applied sciences

MDPI

Article

FAECCD-CNet: Fast Automotive Engine Components Crack Detection and Classification Using ConvNet on Images

Michael Abebe Berwo, Yong Fang *, Jabar Mahmood, Nan Yang, Zhijie Liu and Yimeng Li

School of Information Engineering, Chang'an University, Xi'an 710064, China
* Correspondence: fy@chd.edu.cn

Abstract: Crack inspections of automotive engine components are usually conducted manually; this is often tedious, with a high degree of subjectivity and cost. Therefore, establishing a robust and efficient method will improve the accuracy and minimize the subjectivity of the inspection. This paper presents a robust approach towards crack classification, using transfer learning and fine-tuning to train a pre-trained ConvNet model. Two deep convolutional neural network (DCNN) approaches to training a crack classifier—namely, via (1) a Light ConvNet architecture from scratch, and (2) fined-tuned and transfer learning top layers of the ConvNet architectures of AlexNet, InceptionV3, and MobileNet—are investigated. Data augmentation was utilized to minimize over-fitting caused by an imbalanced and inadequate training sample. Data augmentation improved the accuracy index by 4%, 5%, 7%, and 4%, respectively, for the proposed four approaches. The transfer learning and fine-tuning approach achieved better recall and precision scores. The transfer learning approach using the fine-tuned features of MobileNet attained better classification accuracy and is thus proposed for the training of crack classifiers. Moreover, we employed an up-to-date YOLOv5s object detector with transfer learning to detect the crack region. We obtained a mean average precision (mAP) of 91.20% on the validation set, indicating that the model effectively distinguished diverse engine part cracks.

Keywords: deep learning; crack classification and detection; transfer learning; fine-tuning; data augmentation; YOLOv5s object detector

Citation: Berwo, M.A.; Fang, Y.; Mahmood, J.; Yang, N.; Liu, Z.; Li, Y. FAECCD-CNet: Fast Automotive Engine Components Crack Detection and Classification Using ConvNet on Images. *Appl. Sci.* **2022**, *12*, 9713. https://doi.org/10.3390/app12199713

Academic Editor: Sungho Kim

Received: 8 August 2022
Accepted: 19 September 2022
Published: 27 September 2022

1. Introduction

Automotive component cracking is a common phenomenon in the mechanical components of an engine and has a significant impact on the structural firmness of engine parts, engine fuel consumption, and environmental pollution. Cracking is one of the signs of deficits in engine components, which can have many cases, such as mechanical stresses, wear mechanisms, overheating, and lubrication mechanisms. Manual crack checking and the inspection of automotive engine components is performed on a regular basis by trained and experienced machinists and mechanics who use a spray, a magnifying glass, and a magnetic particle inspection method [1–3], and this process is labor-intensive and time-consuming. Moreover, inspection outcomes rely immensely on human bias and subjective judgment, leading to inaccuracy and in some faults being overlooked.

In the past couple of years, several image-based, automatic, or semiautomatic crack detection and classification approaches using traditional digital image processing approaches have been proposed, namely, thresholding, segmentation-based, and edge-detection-based approaches. Berwo et al. [4] proposed an automotive engine cylinder head crack detection method based on Canny edge detection with morphological dilation that employs pre-processing, de-noising using Gaussian filtering, calculation of the intensity gradient, double thresholding, non-maximal suppression, and morphological dilation. Generally, traditional crack detection approaches follow image graying, binarization, edge extraction, and morphological operations. Though traditional crack detection approaches are more effective than manual checking or inspections, they are incredibly dependent on models

and processing flows. This dependence leads to poor generalization due to noise formed by structural dynamics, carbon deposits, and other factors related to the life of parts.

ConvNets have recently been proposed for object detection and image classification in vision-based systems. Some of the ConvNet architectures, namely, VGGNet [5], InceptionV3 [6], AlexNet [7], ResNets [8], and GoogleNet [9], were typically trained on high-volume datasets, namely, GoogleNet and ImageNet, and have successfully been used to obtain state-of-the-art outcomes. Many recent works have established ConvNet-based algorithms for the purposes of automotive engine precision part defect detection [10], highway crack classification and detection [11], weld defect detection and image defect recognition [12], weld image on-line defect detection [13], andautomotive paint defect detection [14].

The YOLOv5 architecture [15] utilizes the entire image instantaneously for detection. Moreover, it is considerably faster than the R-CNN series of the framework. It has high precision, high speed, and fewer model parameters than the other object detector algorithms.

Previous works have recommended original ConvNets constructed from scratch and trained to localize or sort defects or cracks in images. Training a new network is typically a costly practice that frequently checks and examines various ConvNet algorithms and tunes several parameters to optimize performance. Thus, transfer learning (TL) of a pre-trained deep learning (DL) architecture on standard datasets for new baseline classification tasks has been implemented. Ref. [16] proposed an image-based approach using TL to detect concrete cracks, Ref. [17] proposed a DL-based transfer learning approach to detect cracks in ceramic plates, Ref. [18] proposed a DL-based approach to detect defects in tire X-ray images, and so on.

in this work, we first developed a ConvNet architecture-based approach to identify and classify cracks using images of automotive engine components that are among the most susceptible to cracks in their engine parts. The datasets were collected and preprocessed. A ConvNet-based model was assembled from scratch to build the primary crack classifier. Following that, a pre-trained network on a high-volume dataset was employed to transfer the learned features that were useful for most vision-based tasks, including face detection, vehicle detection, and crack detection. To improve the classifier's robustness and classification accuracy, DA approaches were used to create extra datasets for model training. For better understandability, the abbreviations used are presented in Table 1.

Table 1. Abbreviation guide .

Abbreviation	Detail of Abbreviations
Adam	Adaptive Momentum
AP_*i*	Average Precision of Class *i*
BN	Batch Normalization
BS	Batch Size
CB	Convolutional Block
CC	Computational Complexity
CL	Classification Layer
ConvNets	Convolutional Neural Networks
DA	Data Augmentation
DCNNs	Deep Convolutional Neural Networks
DL	Deep Learning
DNN	Deep Neural Network
DNNs	Deep Neural Networks
FC	Fully Connected
FF	Fuzzy Filter
FN	False Negative
FP	False Positive
FPR	False Positive Rate

Table 1. *Cont.*

Abbreviation	Detail of Abbreviations
FT	Fine-tuning
GAP	Global Average Pooling
GS	Grid Search
HOG	Histogram of Oriented Gradient
LF	Loss Function
LR	Learning Rate
mAP	mean Average Precision
N	Number of Classes
NW	Number of Weights
PL	Pooling Layer
RCNNs	Regional Convolutional Neural Networks
TL	Transfer Learning
TN	True Negative
TP	True Positive
TPR	True Positive Rate

1.1. Motivations

In this paper, we present a robust crack classifier, and detector architectures are established to detect and classify cracks in engine component images using a DCNN architecture. Standard vision-based approaches to training a crack classifier architecture include (1) constructing a light convNet architecture (LAECNet) from scratch; (2) the FT, TL, and DA techniques, using three pre-trained ConvNet architectures (i.e., AlexNet, InceptionV3, and MobileNet); and (3) a YOLOv5s object detector algorithm with TL. These were utilized in this study.

1.2. Contributions

In this section, we discuss the contributions of this work as follows.

