, it still provides satisfactory results for binary crack detection, while offering a more memory-efficient solution.

3.7.3 DeepCrack

DeepCrack is a specialized architecture explicitly designed for crack detection and segmentation. It adopts an encoder-decoder structure with a unique focus on preserving spatial information throughout the model. Deep Crack's key difference from both U-Net and SegNet lies in the use of batch normalization which is a technique that normalizes the output of each layer in a mini-batch, stabilizing training, improving convergence, and preventing overfitting in semantic segmentation tasks and ReLU activation functions consistently after each convolutional layer, enhancing the model's performance by stabilizing the training process.

Deep Crack's encoder-decoder design allows it to effectively capture both global and local context for accurate pixel-wise classification in binary crack detection tasks. Although the architecture resembles U-Net and SegNet, it has been specifically tailored for crack segmentation, making it highly effective for the target application. The use of batch normalization and ReLU activation functions ensures a stable and efficient training process, resulting in a high-performance model for crack detection and segmentation.

3.8 Improving U-Net architecture.

We decided to improve the U-Net model by adding pretrained models after the fact that U-Net gave better scores in general than the other two models SegNet and DeepCrack Architecture to the existing one (U-Net) to make improvements. These models were VGG-16, VGG-19, and ResNet architectures. They were implemented as encoders of the new experimental model in its first half, and the second half (decoders) were implemented as they originally are (U-Net-decoders). This was a try to enhance the model's ability to detect cracks more accurately than the original U-Net (See Figure 5).

