Deca Convolutional Layer Neural Network (DCL-NN) Method for Categorizing Concrete Cracks in Heritage Building

Dinar Mutiara Kusumo Nugraheni*[1], Andi Kurniawan Nugroho*[2], Diah Intan Kusumo Dewi[3], Beta Noranita[4]
Informatic Department-Faculty of Science and Mathematics, Universitas Diponegoro, Semarang, Indonesia[1, 4]
Electronic Department-Faculty of Engineering, Semarang University, Semarang, Indonesia[2]
Urban Regional Planning Department-Faculty of Engineering, Universitas Diponegoro, Semarang, Indonesia[3]

Abstract*—It is critical to develop a method for detecting cracks in historic building concrete structures. This is due to the fact that it is a method of preserving historic building and protecting visitors from the collapse of a historic structure. The purpose of this research is to determine the best method for identifying cracks in the concrete surface of old buildings by using cracked images of old buildings. The various surface textures, crack irregularities, and background complexity that distinguish crack detection from other forms of image detection research present challenges in crack detection of old buildings. This study presents a framework for detecting concrete cracks in old buildings in Semarang's old town using a modified Convolutional Neural Network with a combination of several convolutional layers. This study employs ten convolutional layers (Deca Convolutional Layer Neural Network (DCL-NN)) to provide mapping features for images of concrete cracks in ancient buildings at preservation area. This study also compares commonly used machine learning models such as KNeighbors (n neighbors=3), Random Forest, Support Vector Machine (SVM), ExtraTrees (n estimators=10), and other CNN-pretained models such as VGG19, Xception, and MobileNet. Four performance indicators are used to validate each model's performance: accuracy, recall, precision, F1-score, Matthews Correlation Coefficient (MCC), and Cohen Kappa (CK). This study's data set is comprised of primary data obtained from cracked and normal images of several buildings in Semarang's old town. The accuracy of this study using DCL-NN is 98.87%, recall is 99.40%, precision is 98.33%, F1 is 98.86%, MCC is 97.74%, and CK is 98.86% for crack class. From this study, it was found that the ten convolution layers have higher classification performance compared to other comparison models such as machine learning and other CNN models and are more effective in detecting cracks in concrete structures.

***Keywords*—Cracks; concrete; Deca-CNN; features mapping; performance**

I. INTRODUCTION

It is critical to understand the shape of cracks in historic buildings in order to preserve historic areas. The Old Town area of Semarang, which is a UNESCO World Heritage Site, is one area that is vulnerable to building cracks [1]. The total number of buildings, which reached 274 units, demonstrates that the old town area was previously a residential area; now, out of a total of 157 units with the status of occupied buildings (both for housing and offices, dominated by offices), 87 units are status as vacant buildings (both those that are still being maintained or damaged/abandoned), 28 units are leased (offices), and only 2 units are currently sold [2]. Historic ancient buildings are architectural creations that serve as a nation's cultural heritage and have very high artistic and historical values, so their long-term viability must be ensured. The ability of a sturdy building structure supports the building's strength, which causes the building to last a long time. Cracks in concrete structures are a common sign of faulty concrete. The presence of cracks affects the structural condition and increases the risk of unexpected damage and collapse of the building [3][4]. Therefore, crack detection must be done on a regular basis in order to maintain the concrete structures of historic ancient buildings.

II. RELATED WORKS

Traditionally, professional images of cracks in a building's concrete structure are used. The use of an expert to inspect concrete structures for cracks is costly, time-consuming, and sometimes dangerous for direct inspection [5]. The Ultrasonic Pulse Velocity Test (UPVT) in the form of ultrasonic waves in the cracks where holes are made for the ultrasonic wave propagation area is then used as another method of measuring building cracks [6]. This creates the possibility for buildings to become more dangerous as a result of the influence of other building structures. However, as computer-aided design (CAD)-based image processing technology advances, many experts are turning to machine learning-based image processing for the automatic detection of cracks in concrete structures [7]. Many techniques for detecting building cracks have been proposed by researchers, including the use of thresholding methods [8], edge detection, and wavelet transform. Surface texture, crack irregularity, and background complexity distinguish crack detection from other images in research that leads to machine learning-based image processing solutions for automatic detection of cracks in concrete structures.

Deep learning-based models, especially multilayer neural networks, currently play an important role in feature learning [9]. Moreover, the availability of high-performance computers and the continuous improvement of good training methods on

*Corresponding Author.

Universitas Diponegoro Research Publication International {RPI} Grant SK. Rector Universitas Diponegoro No: 215 /UN7.A/HK/VII/2022; No.SPK:569-108/UN7.D2/PP/VII/2022

available datasets are driving the rapid development of deep learning. Conversely, convolutional neural networks (CNNs) are feed-forward neural networks characterized by high-resolution image processing [10]. Some of these models are suitable for feature extraction in various applications, but their accuracy needs to be improved to detect cracks in concrete.

In this work, CNN-based transfer learning using pre-trained models to achieve efficient performance, reduce training time, overcome the drawbacks of large datasets, and yield significant results. A law has been proposed [11]. Some previous researchers have done some work to identify and classify cracks in buildings. Zhang et al. [12] proposed a 6-layer convolutional neural network (CNN) architecture for road crack detection and used 640,000, 160,000, and 200,000 images to train, validate, and validate the network. bottom. and tested. Kings. [13] proposed a CNN architecture with three convolutions and he two fully connected layers to detect cracks in asphalt. On the other hand, the author of this study says that for training he used 640,000 images and for testing he used 120,000 models. Fan et al. [14] proposed an efficient automatic road crack detection and measurement model based on an ensemble of CNN models. The authors of this study calculated the final failure probability by combining the probability values from each CNN model using the weighted overall average technique. Xu et al. [15] He trained a 28-layer end-to-end CNN model to detect cracks in concrete bridges. To obtain multiscale contextual information, the authors of this study used the concept of combining Atlas Spatial Pyramids (ASPP) and depth convolution to reduce the number of parameters in the network. This study describes a framework for detecting concrete cracks in old buildings in Semarang Old Town using a modified convolutional neural network that combines multiple layers of convolution. Scaffolding helps identify the presence and location of cracks in concrete surface patterns.