- Several deep CNN object classification and detection architectures with numerous hyper-parameter optimization, fine-tuning, transfer learning, and DA techniques were evaluated to find the optimal solution for crack detection and classification for automotive engine components.
- Deep learning-based architectures are sensitive to input image noise, which affects the detection power. Therefore, various preprocessing and fuzzy filter (FF) approaches were employed to handle the lighting effects and noise in the engine component images.
- The proposed framework exhibited relatively good automotive engine crack detection performance on custom data sets.
- Furthermore, to compute the computational cost and offer a better benchmark between accuracy, speed, and time of training, testing and detection approaches utilizing various image results were investigated.

1.3. Paper Organization

The rest of this paper is organized as follows. The related work is introduced in Section 2. In Section 3, we explain the suggested framework for an automotive engine component crack detection system with diverse detection architectures. In Section 4, we present the experimental results of our approaches and visualization methods. Section 5 concludes our paper.

2. Related Work

Nowadays, computer-vision-based crack detection approaches can be grouped into three categories: traditional, machine-learning-based, and deep-learning-based approaches.

Traditional approach: In recent works, a traditional approach to detecting cracks in various images, such as road images, concrete images, pavement, buildings, welded parts, and others, has been employed, using traditional techniques to extract crack features.

However, these techniques encounter problems related to decisions about optimized thresholding. In addition, they are sensitive to noise and imaging conditions, which ultimately results in poor performance and greater subjectivity. Some of the crack detection approaches based on traditional methods that have been employed to images include the Canny edge detector [19], the Gabor filter [20,21], wavelet transform [22,23], and histogram-oriented gradient (HOG) method [24,25], which show significant improvements in detecting simple cracks, but they are not suitable for diverse and complex cracks, and are laborious and time-consuming.

Machine-learning-based approach: With the rapid growth of machine learning and vision-based technologies, several approaches have been applied to crack detection tasks. These include support vector machines (SVMs) [26–28], AdaBoost [29], and so on. However, these detection approaches are restricted to detecting learned cracks, making it hard to detect new cracks, and making it challenging to design universal features that can be employed on all types of cracks.

Deep-learning-based approach: Recently, researchers have successfully used deep learning-based methods in various crack detection applications, such as road crack detection [30], weld part defect detection [31,32], concrete crack detection [33], and so on. These techniques have significant merits over the traditional and machine-learning-based approaches in that they can easily detect various cracks, including complex and diverse cracks; minimize subjectivity; and easily design and employ all types of cracks. However, deep-learning-based crack detection approaches require enormous resources and time to train the model. Therefore, the techniques of TL and FT have been employed to solve the problem of resource usage and cost in the training of a ConvNet architecture from scratch, using pre-trained deep architectures trained on massive ImageNet datasets. Researchers have employed the DCNNs in various applications, such as Alexnet [34,35], InceptionV3 [36], and MobileNet [37].

3. Proposed Methodology

In this section, we present the proposed work methodology and the frameworks of the AlexNet, InceptionV3, LAECNet, and MobileNet architectures for classification, as well as the modified YOLOv5s technique for detecting cracks in engine component images.

3.1. Methodology

In this study we constructed, implemented, compared, and chose the best architecture for detecting and classifying cracks in automotive engine components. This task was achieved by training the dataset on the following ConvNet architectures: LAECNet training from scratch, AlexNet, InceptionV3, MobileNet, and the YOLOv5s model. Figure 1, displays the workflow of the recommended strategy.

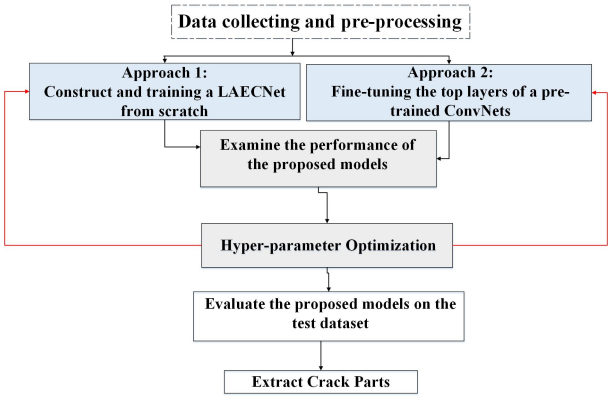

Figure 1. Workflow of the recommended strategy.

3.1.1. Approach 1

Approach 1 (LAECNet) consisted of constructing and training a crack classifier with a light ConvNet-based framework from scratch.The term "light convolutional neural network" refers to a ConvNet, which had few weight layers compared to very deep ConvNets. The light ConvNet contained two (2) convolutional blocks (conv2D1 and Conv2D) and one FC layer, as shown in Figure 2. Every convolutional block of the network consisted of a pooling layer (PL), a linear rectifier unit (ReLu), and a convolutional layer (CL).

The input image for LAECNet had dimensions of 227 × 227 × 3, and was classified as either a 'non-crack' or a 'crack' image. A CL performed a convolutional task on the output of the previous layers, using a group of filters to acquire the features that are significant for distinguishing a 'non-crack' image from a 'crack' image. Deeper CLs extract more global features, compared to light layers, which learn the local features that are essential for crack image classification.

Activation layers were employed to introduce

non-linearity to the primary net. The linear rectifier unit was employed due to its faster computation speed and its better classification accuracy compared to the convolutional sigmoidal function [38,39]. The flattened output of the second CL developed the input of the fully connected layer, which comprised 32 units with linear rectifier unit activation [40]. The output layer comprised a single unit, and L2 regularization [41,42] was performed to avoid over-fitting,

which is suitable for binary classification tasks, as shown in Equation (1). The squared hinge loss function (LF) was utilized for the 'maximum margin'

binary classification problem to compile the proposed LAECNet architecture and make the model robust [43,44]. Mathematically, it is defined as in Equation (2).

$$L_2 = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} {w_j}^2 \quad (1)$$

Here, λ was called the "tuning parameter", which enabled us to decide how heavily we wanted to penalize the flexibility of our model.

$$l(y) = max(0, 1 - t.y) \quad (2)$$

Here, (y) is the prediction value and (t) is the actual target for prediction.

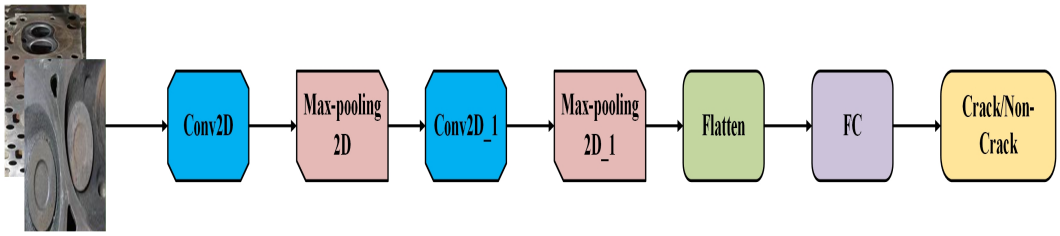

Figure 2. Framework of the LAECNet architecture.

3.1.2. Approach 2

We implemented the fine-tuning (FT) technique for the top layers of a pre-trained deep ConvNet with the AlexNet, InceptionV3, and MobileNet architectures.