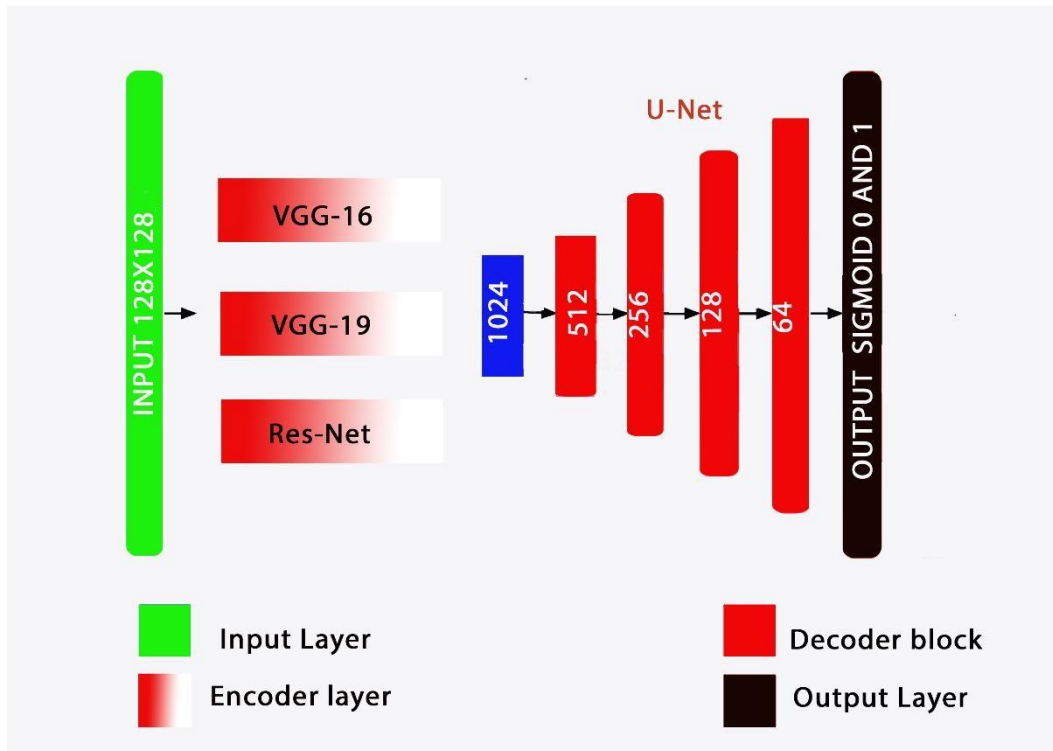


Figure 5: Encoders model with U-Net decoder

3.8.1 VGG-16 with U-NET

VGG16-U-Net is a semantic segmentation model designed for binary crack detection. This architecture combines the VGG16 backbone with the U-Net decoder structure, leveraging the robust feature extraction capabilities of VGG16 and the efficient upsampling and skip connections of U-Net. By using VGG16 as a pre-trained encoder, the model can quickly learn and adapt to various image patterns. The U-Net decoder, on the other hand, helps preserve spatial information throughout the layers, which is crucial for accurate segmentation.

3.8.2 VGG-19 with U-NET

We have decided to use VGG -19 to see the differences between VGG-16 which was explained briefly in the previous section to compare the two enhanced U-Net structures and see if VGG-19 could outmatch VGG-16 in terms of accuracy and detecting cracks. There is no need to mention the structure of VGG-19 here because it is like the VGG-16 Combined with the U-Net

Structure just above. The results of all comparisons of those models will be in the results and discussion section.

3.8.3 ResNet with U-NET

ResNet-U-Net is a semantic segmentation model tailored for binary crack detection tasks. This architecture fuses the power of ResNet as the encoder and the U-Net decoder structure. ResNet provides strong feature extraction capabilities due to its deep residual learning framework, while the U-Net decoder incorporates upsampling and skip connections to preserve spatial information and enable precise segmentation.

The ResNet-U-Net model takes advantage of ResNet50's deep residual structure as the encoder backbone, consisting of a series of residual blocks which refers to skip connections that allow the direct flow of information, aiding in the preservation of spatial details and improving the performance of deep neural networks. The U-Net decoder follows the typical U-Net approach, combining encoder features with corresponding decoder layers via skip connections. The final output layer uses a sigmoid activation function to provide binary segmentation.

3.9 Extra Layer technique

An innovative approach to enhancing the U-Net architecture was explored, involving the incorporation of an additional layer. We have added new extra 32 filters convolutional layer. This extra layer was mirrored in the decoder, thus expanding the architecture on both ends. The expanded architecture has implications for a variety of areas, including feature extraction, performance metrics, model capability, hierarchical feature learning, and model adaptability.

The enhanced feature extraction ability provided by the additional layer facilitates a more intricate representation of the input images. This, in turn, aids in detecting complex patterns crucial for the precise segmentation of images. Performance metrics such as accuracy and F1 scores show notable improvements, attributed to the enhanced feature extraction. By capturing intricate features, the expanded architecture can deliver more accurate segmentation, proving beneficial for multiple applications.

The added layer also expands the model's capacity to handle complex real-world data variations. However, the balance between the benefits of increased complexity and the accompanying computational costs and overfitting risks is critical. The hierarchical feature learning capability of the model is also enhanced by the extra layer, allowing the model to learn features at different scales and levels of abstraction. This aids in capturing both local and global patterns in the images. Furthermore, the extra layer provides flexibility for customization, allowing researchers to experiment with different layer configurations to find the optimal structure for a specific task or dataset.

4 Result and discussion

Before showing the results of this study, it is recommended to investigate ethical and social considerations in engineering and academic writing, focusing on the application of CNN and semantic segmentation for crack detection in infrastructure. Key ethical concerns include fairness, bias, data security, privacy, responsibility, and transparency. From a societal perspective, the project's economic impact includes cost savings from timely crack detection and potential job creation. Environmentally, the technology can reduce carbon emissions by preventing infrastructure damage. Societally, it enhances public safety, improves transportation, and minimizes disruptions. In this study, we investigated various deep-learning models for the semantic segmentation of cracks in images. The results have been divided into six sections, which they are Canny Edge results, Semantic segmentation comparison results, pre-trained vs U-Net results, new improved extra layer vs U-Net results, datasets comparison results and finally plots of F1 score.