This study suggests early detection of cracks in historic buildings. Citra captures images that do not show cracks and uses existing image processing algorithms to distinguish them from images that show cracks. The uniqueness of this study is that this study DCL-CNN model to help identify and classify crack types. Do these cracks appear in concrete or only on the surface of old buildings in Semarang city? Because of the insufficiency of observational methods for concrete structures, the study intends to make the following major contributions:

1) An effective and efficient classification framework with a combination of the number of convolution layers based on crack candidate areas is proposed to effectively categorize cracks and non-cracks. The more feature mapping that results from the number of convolution layers, the more detailed the system will be in detecting cracks in surface structures and deep fracture structures.

2) Comparison with other transfer learning methods and machine learning so that the CNN model produces the desired performance in classifying cracked and normal building locations

The image of cracks in ancient and historic buildings in Semarang, Indonesia, was used as data in this study. In this study, crack classification in old town buildings in Semarang requires several stages, as illustrated in Fig. 1. Images of old building cracks are preprocessed. Preprocessing includes size adjustment, rotation, position translation, and flip processes.

The Citra image data set is split into a training data set and a validation (test) data set. Use preprocessed data to extract modal feature information from images using a pretrained model with transfer learning. This model is fed to a fully connected layer (FC) and trained after fusion. The top two layers of the FC layer contain 512 hidden units followed by the ReLU activation function. The final layer contains hidden units, followed by a sigmoidal activation function used for crack detection. Evaluate system performance using metrics such as Accuracy, Search Rate, Accuracy, F1 Score, MCC, and CK.

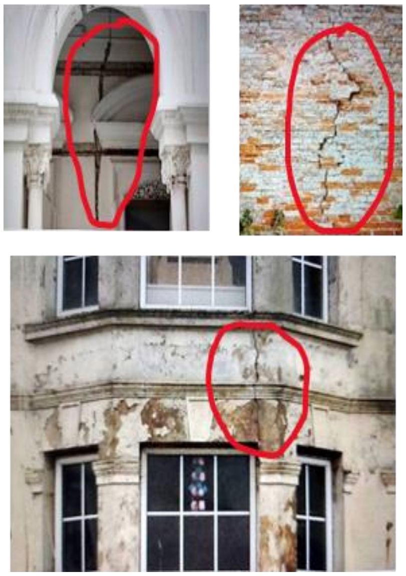


Fig. 1. Condition from the front side of the cracks of several ancient buildings in the old town of Semarang

III. Methodology

The methodology for this paper shows in the Fig. 2. At the beginning, we need to process the crack image into through several levels of pre-processing, data augmentation, and training for each model. It is including modifications to the CNN model, retained model, and machine learning model, and all algorithms are tested using data testing

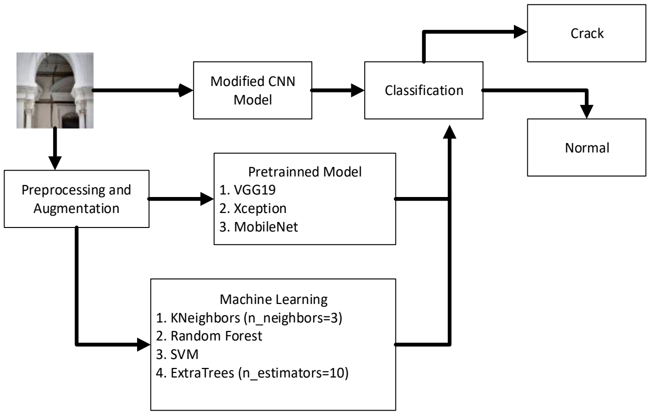


Fig. 2. Research methodology diagram

The experimental program is divided into four phases:

- creating a classified image dataset from primary data;
- developing the convolution layer from the CNN standard;
- comparing with machine learning and pre-trained models;
- running training experiments. The following sections provide information about each phase.

A. Image Data Set

The dataset employs for this study consists of 10,000 images with 512 x 512-pixel resolution. The image was taken from various concrete specimens after mechanical testing in the Semarang old town building. The main idea is to collect concrete service images from various surface views in order to diversify the data set and, as a result, the AI system that learns from this data set. The images were sliced into 224 x 224-pixel images to increase the data set without sacrificing resolution, resulting in a final data set of 10,000 samples, which were then manually classified into two categories: concrete surfaces with and without cracks. The dataset contains 5,000 images with cracks and 5,000 images without cracks. With a 70/30 split, the dataset is divided into training and validation datasets.

B. Convolution Layer from the CNN Standard

CNN1 is known as the base of the CNN method. For identified the crack building using Citra, this study used four variations of the CNN architecture. The usage of four variations of the CNN architecture used to find the best architecture for detecting the condition of cracks in ancient building structures in Semarang's old town. This study used GPU GTX 1650 RAM 2 x 8 GB 2400 MHz DDR4 for the computation.

Fig. 3 shows a diagram of the CNN1 architecture. This basic design can be extended to create CNN2 or CNN3 architectures. CNN3 is extensible to CNN4. CNN architectures are built to determine the impact of CONV layers and their activation functions on classification accuracy. Additionally, comparisons can be made between CNN1, CNN2, CNN3, and CNN4, or between design groups to determine if the best type of design is used for classifying old building crack images.

C. The Performance between Machine Learning and Pre-Trained Models

To evaluate classification performance, precision, recall, and accuracy matrices are used [16], [17]. To calculate the metric, add the sums of TP, FP, FN, and TN. True positive is represented by TP, false positive by FP, false negative by FN, and true negative by TN and properly measure ratios as positive detected elements, taking only positive predictions into account.

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (1)$$

The precision is stored in the denominator by FP; if it is high, the precision is low. However, the majority of the elements are predictably incorrect, and only a few are correct as positive, resulting in high precision values even if there are many FNs. As a result, a measure of the number of FN, namely recall, is required.

$$\text{Recall} = \frac{TP}{TP+FN}, \quad (2)$$

and correctly measure ratios as positively detected elements, taking into account only elements with positive ground truth annotations. If the FN amount is large, the drawdown will be small and measures the ratio of correct predictions to all predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}\ ,(3)$$

Accuracy can be used as a reliable summary metric for classification performance because the dataset is symmetric.