AlexNet architecture: The AlexNet architecture, developed by Krichevsky et al. [7], was trained using millions of images to classify various categories in ImageNet. In the fine-tuning approach, the last convolutional block (CB) of the AlexNet model, as shown in Figure 3, is modified/fine-tuned with the fully connected top-level layer. This approach was accomplished by

(1) loading the model's weights, (2) freezing/halting the model up to the last CB, and training only the last layer/block, which prevented the fully connected layer from overfitting the model. This architecture comprised five (5) CLs and three (3) FC layers

with a sigmodal function. An Adam optimizer with a learning rate (LR) of 0.0001 and binary cross-entropy loss was chosen. The selected size of the training input image was $227 \times 227 \times 3$.

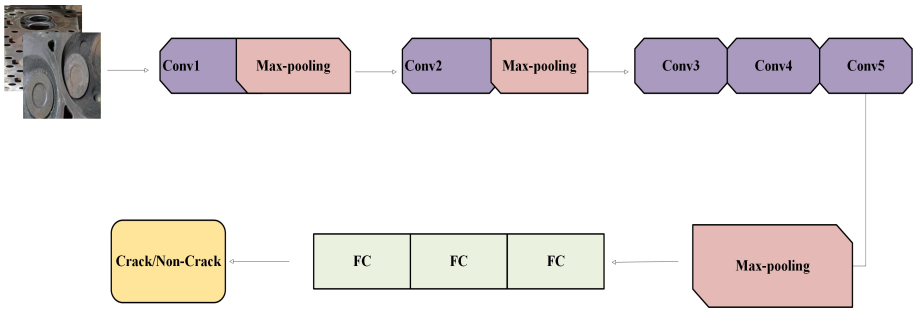

Figure 3. Framework of the AlexNet architecture.

InceptionV3 architecture: The InceptionV3 architecture is mostly used in image recognition on pre-defined ConvNets. The authors in [6] proposed an InceptionV3 architecture for flower classification. The FC layer of the architecture net was replaced with one 531 dense ReLu layer and a sigmodal classification layer. The FC layers were pre-trained on bottleneck features before being attached to the CLs, and training was performed on the final two (2) Inception blocks with an input image size of $299 \times 299 \times 3$ (Figure 4).

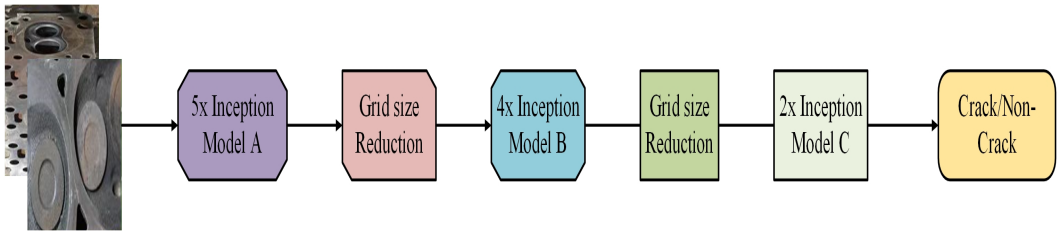

Figure 4. Framework of the InceptionV3 architecture.

MobileNet architecture: MobileNet was presented in [45], and was demonstrated to achieve a good balance between classification performance and computational cost. This architecture was designed and based on the use of depthwise separable convolutions to construct a shallow/deepnet that makes the architecture smalland minimizes the computation time. We removed three layers, including the average pooling, FC, and sigmoidal function. Next, three extra layers, the global average pooling (GAP) and the BN and sigmodal layers, were attached. This technique helps us to train the model faster and achieve better classification accuracy. Then, drop out and ReLu activation functions were proposed to prevent overfitting. The architecture was trained with an input image size of $224 \times 224 \times 3$. Figure 5 depicts the framework of the suggested MobileNet architecture.

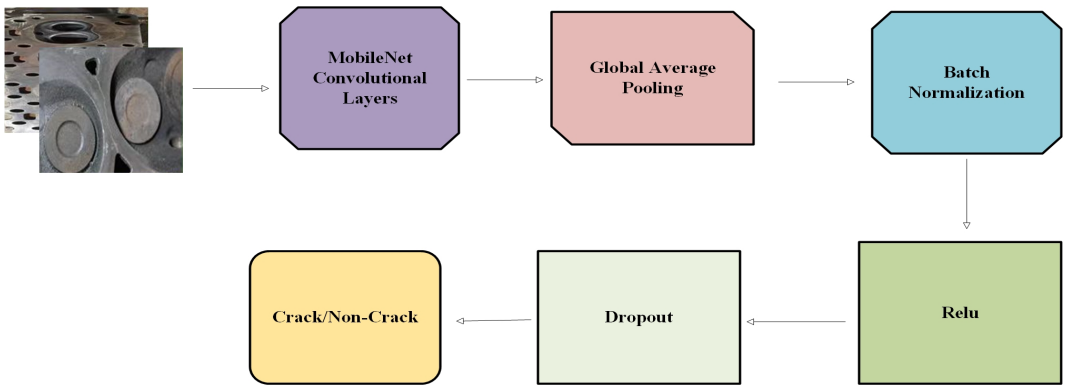

Figure 5. Framework of the MobileNet architecture.

3.1.3. Approach 3

YOLOv5 is an excellent object detection architecture that is part of the YOLO network series. It is an outcome of the ongoing combination and improvements that have been built upon the two previous series. YOLOv5 has obtained remarkable outcomes in COCO and PASCAL VOC object detection tasks. It has produced four net architectures: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The YOLOv5 architecture is split into 4 (four) categories: input, backbone, neck, and prediction. Like the focus structure, it uses the CSP network as a backbone, the PP block and PANet as necks, and GIoU_loss as a prediction (head). The innovative focus structure in the backbone of YOLOv5 is employed for dividing tasks.

In the YOLOv5 algorithm, an input image size of $3 \times 640 \times 640$ was fed into the model. Subsequently, in the focus slice task, the feature maps with $12 \times 320 \times 320$ were transformed, followed by the regular convolution operation of 32 convolution size kernels. Finally, it was transformed into a feature map size of $32 \times 320 \times 320$ with the CSP structure in the backbone network. Due to the advantages of this network, several studies have been conducted in various areas, such as in safety helmet detection [46], face mask detection [47], infrared image detection [48], the detection of mold on food surfaces [49], the detection of heavy-goods vehicles [50], and densely-populated traffic detection [51]. The architecture of YOLOv5s is depicted in Figure 6.

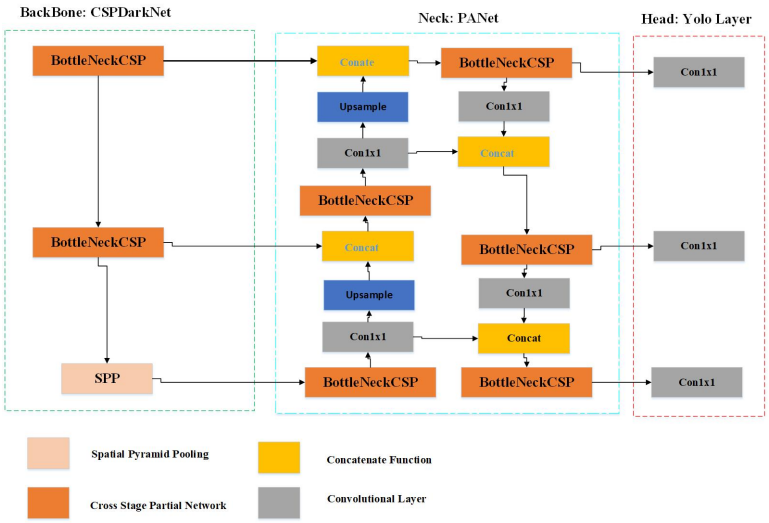

Figure 6. Framework of the YOLOv5s architecture.