The research journey was not without its trials. We first attempted to add two layers: one each in the encoder and decoder, adjacent to the 1024 filters bridge and extended the bridge to 2048 filters while adjusting the neighbouring layers to (1024, 1024). This experiment, however, was unsuccessful due to overfitting and the large number of parameters involved. A subsequent attempt involved modifying the bridge layer between the encoder and decoder, expanding it to 2048 filters instead of the standard 1024. This effort, aimed at maximizing the potential of the U-Net architecture, also led to overfitting. We further experimented with adding 64 layers as the initial decoder and final encoder layer. They utilized the concatenate function to append additional encoder and decoder layers at the start and end of the architecture, respectively. However, this model did not outperform the original U-Net structure.

In the final try, the breakthrough came when we added an extra layer with 32 filters at the beginning of the encoder, preceding the conventional 64 layers, and at the end of the decoder, succeeding the final 64 layers. This resulted in a model beginning and concluding with the newly added 32 filters layer. As detailed in section 1.5 of this thesis, this final model proved successful, thereby enhancing the conventional U-Net architecture in a significant and innovative way.

4.1 Canny Edge detection method result

The authors implemented Canny Edge method as mentioned in method section; to be the first step of process of finding best method to read cracks. As the figure below shows the detecting of cracks was successful, but with a lot of noise. That could harm the final expected results to deliver to the system or PLC part in the user's side which leads to misjudgements.

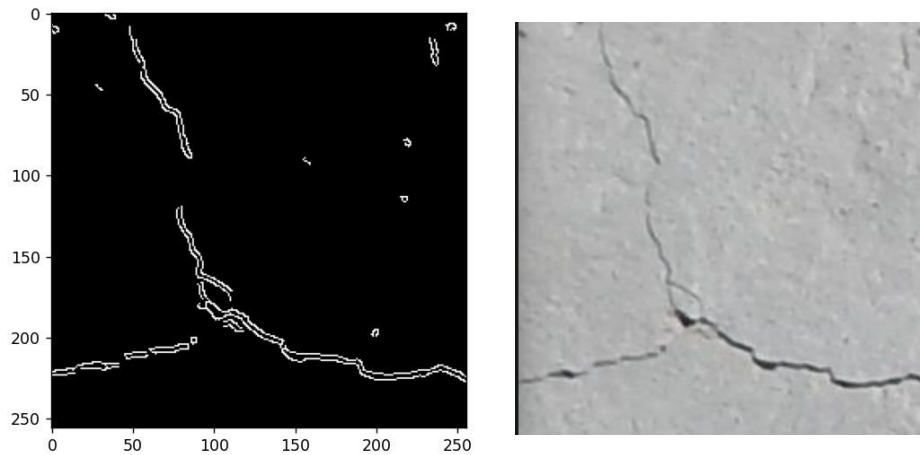


Figure 6: Canny Edge detection (On the left is the canny edge result image with noise... On the right is the original input image in the algorithm)

4.2 Semantic Segmentation comparison

We have focused on evaluating the performance of three semantic segmentation models: U-Net, SegNet, and DeepCrack. These models were trained on a dataset of images containing cracks and were subsequently tested on a set of cracked images to assess their ability to detect and segment the cracks. We have quantitatively analysed their performance using evaluation metrics such as accuracy, loss, F1 score, precision, and recall. In the results section, we will present a comprehensive comparison between the three models in terms of these evaluation metrics. This comparison presented in table 5, allowing for easy interpretation of the results. It should be mentioned that the tested trained models were having 50 Epochs and the batch size was 16 for all three models in this comparison (Check Table 5 and Table 6).