Table III displays the outcomes of all models in the primary dataset. To validate the model, we only use the test folder dataset. To compare performance, two statistical tests were performed: the Matthews correlation coefficient (MCC) and the Cohen's Kappa statistic [18-19]. In the case of unbalanced data sets, the Matthews correlation coefficient (MCC) is a popular performance metric. Despite the fact that the dataset used in this paper is balanced, it is defined by the mathematical equation number (4):

$$MCC = \frac{(TP\ x\ TN)-(FPxFN)}{\sqrt{(TP+FP)x(TP+FN)x(TN+FP)x(TN+FN)}} \quad (4)$$

The MCC range is [1-1]. A MCC value closer to one is preferable. All of the models that were used performed admirably. The value is close to 1. In other words, the model correctly classifies the fracture image. Cohen's Kappa statistic is used to assess the degree of agreement between two raters who categorize objects into mutually exclusive groups, as shown mathematically in Equation (5).

$$CK = \frac{(p_o - p_e)}{(1 - p_e)} \qquad (5)$$

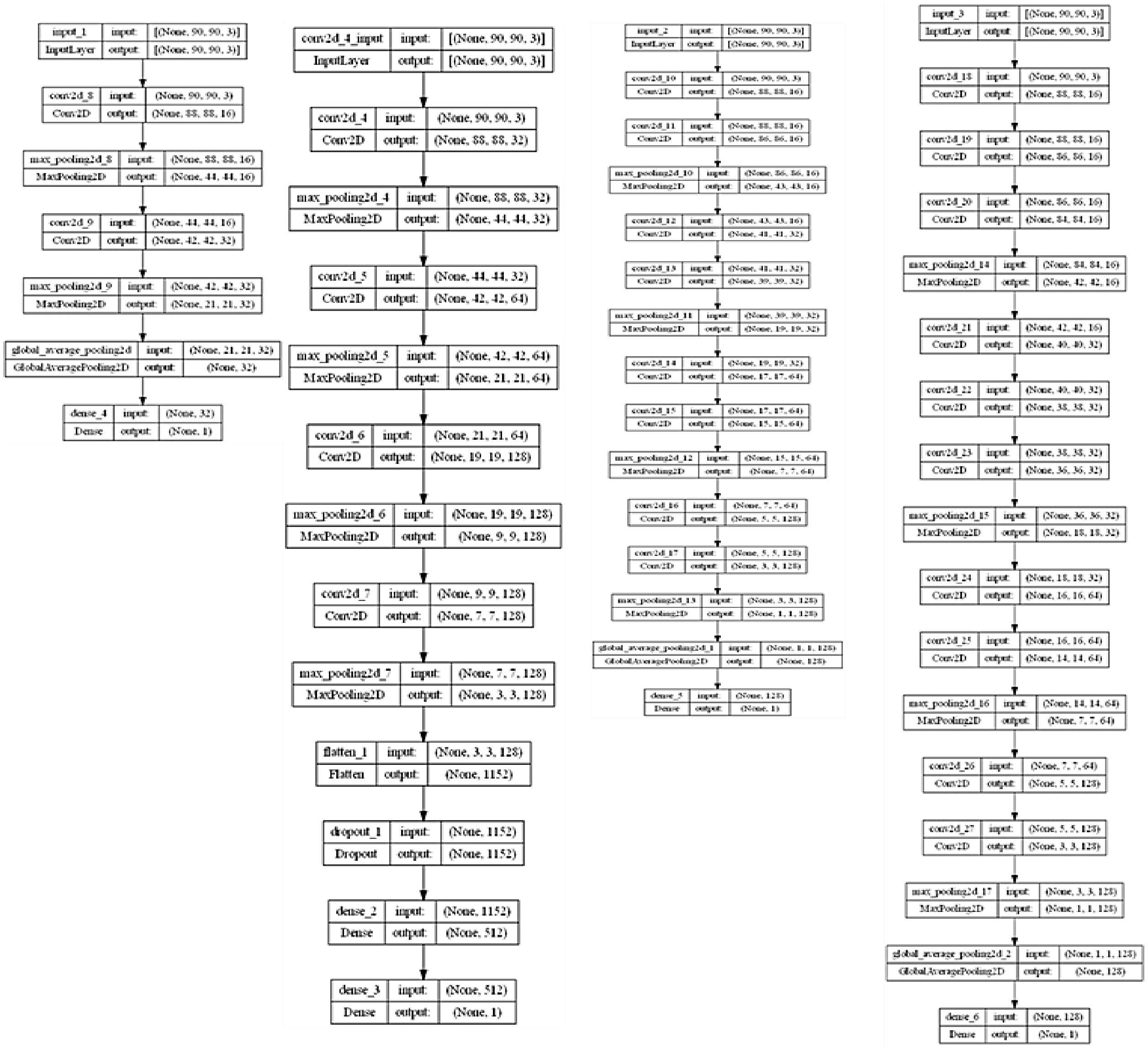
In this case, po represents the rater's observations' relative agreement. The theoretical probability of random agreement is denoted by pe. Using Equations (6)-(8), we can calculate po and pe between raters (9).

$$p_o = \frac{TP+TN}{TP+TN+FP+FN} \qquad (6)$$

$$P_e = \text{Probability of Positive} + \text{Propobability of Negative} \ (7)$$

$$Propability\ of\ Positive = \frac{TP+FP}{TP+TN+FP+FN}\ x\ \frac{TP+FN}{TP+TN+FP+FN} \qquad (8)$$

$$Propability\ of\ Negative = \frac{FP+TP}{TP+TN+FP+FN}\ x\ \frac{FN+TN}{TP+TN+FP+FN} \quad (9)$$