3.2. Dataset Description

In Section 3.2, we discuss the experimental data and discuss the techniques used for data augmentation.

3.2.1. Experimental Dataset

In this paper, we collected 56 crack images and 85 non-crack images for automotive engine components from Google using web-scraping. Figure 7 represents some of the various images collected. The collected and the augmented images were acquired at high resolution. However, the input RGB image size for the proposed ConvNet architectures did not exceed 299×299, in order to speed up training and minimize the number of weights. As shown in Table 2, we designated different input sizes for each proposed ConvNet model. We also uses fuzzy filtering (FF) to eliminate both additive white Gaussian noise and salt and pepper noise (de-noising the images from noise caused by structure dynamics, oils,

dirt, and environmental conditions) [52–56]. The mathematical notation of fuzzy sets for an input image of $f(a, b)$ can be defined using Equation (3).

$$f(a, b) = \cup \frac{\mu_{ab}}{I_{ab}} ab \tag{3}$$

Here, $a = 1, 2, 3, ..., M$, $b = 1, 2, 3, ..., N$, I_{ab} is the intensity of the $(a, b)^{th}$ value, and μ_{ab} is the membership value.

We experimented on both real and augmented datasets with 80% for training, 10% for validation, and 10% for testing purposes.

Figure 7. Sample of automotive engine component images from Berwo et al. [4].

Table 2. Detailed parameters of the proposed models.

Models	Input Size	NW	Batch Size	Learning Rate
AlexNet	$227 \times 227 \times 3$	58.29×10^6	16	0.001
InceptionV3	$299 \times 299 \times 3$	6.83×10^6	16	0.001
LAECNet	$227 \times 227 \times 3$	5.71×10^6	16	0.001
MobileNet	$224 \times 224 \times 3$	4.73×10^6	16	0.001

3.2.2. Data Augmentation

In this study, we used data augmentation (DA) approaches, comprising a random rotation of 10 degrees, a zoom-range of 0.2, a shear-range of 0.2, a horizontal flip transformation, as well as scaling and brightness approaches aiming to augment the crack and non-crack samples. These were administered to avoid over-fitting and to improve the performance of the models [57].

3.3. Hyper-Parameter Optimization

During the design and implementation of deep learning architectures, exploring the hyper-parameter space using optimization approaches can be undertaken in order tosearch for the optimal hyper-parameters for the convNets [58–60]. A grid search (GS) is the most widely utilized approach to searching the configuration space in recent studies. It is used to calculate the predetermined values [58]. However, its computational cost rises exponential at a rate of $\mathcal{O}(n^k)$ [61] with k parameters. Consequently, it is the only effective hyper-parameter configuration space explorer for small distinct values (Table 3).

Table 3. Different hyper-parameters of the proposed algorithm.

Parameter	Without DA	With DA
Weight Decay	5×10^{-4}	5×10^{-4}
Momentum	0.90	0.90
Iterations per Epoch	7	38
Maximum Iterations	700	3800
Batch Size	16	16
Maximum Epochs	100	100

3.4. Performance Evaluation Metrics

The performance parameters applied for evaluating the effectiveness of the proposed scheme were *accuracy, recall, precision, F-measure, mAP*, and the classification rate (*Error Rate*).

$$Accuracy = \frac{TN + TP}{(FP + TP + FN + TN)} \tag{4}$$

$$Precision = \frac{TP}{(FP + TP)} \tag{5}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{6}$$

$$F - measure = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{7}$$

$$TPR = \frac{TP}{(TP + FN)} \tag{8}$$

$$FPR = \frac{FP}{(TN + FP)} \tag{9}$$

$$ErrorRate = \frac{FN + FP}{(TP + TN + FP + FN)} \tag{10}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{11}$$

4. Experimental Results and Discussion

Eight architectural classifiers were proposed and trained using four approaches (LAECNet, AlexNet, InceptionV3, and MobileNet) and two datasets (with and without DA) were used. Then, classifiers were utilized to sort and classify the samples of images in the test dataset. Figures 8 and 9 reveal the loss and training accuracy during validation and training per epoch of the suggested algorithms. All the architectures were trained with 100 epochs and a BS of 16 images on both datasets. The detection process was carried out using a modified YOLOv5s [62] object detector with epochs of 128 and 196, a batch size of 2, and a threshold of 0.5.

The performance evaluation indictors used were accuracy, F1, precision, error rate, mAP, and recall, and these were calculated using Equations (4)–(11).

$$I(x, y) = \begin{cases} Crack, & \text{if } Value < 0 \\ Non - Crack, & \text{otherwise} \end{cases} \tag{12}$$

Using Equation (12), the models' ability to predict whether the input image I(x,y) was cracked or non-cracked can be demonstrated. We noted that the models predicted the best possible outcomes.

Figure 8 summarizes the loss (upper) and accuracy (lower) changes during the training of the proposed algorithms. As shown in Figure 8 (upper portion), these validation and training losses fluctuated, with the losses fluctuating separately across the epochs.

The LAECNet algorithm showed a noticeable gap between its two-loss curves, indicating an over-fitting problem (unrepresentative training data compared to validation data). In the same graphs, the Alexnet algorithm also exhibited some over-fitting due to unrepresentative validation data compared to the training data samples. However, Figure 8 (lower portion) indicates training convergence, with a decline in losses and an increase in accuracy.

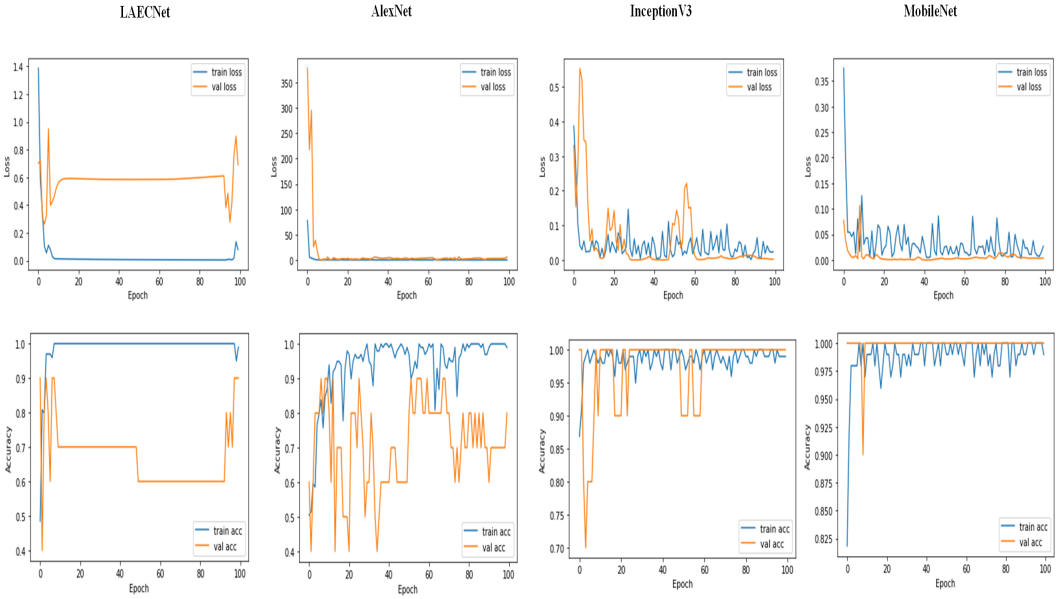

Figure 8. Losses (**upper portion**) and accuracy (**lower portion**) during training and validation.