Model	Precision%	Recall%	F1 Score	Accuracy%	Loss
U-Net	<u>75</u>	<u>62</u>	<u>68</u>	<u>98</u>	<u>0.024</u>
SegNet	77	55	65	97	0.031
DeepCrack	69	53	60	96	0.033

Table 5: U-Net, SegNet and Deep Cracks comparison in (Precision, recall, F1 score, accuracy and loss and reader can see that U-Net outmatched the others two)

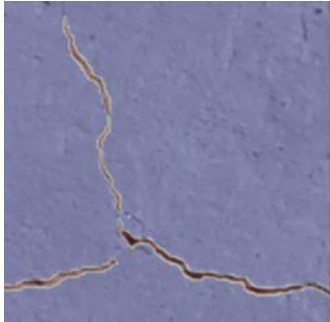

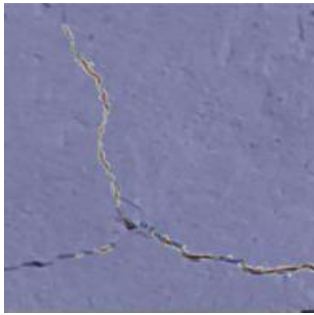
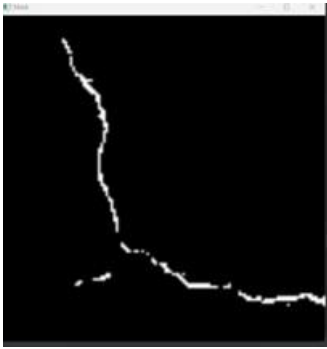
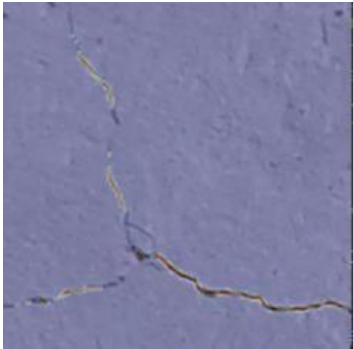
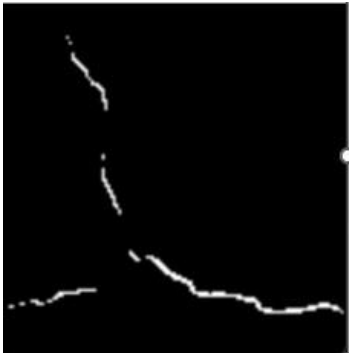
Model	Blended	Masked
U-Net		
SegNet		
DeepCrack		

Table 6: Compare between U-Net, SegNet and Deep Cracks (on the left is the blended image “ how the layers read the cracks while on the right is the final output) we can see here also how U-Net was better than other two models)

After analysing the results in Table 5 and 6 above, we found that the U-Net model consistently demonstrated higher accuracy and robustness in detecting cracks compared to the SegNet and DeepCrack models. Consequently, we decided to focus on the U-Net architecture for further experimentation.

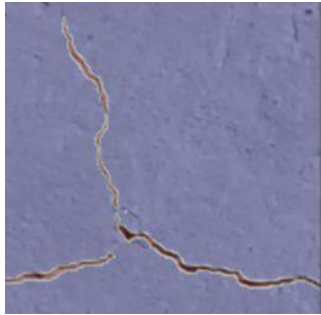

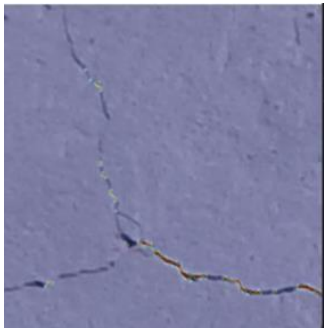

4.3 Combined pre-trained models with U-Net comparison.

We explored the combination of U-Net with popular CNN architectures such as VGG-16, VGG-19, and ResNet. Our goal was to improve the performance of the crack detection process by leveraging the strengths of these pretrained models. We presented a detailed comparison of the performance of the combined models, highlighting the benefits of using the U-Net architecture as a backbone. The tested trained models were having 50 Epochs and the batch size was 16 for all three models in this comparison. (See Table 7).