input_1 | input: | [(None, 90, 90, 3)]
InputLayer | output: | [(None, 90, 90, 3)]
conv2d_8 | input: | (None, 90, 90, 3)
Conv2D | output: | (None, 88, 88, 16)
max_pooling2d_8 | input: | (None, 88, 88, 16)
MaxPooling2D | output: | (None, 44, 44, 16)
conv2d_9 | input: | (None, 44, 44, 16)
Conv2D | output: | (None, 42, 42, 32)
max_pooling2d_9 | input: | (None, 42, 42, 32)
MaxPooling2D | output: | (None, 21, 21, 32)
global_average_pooling2d | input: | (None, 21, 21, 32)
GlobalAveragePooling2D | output: | (None, 32)
dense_4 | input: | (None, 32)
Dense | output: | (None, 1)
conv2d_4_input | input: | [(None, 90, 90, 3)]
InputLayer | output: | [(None, 90, 90, 3)]
conv2d_4 | input: | (None, 90, 90, 3)
Conv2D | output: | (None, 88, 88, 32)
max_pooling2d_4 | input: | (None, 88, 88, 32)
MaxPooling2D | output: | (None, 44, 44, 32)
conv2d_5 | input: | (None, 44, 44, 32)
Conv2D | output: | (None, 42, 42, 64)
max_pooling2d_5 | input: | (None, 42, 42, 64)
MaxPooling2D | output: | (None, 21, 21, 64)
conv2d_6 | input: | (None, 21, 21, 64)
Conv2D | output: | (None, 19, 19, 128)
max_pooling2d_6 | input: | (None, 19, 19, 128)
MaxPooling2D | output: | (None, 9, 9, 128)
conv2d_7 | input: | (None, 9, 9, 128)
Conv2D | output: | (None, 7, 7, 128)
max_pooling2d_7 | input: | (None, 7, 7, 128)
MaxPooling2D | output: | (None, 3, 3, 128)
flatten_1 | input: | (None, 3, 3, 128)
Flatten | output: | (None, 1152)
dropout_1 | input: | (None, 1152)
Dropout | output: | (None, 1152)
dense_2 | input: | (None, 1152)
Dense | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 1)
input_2 | input: | [(None, 90, 90, 3)]
InputLayer | output: | [(None, 90, 90, 3)]
conv2d_10 | input: | (None, 90, 90, 3)
Conv2D | output: | (None, 88, 88, 16)
conv2d_11 | input: | (None, 88, 88, 16)
Conv2D | output: | (None, 86, 86, 16)
max_pooling2d_10 | input: | (None, 86, 86, 16)
MaxPooling2D | output: | (None, 43, 43, 16)
conv2d_12 | input: | (None, 43, 43, 16)
Conv2D | output: | (None, 41, 41, 32)
conv2d_13 | input: | (None, 41, 41, 32)
Conv2D | output: | (None, 39, 39, 32)
max_pooling2d_11 | input: | (None, 39, 39, 32)
MaxPooling2D | output: | (None, 19, 19, 32)
conv2d_14 | input: | (None, 19, 19, 32)
Conv2D | output: | (None, 17, 17, 64)
conv2d_15 | input: | (None, 17, 17, 64)
Conv2D | output: | (None, 15, 15, 64)
max_pooling2d_12 | input: | (None, 15, 15, 64)
MaxPooling2D | output: | (None, 7, 7, 64)
conv2d_16 | input: | (None, 7, 7, 64)
Conv2D | output: | (None, 5, 5, 128)
conv2d_17 | input: | (None, 5, 5, 128)
Conv2D | output: | (None, 3, 3, 128)
max_pooling2d_13 | input: | (None, 3, 3, 128)
MaxPooling2D | output: | (None, 1, 1, 128)
global_average_pooling2d_1 | input: | (None, 1, 1, 128)
GlobalAveragePooling2D | output: | (None, 128)
dense_5 | input: | (None, 128)
Dense | output: | (None, 1)
input_3 | input: | [(None, 90, 90, 3)]
InputLayer | output: | [(None, 90, 90, 3)]
conv2d_18 | input: | (None, 90, 90, 3)
Conv2D | output: | (None, 88, 88, 16)
conv2d_19 | input: | (None, 88, 88, 16)
Conv2D | output: | (None, 86, 86, 16)
conv2d_20 | input: | (None, 86, 86, 16)
Conv2D | output: | (None, 84, 84, 16)
max_pooling2d_14 | input: | (None, 84, 84, 16)
MaxPooling2D | output: | (None, 42, 42, 16)
conv2d_21 | input: | (None, 42, 42, 16)
Conv2D | output: | (None, 40, 40, 32)
conv2d_22 | input: | (None, 40, 40, 32)
Conv2D | output: | (None, 38, 38, 32)
conv2d_23 | input: | (None, 38, 38, 32)
Conv2D | output: | (None, 36, 36, 32)
max_pooling2d_15 | input: | (None, 36, 36, 32)
MaxPooling2D | output: | (None, 18, 18, 32)
conv2d_24 | input: | (None, 18, 18, 32)
Conv2D | output: | (None, 16, 16, 64)
conv2d_25 | input: | (None, 16, 16, 64)
Conv2D | output: | (None, 14, 14, 64)
max_pooling2d_16 | input: | (None, 14, 14, 64)
MaxPooling2D | output: | (None, 7, 7, 64)
conv2d_26 | input: | (None, 7, 7, 64)
Conv2D | output: | (None, 5, 5, 128)
conv2d_27 | input: | (None, 5, 5, 128)
Conv2D | output: | (None, 3, 3, 128)
max_pooling2d_17 | input: | (None, 3, 3, 128)
MaxPooling2D | output: | (None, 1, 1, 128)
global_average_pooling2d_2 | input: | (None, 1, 1, 128)
GlobalAveragePooling2D | output: | (None, 128)
dense_6 | input: | (None, 128)
Dense | output: | (None, 1)

Model 1.
2 Convolutional Layer

Model 2.
4 Convolutional Layer

Model 3.
8 Convolutional Layer

Model 4.
10 Convolutional Layer

Fig. 3. Visualization for each CNN mode

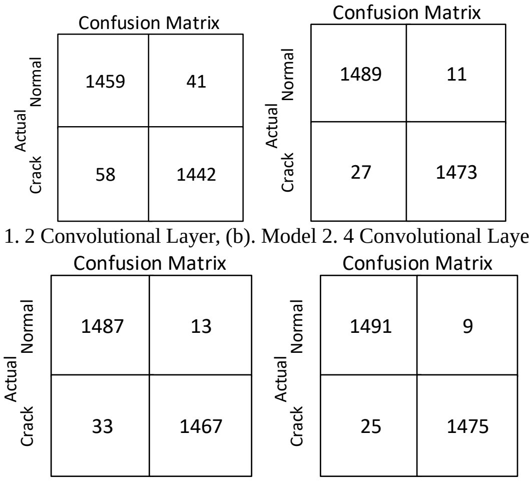
Fig. 3 shows the architectural layout. CNN1 is the basis of the CNN method. This study employed four variations of the CNN architecture to determine the best architecture for detecting the presence or absence of cracks in ancient building structures in Semarang's old town. For the computation this study used GPU GTX 1650 RAM 2 x 8 GB 2400 MHz DDR4.