Figure 9 summarizes both the loss and accuracy of the algorithms obtained when using the DA techniques.

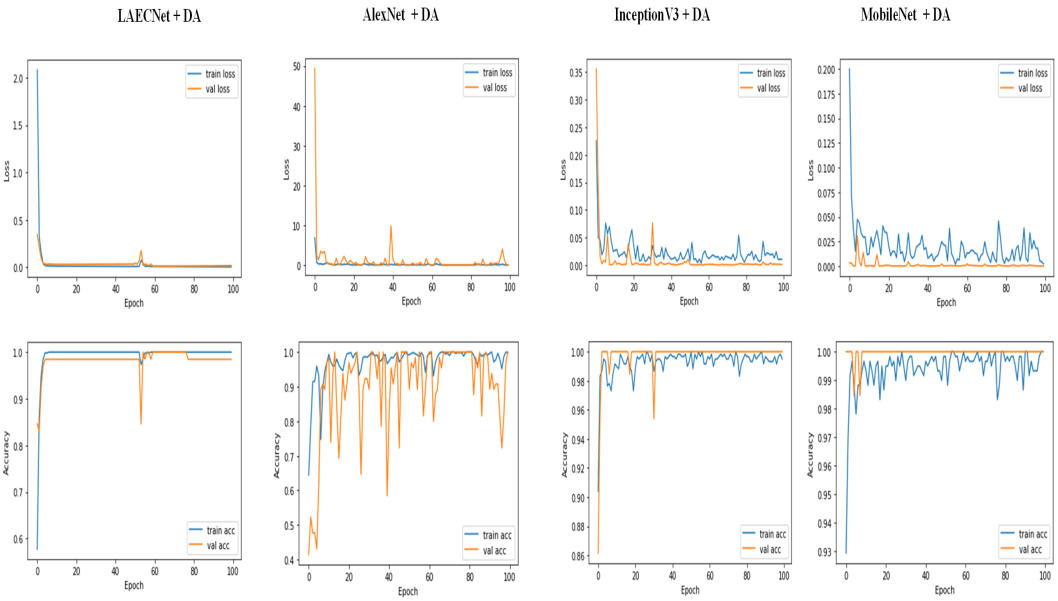

Figure 9. Losses (**upper**) and accuracy (**lower**) during training and validation using DA techniques.

As shown in Figure 9 (upper portion), the algorithms still exhibited the problem of some over-fitting in the case of LAECNet + DA and AlexNet + DA. In addition, the InceptionV3 + DA and MobileNet + DA algorithms also showed some under-fitting. However, the algorithms with DA achieved better classification accuracy compared to the algorithms without DA.

Among the ConvNets trained with 100 epochsand with a batch size of 16 (Table 2), on the dataset without DA, the LAECNet classifier achieved accuracy, F1, precision, and recall values of 89%, 83%, 89%, and 82%, respectively; the AlexNet classifier achieved accuracy, F1, precision, and recall values of 93%, 93%, 93%, 93%, and 0.93, respectively; and the InceptionV3 classifier achieved accuracy, F1, precision, and recall values of 93%, 93%, 94%, and 93%, respectively. However, the MobileNet ConvNet achieved better classification accuracy, F1, precision, and recall values of 96%, 96%, 97%, and 96%, respectively. Table 4 and the accuracy graph presented in Figure 10 indicate that were was robustness among the results of the rest of the classifiers. In addition, there was still a problem of misclassification and over-fitting, and the MobileNet model without data augmentation was found to be the model with better performance, having lower recall due to the lower quantity of images labeled as "crack" images compared to the number of samples labeled "non-crack" images, as shown in Table 5.

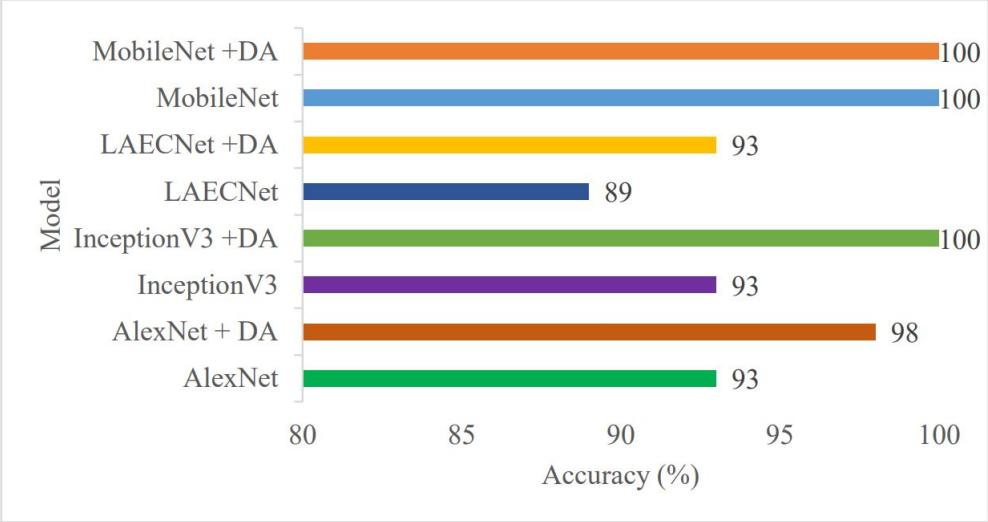

Figure 10. The classification accuracy of the proposed model.

Table 4. Accuracy comparison of the proposed models.

Models	Accuracy (%)
AlexNet	93
AlexNet + DA	98
InceptionV3	93
InceptionV3 + DA	100
LAECNet	89
LAECNet + DA	93
MobileNet	100
MobileNet + DA	100

Table 5. Classification results for each class type.

Models	Classes	Precision (%)	Recall (%)	F1 (%)
AlexNet	Crack	88	881	88
	Non-Crack	95	95	95
AlexNet + DA	Crack	94	100	97
	Non-Crack	100	97	99
InceptionV3	Crack	100	75	86
	Non-Crack	91	100	95
InceptionV3 + DA	Crack	100	100	100
	Non-Crack	100	100	100
LAECNet	Crack	62	100	76
	Non-Crack	100	75	86
LAECNet + DA	Crack	89	981	93
	Non-Crack	99	95	97
MobileNet	Crack	100	88	93
	Non-Crack	95	100	98
MobileNet + DA	Crack	100	100	100
	Non-Crack	100	100	100

Table 6 depicts the test performance of the proposed algorithms on the test dataset samples. It can be seen that both the InceptionV3 + DA and MobileNet + DA models were able to predict the categories successfully with an error rate of zero (0).

Table 6. Test performance of the proposed crack classifiers.