Model	Precision%	Recall%	F1 Score	Accuracy%	Loss
<u>U-Net</u>	<u>75</u>	<u>62</u>	<u>68</u>	<u>98</u>	<u>0.024</u>
VGG-16-U-Net	72	40	51	95	0.037
VGG-19-U-Net	70	44	54	96	0.035
ResNet-U-Net	73	38	49	94	0.045

Table 7: VGG16,19 and ResNet With U-Net decoder comparison. (It can be noticed here how U-Net is still better than the pre-trained models as encoders)

Next is the table of the image's comparison between the three new architectures VGG-16 as an encoder combined with U-Net, VGG-19 combined with U-Net structure and finally the ResNet-U-net architecture. All three compared in the next table in terms of output-masked images to see which model of those four is still the most accurate to detect cracks (See Table 8).

Model	Blended image	Output (masked) image
U-Net		
Vgg-16/U-Net		

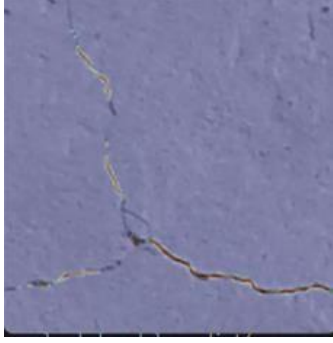

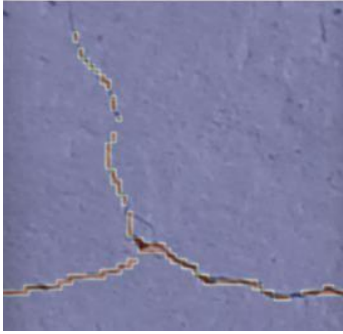
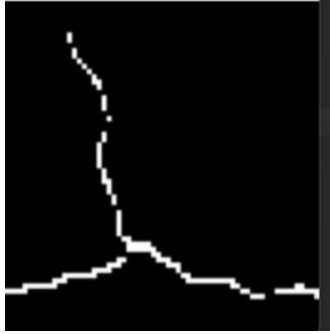
Vgg-19/U-Net		
ResNet/U-Net		

Table 8: Compare U-Net with pre-trained models' comparison of images and it can be noticed again how U-Net is better.

4.4 New Improved Extra Layer U-Net Vs Original U-Net

The new architecture has been introduced as an innovative extra layer to strengthen the basic U-Net structure. We implemented this extra layer into the U-Net architecture and evaluated its performance in comparison to the original U-Net model. Our results demonstrate that the addition of this extra layer significantly enhances the U-Net model's ability to detect cracks, providing further evidence for the efficacy of our proposed method. See the results in the next two comparison tables 9 and 10.

Model	Precision%	Recall%	F1 Score	Accuracy%	Loss
HH-Improved Extra Layer (U-Net)	<u>90</u>	<u>60</u>	<u>72</u>	<u>98</u>	<u>0.0168</u>
Original (U-Net)	75	62	68	98	0.024

Table 9: Compare Between Original U-Net and HH-U-Net (evaluation metrics) "Can be noticed clearly how our work is better than the original U-Net).

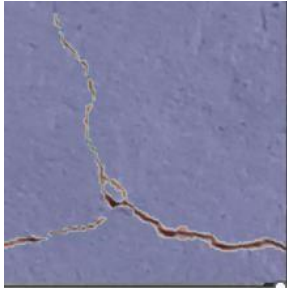
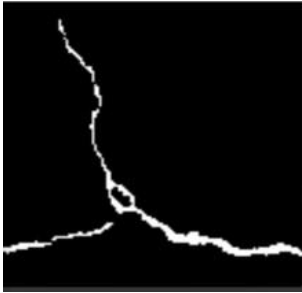
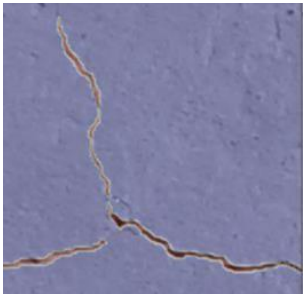

Model	Blended image	Output Masked image
HH-Improved Extra Layer (U-Net)		
Original (U-Net)		

Table 10: Compare Between Original U-Net and HH-U-Net (Images) “our model is outmatching and outperforming the original U-Net”.