This basic design can be extended to create CNN2 or CNN3 architectures, as shown in the CNN1 architecture diagram in Fig. 3. CNN3 is extensible to CNN4. CNN architectures are built to determine the impact of CONV layers and their activation functions on classification accuracy. In addition, comparisons can be made between CNN1, CNN2, CNN3, and CNN4 or between design groups to determine if the most appropriate type of design is being used for classifying old building crack images.

IV. Experiment and Discussion

Fig. 4(a) shows Model 1 with two convolution layers correctly predicts 1459 (TP) normal and 1442 (TN) cracked images, while 41 (FN) normal images are predicted to be cracked and 58 (FP) cracked images are predicted to be normal

Fig. 4(b) shows Model 2 with four convolution layers correctly predicts 1489 (TP) normal and 1473 (TN) cracked images, while 11 (FN) normal images are predicted to be cracked and 27 (FP) cracked images are predicted to be normal. Fig. 4(c) shows Model 3 with 8 convolution layers correctly predicts 1487 (TP) normal and 1467 (TN) cracked images, as well as 13 (FN) normal images that are cracked and 33 (FP) cracked images that are normal. Fig. 4(d) shows Model 4 with 10 convolution layers correctly predicts 1491 (TP) normal and 1475 (TN) crack images, as well as 9 (FN) normal images that are cracked and 25 (FP) cracked images that are normal. The FN on Model 1.d is smaller than on the other three models.



(a) Model 1. 2 Convolutional Layer, (b). Model 2. 4 Convolutional Layer

(c)Model 3. 8 Convolutional Layer, (d). Model 4. 10 Convolutional Layer

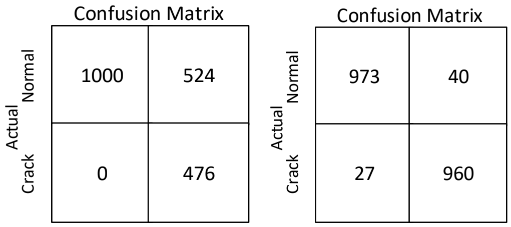
Fig. 4. Confusion matrix results for each CNN modification model

By identifying knowing TP, TN, FN, FP, performance values it generates such as accuracy, precision, recall, F1, MCC and CK values as shown in Table I, Table II and Table III. Table I shows the performance of Fig. 4 Confusion Matrix Results for each CNN modification model

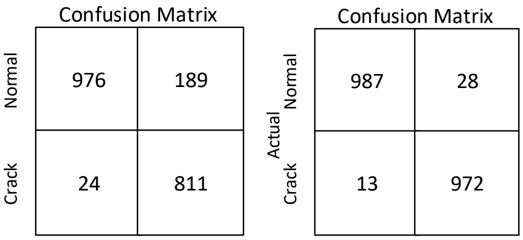
TABLE I. PERFORMANCE COMPARISON OF DIFFERENT CNN MODIFICATION METHODS

N_LAYER CONVOLUTIONAL	ACCURACY SCORE	RECALL SCORE	PRECISION SCORE	F1 SCORE	MCC (%)	CK (%)
2 LAYER	96.70	97.23	96.13	96.68	93.41	96.69
4 LAYER	98.73	99.26	98.20	98.73	97.47	98.73
8 LAYER	98.43	98.34	98.53	98.43	96.94	98.46
10 LAYER	98.87	99.40	98.33	98.86	97.74	98.86

Table I displays that the convolution layer design with 10 convolution layers has the highest F1 score of 98.86% and 98.87% accuracy among other models, as well as 98.33% precision and 99.40% recall. The Deca Convolutiol Layer Neural Network (DCL-NN) outperforms the other models in the popular statistical tests MCC and CK (Cohen's Kappa), with values of 97.74% and 98.86%.



(a). KNeighbors (n_neighbors=3) (b). Random Forest



(c) SVM (d) ExtraTrees (n_estimators=10)

Fig. 5. Confusion matrix results for each machine learning model

In Fig. 5(a), the KNN model correctly predicts 1000 (TP) normal and 476 (TN) crack images, while 524 (FN) normal images are predicted as cracked and 0 (FP) cracked images are predicted as normal. In Fig. 5(b), the Random Forest model correctly predicts 973 (TP) normal and 960 (TN) cracked images, with 40 (FN) normal images predicted as cracked and 27 (FP) cracked images predicted as normal. In Fig. 5(c), the SVM model correctly predicts 976 (TP) normal and 811 (TN) cracked images, with 189 (FN) normal images predicted to be cracked and 13 (FP) cracked images predicted to be normal. In Fig. 5(d), the Extra Tree model correctly predicts 987 (TP) normal and 972 (TN) cracked images, with 28 (FN) normal images predicted as cracked and 13 (FP) cracked images predicted as normal. When compared to the other three machine learning models, the Extra Trees model has a lower FN. Table II shows the performance of Fig. 5. Confusion Matrix Results for each machine learning model

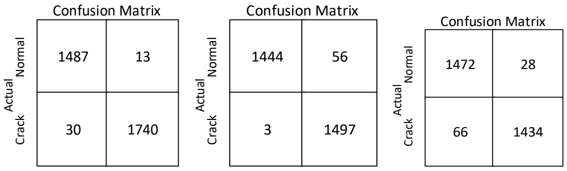
TABLE II. PERFORMANCE COMPARISON OF DIFFERENT MACHINE LEARNING METHODS

Machine Learning	Accuracy Score	Recall Score	Precision Score	F1 Score	MCC (%)	CK (%)
K Neighbors (n_neighbors=3)	74	83	74	72	55.89	69.85
Random Forest	97	97	97	97	93.31	96.64
SVM	89	90	89	90	79.79	88.89
ExtraTrees (n_estimators=10)	98	98	98	98	95.91	97.94

According to Table II, the ExtraTrees machine learning model has the highest F1 score of 98% and 98% accuracy among other models, as well as 98% precision and 98% recall. In the case of popular statistical tests such as MCC and CK (Cohen's Kappa), the ExtraTrees machine learning model outperforms the other models with values of 95.91% and 97.94%, respectively.