Models	TP	FP	FN	TN	Error Rate
AlexNet	7	1	1	19	0.0714
AlexNet + DA	51	0	3	111	0.018
InceptionV3	6	2	0	20	0.071
InceptionV3 + DA	51	0	0	20	0.071
LAECNet	8	0	5	15	0.179
LAECNet + DA	50	1	6	108	0.042
MobileNet	7	1	0	20	0.035
MobileNet + DA	51	0	0	114	0

Table 5 shows detection and classification performance measures for the crack and non-crack type classes in the sample images.The proposed frameworks (Alexnet, InceptionV3, LAECNet, and MobileNet) analyzed cracks with F1 scores of 97%, 100%, 93%, and 100%, respectively, with the DA techniques, and for non-crack samples they achieved F1 scores of 99%, 100%, 97%, and 100%, respectively. However, without DA training techniques and with fewer crack image samples, they obtained scores of 80%, 86%, 76%, and 93%.

To minimize the problems of misclassification and over-fitting, and to improve their recall, we implemented DA techniques to improve the performance and over-fitting of the classifiers with 100 epochs and a BS of 16 images. The LAECNet architecture improved its accuracy, F1, precision, and recall by 4%, 10%, 6%, and 11%, respectively. The AlexNet architecture also improved its accuracy, F1, precision, and recall by 5%, 5%, 5%, and 5%, respectively; The InceptionV3 architecture improves its accuracy, F1, Precision, and Recall by 7%, 7%, 6%, and 7%, respectively. The MobileNet architecture improved its accuracy by 5%. As shown in Table 5, the MobileNet classifier achieved improved F1, precision, and recall scores of 100%, 100%, and 100%, respectively, which were superior to those of the other seven (7) classifier models trained with 100 epochs and a BS of 16 images on the augmented dataset.

Table 3 shows the values of the diverse hyper-parameters of the proposed MobileNet architecture with and without DA. These parameters were chosen based on a comparison between the other proposed models such as LAECNets and the other pre-trained net architectures in order to fix the ability of each neural network architecture to identify/detect and classify cracks in automotive engine components using samples of test images.

4.1. Computational Complexity

The computational complexity (CC) was evaluated by exploring various parameters for the training and testing of the algorithms. This measure is related to the complexity of the DL algorithm, with the costs during training and testing are used to compute the measures of complexity for the DL models.

The CC of the presented algorithms was investigated according to their training and testing times and was compared with those of six (6) pre-trained algorithms (AlexNet, AlexNet + DA, InceptionV3, InceptionV3 + DA, MobileNet, and MobileNet + DA) and two Light ConvNets (LAECNet, and LAECNet + DA). Here, we compare the CC scores of the DL algorithms. As depicted in Table 7, the InceptionV3 model exhibited the most lengthy testing and training time among the four models without DA. MobileNet + DA had the shortest training time and considerable testing time compared to other networks because MobileNet + DA adopted fewer parameters than the AlexNet, InceptionV3, and LAECNet network architectures. From the perspective of classification accuracy (Table 4), and reducing computing resources, the MobileNet model (with DA) achieved remarkable performance with less computational complexity, which also demonstrates the feasibility and superiority of this approach for automotive engine component crack classification.

Table 7. Complexity comparison of various proposed networks using real and augmented data.

Models	TrT	TT
AlexNet	312.51	1045
AlexNet + DA	759.75	372.60
InceptionV3	488.77	4598
InceptionV3 + DA	1455.23	2870
LAECNet	348.12	568.10
LAECNet + DA	872.54	192.80
MobileNet	341.52	1392
MobileNet + DA	702.02	1300

4.2. Crack Detector Using Modified YOLOv5s

In this study, the YOLOv5 detection algorithm was adopted [62]. It has high precision, high speed, and fewer model parameters Table 8 than the other object detector algorithms used.

We evaluated the detection ability of the algorithm by labeling and annotating the crack area of the engine part. Table 9 depicts the evaluation metrics of the proposed algorithm with 128, and 196 best epochs and a threshold of 0.5. Average mean precision, precision, and recall scores of 54%/91.2%, 99.9%/100%, and 50%/75% were achieved, respectively. As we observed, the recall of the crack area was comparatively lower due to the lower volume of the training image dataset. Figure 11 depicts the results for the detected parts.

Table 8. Different hyper-parameters of the modified YOLOv5s architecture.

Parameters	Values
Weight Decay	5×10^{-4}
Momentum	0.937
Warm up Epochs	3
Learning Rate	0.01
Batch Size	2
Image Size	$640 \times 640 \times 3$
Training Weights	7.01×10^{6}
GFLOPs	15.80

Table 9. Detection results obtained using the modified YOLOv5s architecture.

Epochs	Precision	Recall	mAP
128	99.90%	50%	54%
196	100%	75%	91.20%

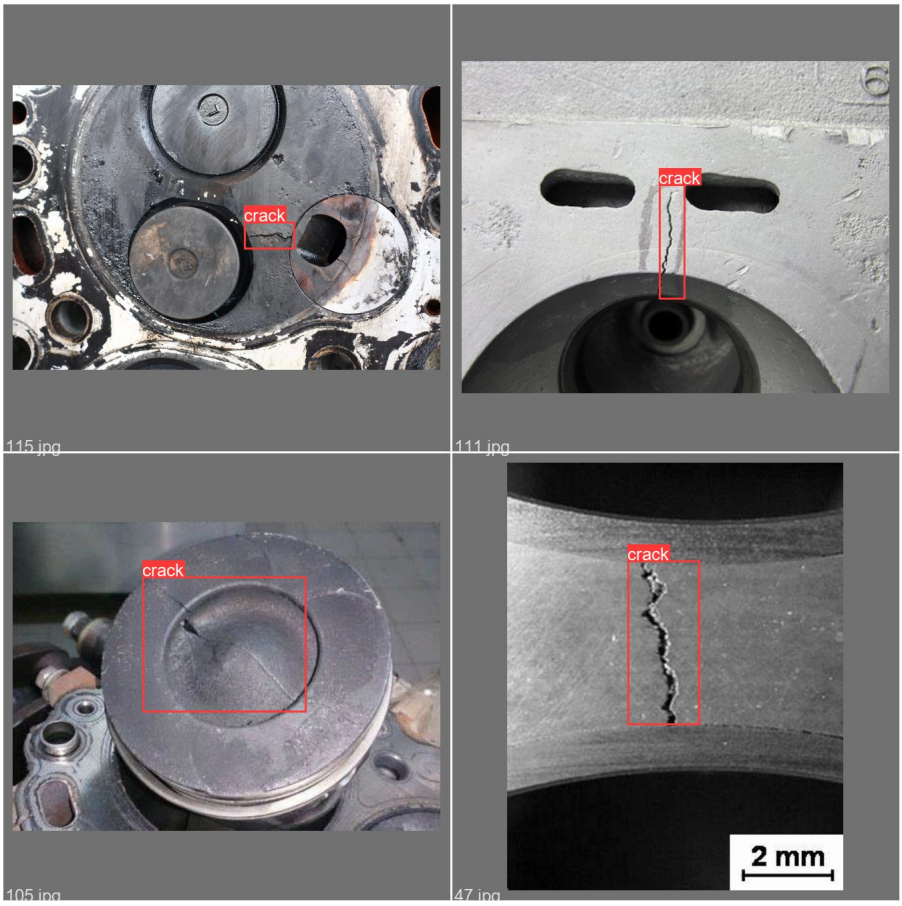

Figure 11. Sample images of cracks detected using YOLOv5.

5. Conclusions

In this study, we established a robust crack classifier architecture for classifying cracks in automotive engine components using a DCNN architecture. Two standard approaches to training a crack classifier in vision-based research include (1) constructing a light ConvNet architecture (LAECNet) from scratch, and (2) using the FT and TL techniques of three pre-trained ConvNet architectures (i.e., AlexNet, InceptionV3, and MobileNet).