4.5 Authors dataset(Ex) Vs Original on Extra layer U-Net

The second contribution that was mentioned in contributions section 1.5.2 was the newly made Extra dataset. This new dataset was more efficient even on the new extra layer architecture U-Net, which means that the trained model on this extra new dataset got a higher F1 score, higher Accuracy, and better loss than the improved Extra U-Net that was trained on the original dataset from Kaggle “crack_segmentation_dataset”. See table 11 for more details.

Dataset	Precision%	Recall%	F1 Score	Accuracy%	Loss
Authors’ Extra Improved Dataset (HH-Improved Extra Layer (U-Net))	<u>96</u>	<u>61</u>	<u>75</u>	<u>99</u>	<u>0.0146</u>
Original Dataset HH-Improved Extra Layer (U-Net)	90	60	72	98	0.0168

Table 11: Authors’ Extra Improved dataset and original dataset “our new dataset is outperforming the original dataset”.

4.6 F1 score plots comparison

In the presented research, a comprehensive approach was employed to identify cracks, leveraging various computational techniques from the domains of image processing and computer vision. The methodologies applied can be broadly categorized into two primary themes: traditional image processing techniques and machine learning-based strategies. As a representative of conventional techniques, the Canny Edge detection method was utilized. Following this, the study ventured into the realm of machine learning, employing a Basic Convolutional Neural Network (CNN), and subsequently, delving into more sophisticated techniques such as Semantic Segmentation.

Among the numerous architectures explored within the scope of Semantic Segmentation, the primary focus was on U-Net, SegNet, and DeepCrack. From the U-Net family, several pre-trained models were tried, including VGG16 and VGG19. However, the U-Net variant that incorporated an extra layer, referred to as the Improved U-Net (HH), yielded the most promising results in terms of accuracy and F1 score. Furthermore, the study revealed that the introduction of a novel, modified extra dataset had a significant positive impact on the performance, pushing the F1 score to new heights. The ensuing section presents a comparative illustration of the F1 scores generated by different methodologies (See figure 7).