Fig. 6(a) illustrates the transfer learning model VGG19 correctly predicts 1487 (TP) normal and 1470 (TN) cracked images, with 13 (FN) normal images predicted to be cracked and 30 (FP) cracked images predicted to be normal. As for the Fig. 6(b) shows the Xception model correctly predicts 1444 (TP) normal and 1497 (TN) cracked images, with 56 (FN) normal images predicted as cracked and 3 (FP) cracked images predicted as normal. Fig. 6(c) illustrates the MobileNet model correctly predicts 1472 (TP) normal and 1434 (TN) cracked images, with 28 (FN) normal images predicted as cracked and

66 (FP) cracked images predicted as normal. The VGG19 model has a lower FN than the other three transfer learning models. The performance of Fig. 6, Matric Confusion Results for each model of Pre-Trained CNN is shown in Table III.



(a) VGG 19 (b) XCeption (c) Mobile Net

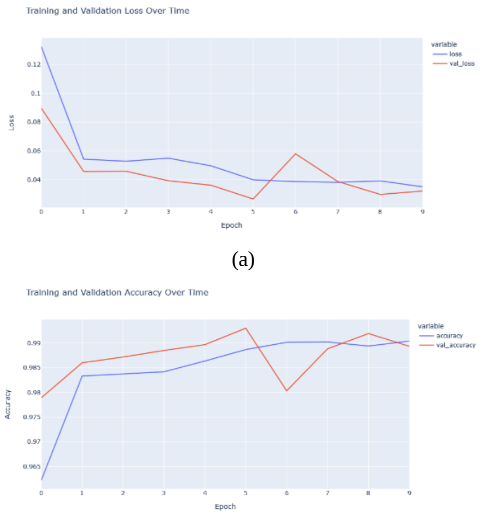
Fig. 6. Matric confusion results for each model of pre-trained CNN

TABLE III. PERFORMANCE COMPARISON OF DIFFERENT TRANSFER LEARNING METHODS

Transfer Learning	Accuracy Score	Recall Score	Precision Score	F1_Score	MCC (%)	CK (%)
VGG19	98.57	99.12	98.00	98.56	97.14	98.56
Xception	98.03	96.39	99.80	98.07	96.13	98.05
MobileNet	96.87	98.08	95.60	96.83	93.76	96.85

Table III shows that the VGG19 model transfer learning design had the highest F1 score of 98.56% and 98.57% accuracy, as well as 98.00% precision and 99.12% recall among other models. The transfer learning performance of the VGG19 model outperforms the other models in popular statistical tests such as MCC and CK (Cohen's Kappa), with values of 97.14% and 98.56%, respectively.

In Model 1, after 10 iterations, the program automatically stops training, and the correct model rate is approximately 97.67%. (Loss 0.1141). As illustrated in Fig. 9, the maximum true rate is reached after the tenth iteration. It takes about 13 seconds. As illustrated in Fig. 7, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. In Model 2, after 10 iterations, the program automatically stops training, and the correct model rate is approximately 98.57% (0.0646).

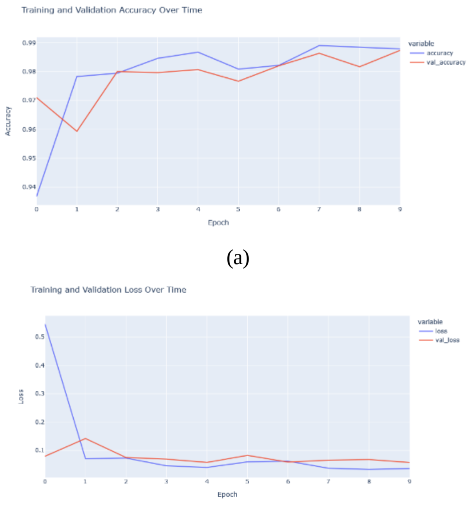


(a)

(b)

Fig. 7. Graph of accuracy and loss in model 1 with 2 convolutional layer

Fig. 8 shows that the maximum true rate is reached after the tenth iteration. It takes about 28 seconds. As shown in Fig. 8, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. In Model 2, after 10 iterations, the program automatically stops training, and the correct model rate is approximately 98.90% (Loss 0.0561).

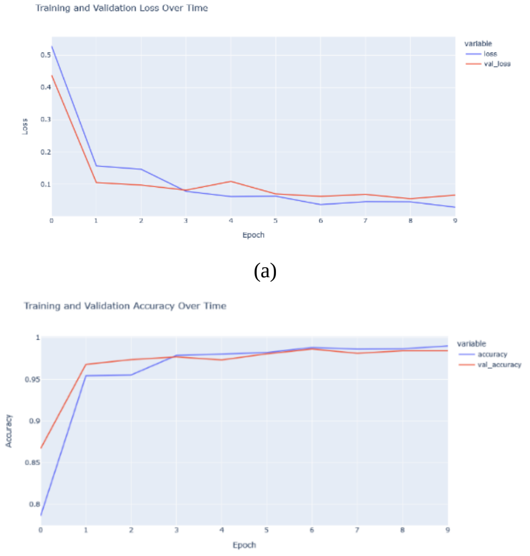


(a)

(b)

Fig. 8. Graph of accuracy and loss in model 2 with 4 convolutional layer

Figure 9 shows that the maximum true rate is reached after the tenth iteration. It takes about 31 seconds. As illustrated in Figure 9, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. In Model 4, after 10 iterations, the program automatically stops training, and the correct model rate is approximately 99.17%. (Loss 0.0284)

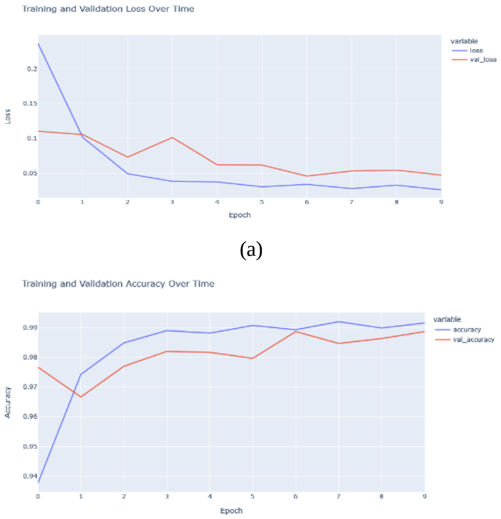


(a)

(b)

Fig. 9. Graph of accuracy and loss in model 2 with 8 convolutional layer

Fig. 12 shows that the maximum true rate is reached after the tenth iteration. It takes about 32 seconds. As illustrated in Fig. 12, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate.