The performance of all the suggested ConvNet architectures was evaluated based on their accuracy, F1, precision, error rate, and recall scores obtained on a test sample of images. LAECNet, AlexNet, InceptionV3, and MobileNet achieved accuracy scores of 89%, 93%, 93%, and 96%, respectively. DA approaches robustly increased the classification accuracy of each model by 4%, 5%, 7%, and 4%, respectively. Generally, using TL and FT of the MobileNet architecture achieved better performance indexes of the error rate, recall, and precision and robustness. Thus, these findings imply that a robust crack classifier and detector can be trained powerfully on inadequate crack image samples using the FT and TL techniques from a benchmark ConvNet pre-trained model on the whole image sample in association with the use of DA techniques. We were also successful in detecting the cracked parts of the engine components using the YOLOv5s architecture with TL. Nevertheless, the training dataset was relatively small compared to other deep-learning architecture datasets, but numerous measurable calculations revealed the remarkable performance of the proposed method.

In this work, we aimed to establish a robust crack detection and classification system for cracks in automotive engine components. The establishment of a real-time classification and detection scheme will be the focus of future works.

Author Contributions: Conceptualization M.A.B. and Y.F.; investigation. M.A.B., J.M. and N.Y.; writing—original draft preparation, M.A.B., J.M., N.Y., Z.L. and Y.L.; experiments, M.A.B. and Y.F.; review and editing, M.A.B., J.M., N.Y., Z.L. and Y.L.; supervision, Y.F.; funding acquisition, Y.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by funds for the National Science Foundation of China under Grant no. 62141101, and Natural Science Basic Research Program of Shaanxi Province, China under grant no. 2021JM-188.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Erjavec, J.; Thompson, R. *Automotive Technology: A Systems Approach*; Cengage Learning: Boston, MA, USA, 2014.
2. Asai, K.; Takeuchi, A.; Ueda, N.; Kawamoto, J. Computerized ultrasonic inspection system for ceramic pre-combustion chambers of automotive diesel engines. *SAE Trans.* **1985**, *94*, 959–969.
3. Almubarak, H.A.; Albloushi, A.H. Automotive Engine Tests "The Basics". Available online: https://www.academia.edu/50079718/Automotive_Engine_Tests_The_Basics (accessed on 17 January 2022).
4. Berwo, M.A.; Fang, Y.; Mahmood, J.; Retta, E.A. Automotive Engine Cylinder Head Crack Detection: Canny Edge Detection with Morphological Dilation. In Proceedings of the 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Tokyo, Japan, 14–17 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1519–1527.
5. Wu, J.; Zhang, Q.; Xu, G. Tiny imagenet challenge. *Tech. Rep.* **2017**, 1–9.
6. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.
7. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
9. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
10. Qu, Z.; Shen, J.; Li, R.; Liu, J.; Guan, Q. Partsnet: A unified deep network for automotive engine precision parts defect detection. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen China, 8–10 December 2018; pp. 594–599.
11. Elghaish, F.; Talebi, S.; Abdellatef, E.; Matarneh, S.T.; Hosseini, M.R.; Wu, S.; Mayouf, M.; Hajirasouli, A.; et al. Developing a new deep learning CNN model to detect and classify highway cracks. *J. Eng. Des. Technol.* **2021**. [CrossRef]
12. Cheng, Y.; Deng, H.; Feng, Y.; Xiang, J. Weld Defect Detection and Image Defect Recognition Using Deep Learning Technology. **2021**, 1–14. Available online: https://www.researchsquare.com/article/rs-149365/v1 (accessed on 17 January 2022). .
13. Zhang, Z.; Wen, G.; Chen, S. Weld image deep learning-based on-line defects detection using convolutional neural networks for Al alloy in robotic arc welding. *J. Manuf. Process.* **2019**, *45*, 208–216. [CrossRef]
14. Akhtar, S.; Tandiya, A.; Moussa, M.; Tarry, C. An Efficient Automotive Paint Defect Detection System. *Adv. Sci. Technol. Eng. Syst. J.* **2019**, *4*, 171–182. [CrossRef]
15. Thuan, D. Evolution of Yolo Algorithm and Yolov5: The State-of-the-Art Object Detection Algorithm. 2021; pp. 1-61. Available online: https://www.theseus.fi/bitstream/handle/10024/452552/Do_Thuan.pdf?isAllowed=y&sequence=2 (accessed on 17 January 2022).
16. Li, S.; Zhao, X. Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Adv. Civ. Eng.* **2019**. [CrossRef]
17. Nogay, H.S.; Akinci, T.C.; Yilmaz, M. Detection of invisible cracks in ceramic materials using by pre-trained deep convolutional neural network. *Neural Comput. Appl.* **2021**, 1–10. [CrossRef]