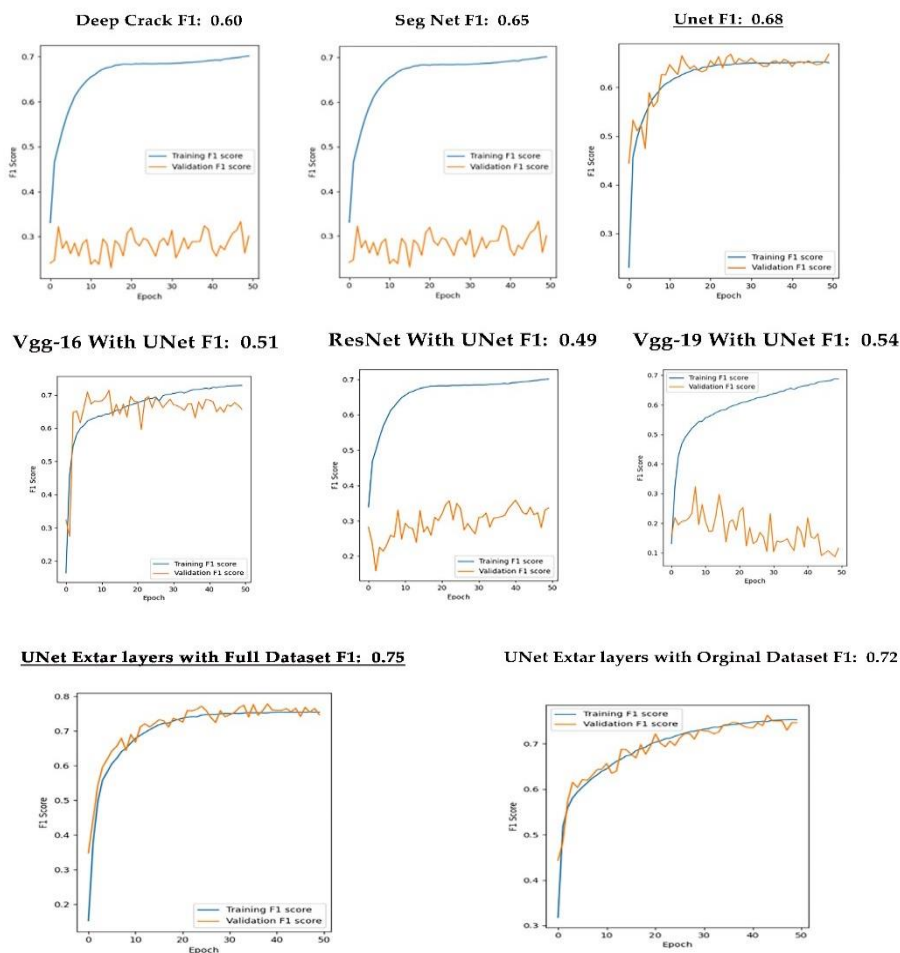


Figure 7: F1 Score all models “we can notice how our new made extra improved Model of U-Net is beating the other models in F1 score”.

Upon examining the plots, two significant contributions from the authors are readily observable. The first notable finding is related to the implementation of the newly developed architecture - the Improved Extra-Layer U-Net (HH). This architecture achieved an F1-Score of 72, surpassing the performance of all other architectures tested within this study. This robust outcome substantiates the superiority of the HH model in the specific task of detecting cracks in sand mould samples.

The second key contribution is the augmentation of the original dataset with a newly collected, additional dataset. This enhancement not only diversified the data available for model training but also resulted in a substantial improvement in the F1-Score, increasing it from 72 to 75. This underlines the efficacy of the extra dataset in enhancing model performance, thereby affirming the importance of diverse and representative data in machine learning tasks.

5 Conclusion and future work

In this study, we have extensively investigated various semantic segmentation architectures to accurately detect cracks in sand mould structures in factory line production. The primary focus was on three main architectures: U-Net, SegNet, and DeepCrack. Through rigorous experimentation and testing, we have found that the U-Net architecture provided the best results on our chosen tuned dataset for this specific application.

Subsequently, we explored the possibility of utilizing pre-trained models (VGG-16, VGG19, ResNet) as encoders for the U-Net's original structure decoder. However, this approach did not yield satisfactory results. Motivated by this observation, we transitioned to the third section of our work, which involved the development of an improved U-Net architecture by incorporating an extra layer. We addressed potential drawbacks, such as overfitting, through the creation of a suitable variable augmented dataset, early stopping mechanisms, and checkpoints for the model.

Our study concludes that the newly developed improved U-Net architecture, with the inclusion of an extra layer, was a success. It outperformed other architectures in terms of F1 score, accuracy, and other evaluation metrics. Furthermore, qualitative image analysis revealed that our improved extra-layer U-Net was more adept at detecting cracks than the other architectures examined in the thesis experiments.

Moreover, we successfully created one of the most accurate datasets for cracks in sand moulds. This dataset was curated based on the original Kaggle website dataset by adding two groups of images to the original groups of images which noticeably raised the F1 score. This dataset is a valuable resource that can be utilized by future studies or improved upon to facilitate even more precise and accurate crack detection. Our research sets the stage for further advancements in the field of semantic segmentation, with the potential to enhance quality control and reduce production costs in the manufacturing industry.

As for future work, our study opens doors for further exploration in several directions. First, other research teams may consider building upon our improved U-Net architecture or enhancing other architectures like SegNet. The versatility of our approach allows for the investigation of different modifications or layer configurations, thereby fostering novel and innovative techniques in semantic segmentation.

Second, researchers could continue comparing various models on different datasets to better understand the adaptability and robustness of these models under varying conditions. Our work, along with future studies, may contribute to the development of a comprehensive understanding of which models and techniques perform optimally under different circumstances.