(a)

(b)

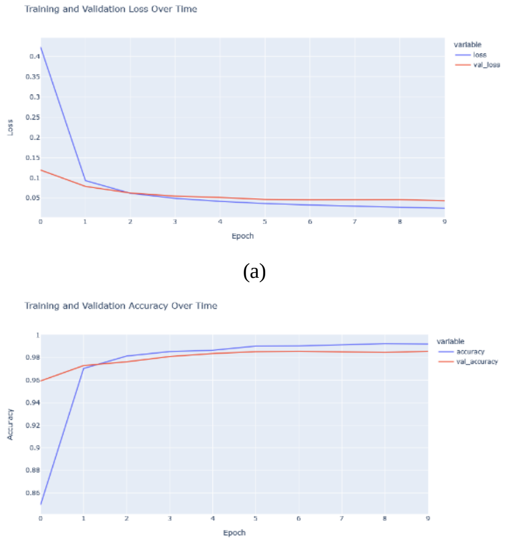
Fig. 10. Graph of accuracy and loss in model 2 with 10 convolutional layer

Fig. 10 shows that the maximum true rate is reached after the tenth iteration. It takes about 32 seconds. As illustrated in Fig. 10, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate.

The CNN model's performance is also compared to that of other transfer learning models, such as VGG19, Exception, and MobilNet. The following performance graphs are generated for each transfer learning model:

The VGG model automatically stops training after 10 iterations, and the correct model rate is approximately 98.93%. (Loss 0.0384). As shown in Fig. 13, the maximum true rate is reached after the tenth iteration. It takes approximately 61 seconds. As shown in Fig. 13, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. In Model Exception, the program automatically stops training after 10 iterations, and the correct model rate is approximately 98.67% (Loss 0.0526).

As shown in Fig. 11, the maximum true rate is reached after the tenth iteration. It takes approximately 61 seconds. As shown in Fig. 11, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. In Model Exception, the program automatically stops training after 10 iterations, and the correct model rate is approximately 98.67% (Loss 0.0526).

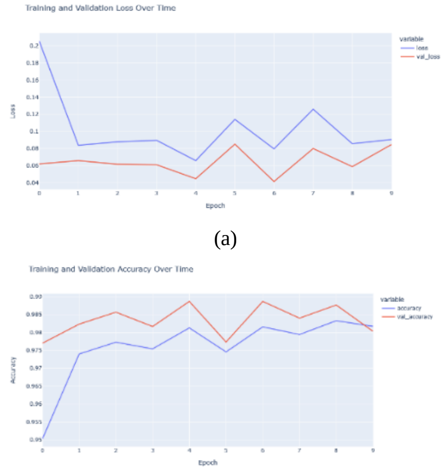


(a)

(b)

Fig. 11. VGG19 model accuracy and loss graphs

As illustrated in Fig. 12, the maximum true rate is reached after the tenth iteration. It takes about 63 seconds. As illustrated in Fig. 12, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate. The MobileNet model, after 10 iterations, the program automatically stops training, and the correct model rate is approximately 96.67% (Loss 0.0993).

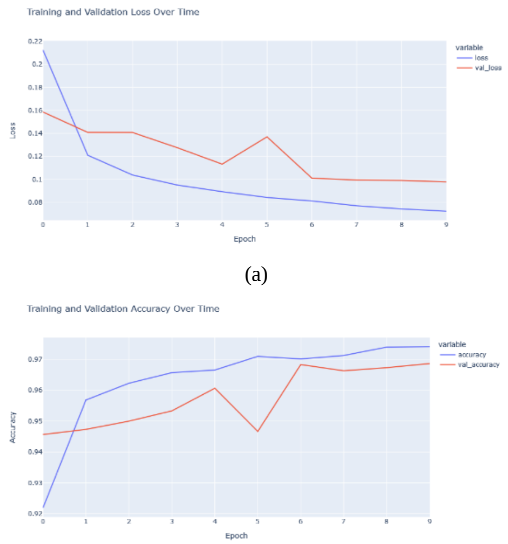


(a)

(b)

Fig. 12. Graph of accuracy and loss in exception model

As shown in Fig. 13, the maximum true rate is reached after the tenth iteration. It takes approximately 30 seconds. As shown in Fig. 13, the correct rate gradually stabilizes in later training stages due to the continuous reduction of the learning rate.



(a)

(b)

Fig. 13. Accuracy and loss graphs on the MobileNet mode

Researchers in this study conducted research that was also compared to other researchers using the same dataset. Zhang et al. [20] classified cracks using CNN and four convolution layers, two of which were fully connected. ConvNets is the name of the method used. Precision and recall rates are 86.96% and 92.51%, respectively. In classifying aligned images, Fang et al. [21] employ three convolution layers and three Fully Connected Layers. Precision and recall rates for experiments using the same data as the researchers were 18.4% and 94.3%, respectively. While the researchers used 10 convolution layers and one Fully Connected layer to achieve a recall and precision of 99.40% and 98.33%, respectively. Our model outperforms both works [1] and [2], as shown in the Table IV below.

TABLE IV. COMPARISON OF PERFORMA WITH OTHER RESEARCHERS

Other Researchers	Number of Convolution Layers	Number of Fully Connected Layers	Precision	Recall
Zhang et al. [20]	4	2	86.96%	92.51%
Fang et al. [21]	3	2	18.40%	94.30%
Proposed Model	10	1	98.33%	99.40%

Each neuron in the convolution layer needs to be transformed into one-dimensional data. First before it can be included in a fully-connected layer. Also, because it causes the data to lose its spatial information and is not reversible, while the fully connected layer can only be implemented at the end of the network CNN [22].

Prior researchers used 2 Fully Connected Layers (FCL) to prevent loss of image spatial information and a long duration so that classification performance can be maintained, but with this study, we used a large number of convolution layers and a low number of FCL, to produce high performance.

V. CONCLUSION

This research contributes to crack classification at historical building by using four architectural design variants of convolutional neural networks (CNNs), namely, CNN1-CNN4. The research methodology used in this study is an experimental cracking classification using four variations of convolutional neural network (CNN) architecture design (CNN1 - CNN4).