18. Wang, R.; Guo, Q.; Lu, S.; Zhang, C. Tire defect detection using fully convolutional network. *IEEE Access* **2019**, *7*, 43502–43510. [CrossRef]
19. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, pp. 679–698. [CrossRef]
20. Salman, M.; Mathavan, S.; Kamal, K.; Rahman, M. Pavement crack detection using the Gabor filter. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, 6–9 October 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 2039–2044.
21. Zalama, E.; Gómez-García-Bermejo, J.; Medina, R.; Llamas, J. Road crack detection using visual features extracted by Gabor filters. *Comput.-Aided Civ. Infrastruct. Eng.* **2014**, *29*, 342–358. [CrossRef]
22. Nigam, R.; Singh, S.K. Crack detection in a beam using wavelet transform and photographic measurements. In *Proceedings of the Structures*; Elsevier: Amsterdam, The Netherlands, 2020; Volume. 25, pp. 436–447.
23. Jiang, X.; Ma, Z.J.; Ren, W.X. Crack detection from the slope of the mode shape using complex continuous wavelet transform. *Comput.-Aided Civ. Infrastruct. Eng.* **2012**, *27*, 187–201. [CrossRef]
24. Kanter, J.M. Color Crack: Identifying Cracks in Glass. *Dated Dec.* **2014**, *9*. Available online: https://www.jmaxkanter.com/papers/color_crack.pdf (accessed on 17 January 2022).
25. Elhariri, E.; El-Bendary, N.; Taie, S.A. Performance analysis of using feature fusion for crack detection in images of historical buildings. In Proceedings of the 11th International Conference on Management of Digital EcoSystems, Limassol, Cyprus, 12–14 November 2019; pp. 308–315.
26. Prasanna, P.; Dana, K.; Gucunski, N.; Basily, B. Computer-vision based crack detection and analysis. In Proceedings of the Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012, San Diego, CA, USA, 12–15 December 2012; International Society for Optics and Photonics: Bellingham, WA, USA, 2012; Volume 8345, p. 834542.
27. Prasanna, P.; Dana, K.J.; Gucunski, N.; Basily, B.B.; La, H.M.; Lim, R.S.; Parvardeh, H. Automated crack detection on concrete bridges. *IEEE Trans. Autom. Sci. Eng.* **2014**, *13*, 591–599. [CrossRef]
28. Bu, G.; Chanda, S.; Guan, H.; Jo, J.; Blumenstein, M.; Loo, Y. Crack detection using a texture analysis-based technique for visual bridge inspection. *Electron. J. Struct. Eng* **2015**, *14*, 41–48.
29. Wang, S.; Yang, F.; Cheng, Y.; Yang, Y.; Wang, Y. Adaboost-based Crack Detection Method for Pavement. In *Proceedings of the IOP Conference Series: Earth and Environmental Science*; IOP Publishing: Bristol, UK, 2018; Volume 189, p. 022005.
30. Fan, R.; Bocus, M.J.; Zhu, Y.; Jiao, J.; Wang, L.; Ma, F.; Cheng, S.; Liu, M. Road crack detection using deep convolutional neural network and adaptive thresholding. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 474–479.
31. Lee, K.; Yi, S.; Hyun, S.; Kim, C. Review on the Recent Welding Research with Application of CNN-Based Deep Learning Part I: Models and applications. *J. Weld. Join.* **2021**, *39*, 10–19. [CrossRef]
32. Mashayekhi, M.; Santini-Bell, E.; Azam, S.E. Fatigue crack detection in welded structural components of steel bridges using artificial neural network. *J. Civ. Struct. Health Monit.* **2021**, 1–17. [CrossRef]
33. Lee, D.; Kim, J.; Lee, D. Robust concrete crack detection using deep learning-based semantic segmentation. *Int. J. Aeronaut. Space Sci.* **2019**, *20*, 287–299. [CrossRef]
34. Dorafshan, S.; Thomas, R.J.; Maguire, M. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data Brief* **2018**, *21*, 1664–1668. [CrossRef]
35. Rajadurai, R.S.; Kang, S.T. Automated Vision-Based Crack Detection on Concrete Surfaces Using Deep Learning. *Appl. Sci.* **2021**, *11*, 5229. [CrossRef]
36. Ali, L.; Alnajjar, F.; Jassmi, H.A.; Gochoo, M.; Khan, W.; Serhani, M.A. Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures. *Sensors* **2021**, *21*, 1688. [CrossRef]
37. Han, Z.; Chen, H.; Liu, Y.; Li, Y.; Du, Y.; Zhang, H. Vision-Based Crack Detection of Asphalt Pavement Using Deep Convolutional Neural Network. *Iran. J. Sci. Technol. Trans. Civ. Eng.* **2021**, 1–9. [CrossRef]
38. Han, J.; Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the International Workshop on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 195–201.
39. Costarelli, D.; Spigler, R. Approximation results for neural network operators activated by sigmoidal functions. *Neural Netw.* **2013**, *44*, 101–106. [CrossRef] [PubMed]
40. Cortes, C.; Mohri, M.; Rostamizadeh, A. L2 regularization for learning kernels. *arXiv* **2012**, arXiv:1205.2653.
41. Van Laarhoven, T. L2 regularization versus batch and weight normalization. *arXiv* **2017**.arXiv:1706.05350.
42. Goodfellow, I.; Bengio, Y.; Courville, A. Regularization for deep learning. *Deep. Learn.* **2016**, 216–261. Available online: https://www.deeplearningbook.org/ (accessed on 17 January 2022)
43. Bach, S.; Broecheler, M.; Getoor, L.; O'leary, D. Scaling MPE inference for constrained continuous Markov random fields with consensus optimization. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 2654–2662.
44. Rosasco, L.; De Vito, E.; Caponnetto, A.; Piana, M.; Verri, A. Are loss functions all the same? *Neural Comput.* **2004**, *16*, 1063–1076. [CrossRef]
45. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.

46. Tan, S.; Lu, G.; Jiang, Z.; Huang, L. Improved YOLOv5 Network Model and Application in Safety Helmet Detection. In Proceedings of the 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Tokoname, Japan, 4–6 March 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 330–333.
47. Sharma, V. Face Mask Detection using YOLOv5 for COVID-19. Ph.D. Thesis, California State University San Marcos, San Marcos, CA, USA, 2020.
48. Li, S.; Li, Y.; Li, Y.; Li, M.; Xu, X. YOLO-FIRI: Improved YOLOv5 for Infrared Image Object Detection. *IEEE Access* **2021**, *9*, 141861–141875. [CrossRef]
49. Jubayer, F.; Soeb, J.A.; Mojumder, A.N.; Paul, M.K.; Barua, P.; Kayshar, S.; Akter, S.S.; Rahman, M.; Islam, A. Detection of mold on the food surface using YOLOv5. *Curr. Res. Food Sci.* **2021**, *4*, 724–728. [CrossRef]
50. Kasper-Eulaers, M.; Hahn, N.; Berger, S.; Sebulonsen, T.; Myrland, Ø.; Kummervold, P.E. Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5. *Algorithms* **2021**, *14*, 114. [CrossRef]
51. Rahman, R.; Bin Azad, Z.; Bakhtiar Hasan, M. Densely-Populated Traffic Detection Using YOLOv5 and Non-maximum Suppression Ensembling. In *Proceedings of the International Conference on Big Data, IoT, and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 567–578.
52. Kumar, A.; Sodhi, S.S. Comparative analysis of gaussian filter, median filter and denoise autoenocoder. In Proceedings of the 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 12–14 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 45–51.
53. Safari, M.; Aghagolzadeh, A. FIR filter based fuzzy-genetic mixed noise removal. In Proceedings of the 2007 9th International Symposium on Signal Processing and Its Applications, Sharjah, United Arab Emirates, 12–15 February 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 1–4.
54. Caponetto, R.; Fortuna, L.; Vinci, C. Design of fuzzy filters by genetic algorithms. In Proceedings of the IEEE International Symposium on Circuits and Systems-ISCAS'94, London, UK, 30 May–2 June 1994; IEEE: Piscataway, NJ, USA, 1994; Volume 5, pp. 177–180.
55. Peng, S.; Lucke, L. Fuzzy filtering for mixed noise removal during image processing. In Proceedings of the 1994 IEEE 3rd International Fuzzy Systems Conference, Orlando, FL, USA, 26–29 June 1994; IEEE: Piscataway, NJ, USA, 1994; pp. 89–93.
56. Patidar, P.K.; Dadheech, P. Performance of Fuzzy Filter and Mean Filter for Removing Gaussian Noise. *Int. J. Comput. Appl.* **2019**, *975*, 8887.
57. Inoue, H. Data augmentation by pairing samples for images classification. *arXiv* **2018**, arXiv:1801.02929.
58. Salinas, D.; Shen, H.; Perrone, V. Supplementary Material: A Quantile-based Approach for Hyperparameter Transfer Learning. Available online: https://proceedings.mlr.press/v119/salinas20a.html (accessed on 17 January 2022).
59. Gao, Y.; Liu, Y.; Zhang, H.; Li, Z.; Zhu, Y.; Lin, H.; Yang, M. Estimating gpu memory consumption of deep learning models. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual, 9–13 November 2020; pp. 1342–1352.
60. Toal, D.J.; Bressloff, N.W.; Keane, A.J. Kriging hyperparameter tuning strategies. *Aiaa J.* **2008**, *46*, 1240–1252. [CrossRef]
61. Lorenzo, P.R.; Nalepa, J.; Kawulok, M.; Ramos, L.S.; Pastor, J.R. Particle swarm optimization for hyper-parameter selection in deep neural networks. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin Germany, 15–19 July 2017; pp. 481–488.
62. Ultralytics. Yolov5. Available online: https://github.com/ultralytics/yolov5 (accessed on 17 January 2022).