6 References

- [1] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," 1959.
- [2] L. Xuanran, Z. Liqiang, W. Yaodong and Y. Zujun , "A crack detection system of subway tunnel based on image processing," *Measurement and Control*, vol. 55, pp. 164-177, 04 05 2022.
- [3] B. Kim and S. Cho, "Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique," *Sensors*, vol. 18, p. 3452, 14 10 2018.
- [4] Y. Tomoyuki and H. Shuji, "Fast crack detection method for large-size concrete surface images using percolation-based image processing," *Machine Vision and Applications*, vol. 21, pp. 797-809, 11 02 2009.
- [5] A.-Q. Ikhlas, O. Abudayyah and M. Kelly, "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *Journal of Computing in Civil Engineering*, vol. 17, pp. 255-263, 1 10 2003.
- [6] A. Reza and J. Hashim, "Structural Damage Identification Based on Modal Data and Wavelet Analysis," *Science topic*, 1 10 2012.
- [7] L. S. Ronny, M. Hung and S. Weihua, "A Robotic Crack Inspection and Mapping System for Bridge Deck Maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, pp. 367-378, 01 04 2014.
- [8] F. Liang, L. Quanqing and S. Hongchun, "Research on Fatigue Crack Growth Detection of M (T) Specimen Based on Image Processing Technology," *Journal of Failure Analysis and Prevention*, vol. 18, pp. 1010-1016, 12 06 2018.
- [9] C. V. Dung and L. D. Anh, "Autonomous concrete crack detection using deep fully convolutional neural network," *Automation in Construction*, vol. 99, pp. 52-58, 01 03 2019.
- [10] Y. Yuan, Z. Ge, X. Su, X. Guo, T. Suo, Y. Liu and Q. Yu, "Crack Length Measurement Using Convolutional Neural Networks and Image Processing," *Sensors*, vol. 21, p. 5894, 01 09 2021.
- [11] N. A. M. Yousef, A. Ibrahim, M. H. M. Noor, N. M. Tahir, N. M. Yusef, N. Z. Abidin and M. K. Osman, "Deep convolution neural network for crack detection on asphalt pavement," *Journal of Physics: Conference Series*, vol. 1349, p. 012020, 01 11 2019.
- [12] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghala , S. Pandya, K. Modi and H. Ghayvat, "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," *Electronics*, vol. 10, p. 2470, 11 10 2021.

- [13] L. Ali, F. Alnajjar, H. Jassmi, M. Gocho, W. Khan and M. .. Serhani, "Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures," *Sensors*, vol. 21, p. 1688, 01 03 2021.
- [14] L. Nieradzick, G. Scheueermann, D. Saur and C. Gillmann, "Effect of the output activation function on the probabilities and errors in medical image segmentation," *arXiv*, p. 2109, 02 09 2021.
- [15] L. Alzubaidi, J. Zhang, A. Humaidi, A. Al-dujaili, Y. Duan and O. Al-Shamma, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, 31 03 2021.
- [16] Y. Hamishbahar, H. Guan, S. So and J. Jo, "A Comprehensive Review of Deep Learning-Based Crack Detection Approaches," *Applied Sciences*, vol. 12, no. 3, p. 1374, 27 01 2022.
- [17] Y. Liu, J. Yao, X. Lu, R. Xie and L. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139153, 01 04 2019.
- [18] R. Pu, G. Ren, H. Li, W. Jiang, J. Zhang and H. Qin, "Autonomous Concrete Crack Semantic Segmentation Using Deep Fully Convolutional Encoder–Decoder Network in Concrete Structures Inspection," *Buildings*, vol. 12, no. 11, p. 2019, 18 11 2022.
- [19] J. Tian, N. Mithun, Z. Seymour, H.-P. Chiu and Z. Kira, "Striking the Right Balance: Recall Loss for Semantic Segmentation," *arXiv*, p. 1, 03 02 2022.
- [20] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv*, p. 1, 29 01 2017.