Experimental results show that CNN4 (Deca Convolutional Layer Neural Network/DCL-NN) provides the best classification results for concrete cracks in old buildings compared to other architectural designs tested. The DCL-NN architecture has an accuracy of 98.87%, precision of 99.40%, recall of 98.33%, F1 score of 98.86%, MCC of 97.74%, and CK of 98.86%.

In addition, results from comparisons with pre-trained CNN algorithm methods such as VGG19, Xception, MobileNet, and machine algorithms such as KNeighbors (n neighbors=3), Random Forests, Support Vector Machines (SVM), ExtraTrees (n estimators) Classification using learning. = 10), he shows the superiority of DCL-NN in classifying concrete cracks in old buildings in Semarang city.

ACKNOWLEDGMENT

This research supported by Research grant 2022 (Research Publication International /RPI) from Universitas Diponegoro SK. Rector Universitas Diponegoro No: 215 /UN7.A/HK/VII/2022; No.SPK:569 108/UN7.D2/PP/VII/2022.

REFERENCES

[1] h. werdiningsih s. r. sari, a. r. harani, “Pelestarian dan Pengembangan Kawasan Old Town sebagai landasan budaya kota Semarang,” Modul, vol. 17, no. 1, pp. 49–55, 2017.

[2] H. N. Agastya Grahadwiswara, Zaenal Hidayat, “Pengelolaan Kawasan Old town Semarang Sebagai Salah Satu Kawasan Pariwisata Di Kota Semarang,” J. Public Policy Manag. Rev., vol. 3, no. 4, 2014.

[3] D. G. Aggelis, N. Alver, and H. K. Chai, “Health monitoring of civil infrastructure and materials,” Sci. World J., vol. 2014, 2014, doi: 10.1155/2014/435238.

[4] M. Gavilán et al., “Adaptive road crack detection system by pavement classification,” Sensors, vol. 11, no. 10, pp. 9628–9657, 2011, doi: 10.3390/s111009628.

[5] Y. C. Tsai, V. Kaul, and R. M. Mersereau, “Critical assessment of pavement distress segmentation methods,” J. Transp. Eng., vol. 136, no. 1, pp. 11–19, 2010, doi: 10.1061/(ASCE)TE.1943-5436.0000051.

[6] A. Hafid and D. S. Rahayu, “Pengukuran Kedalaman Retak pada Beton Menggunakan UPVT,” Sigma Epsil., vol. 24, no. 1, pp. 1–8, 2020.

[7] P. Wang, Y. Hu, Y. Dai, and M. Tian, “Asphalt Pavement Pothole Detection and Segmentation Based on Wavelet Energy Field,” Math. Probl. Eng., vol. 2017, 2017, doi: 10.1155/2017/1604130.

[8] T. Yamaguchi, S. Nakamura, R. Saegusa, and S. Hashimoto, “Image-based crack detection for real concrete surfaces,” IEEJ Trans. Electr. Electron. Eng., vol. 3, no. 1, pp. 128–135, 2008, doi: 10.1002/tee.20244.

[9] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[10] J. Zhu and J. Song, “An intelligent classification model for surface defects on cement concrete bridges,” Appl. Sci., vol. 10, no. 3, 2020, doi: 10.3390/app10030972.

[11] M. A. Moni and F. Hasan, “CNN Based on Transfer Learning Models Using Data Augmentation and CNN Based on Transfer Learning Models Using Data Augmentation and Transformation for Detection of Concrete Crack,” Algorithms, no. August, pp. 2–17, 2022, doi: 10.3390/a15080287.

[12] Z. Y. J. Zhang L., Yang F., Zhang Y.D., "Performance Evaluation of Different Algorithms for Crack Detection in Concrete Structures," in Proceedings of the International Conference on Image Processing, ICIP, 2016, pp. 6–11.

[13] W. Wang and C. Su, "Automation in Construction Semi-supervised semantic segmentation network for surface crack detection," Autom. Constr., vol. 128, no. April, p. 103786, 2021, doi: 10.1016/j.autcon.2021.103786.

[14] Z. Fan, C. Li, Y. Chen, P. Di Mascio, X. Chen, and G. Zhu, "Ensemble of Deep Convolutional Neural Networks for Automatic Pavement Crack Detection and Measurement," Coatings, pp. 1–15, 2020, doi: 10.3390/coatings10020152.

[15] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, and X. Chen, "applied sciences Automatic Bridge Crack Detection Using a Convolutional Neural Network," Appl. Sci., pp. 1–14, 2019.

[16] A. K. Nugroho and M. H. Purnomo, "Quad Convolutional Layers (QCL) CNN Approach for Classification of Brain Stroke in Diffusion Weighted (DW) - Magnetic Resonance Images (MRI)," Int. J. Intell. Eng. Syst., vol. 15, no. 1, pp. 414–427, 2022, doi: 10.22266/ijies2022.0228.38.

[17] A. K. Nugroho, "Utilizing the Hepta Convolutional Layer Neural Network (HCL-NN) Based on a Multi Optimizer for the Classification of Brain Stroke MRI," Int. J. Intell. Eng. Syst., vol. 15, no. 3, pp. 304–318, 2022, doi: 10.22266/ijies2022.0630.26.

[18] D. Chicco, M. J. Warrens, and G. Jurman, "The Matthews correlation coefficient (MCC) is more informative than Cohen ' s Kappa and Brier score in binary classification assessment," IEEE Acces, vol. X, pp. 1–15, 2021, doi: 10.1109/ACCESS.2021.3084050.

[19] N. Hasdyna, "Analisis Matthew Correlation Coefficient pada K-Nearest Neighbor dalam Klasifikasi Ikan Hias," Inform. J., vol. 5, no. 2, pp. 57–64, 2020.

[20] Y. J. Z. Lei Zhang , Fan Yang , Yimin Daniel Zhang, "ROAD CRACK DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK," in 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3708–3712, doi: 10.1109/ICIP.2016.7533052.

[21] F. Fang, L. Li, Y. Gu, H. Zhu, and J. Lim, "A novel hybrid approach for crack detection," Pattern Recognit., vol. 107, 2020, doi: 10.1016/j.patcog.2020.107474.

[22] R, O.aditya yanuar (2018) Fully-connected layer CNN Dan Implementasinya, Universitas Gadjah Mada. Available at: https://machinelearning.mipa.ugm.ac.id/2018/06/25/fully-connected-layer-cnn-dan-implementasinya/ (Accessed: January 21, 2023).