

## Article

# UAV-Based Bridge Inspection via Transfer Learning

Mostafa Aliyari <sup>1</sup>, Enrique Lopez Droguett <sup>2</sup> and Yonas Zewdu Ayele <sup>1,3,\*</sup>

<sup>1</sup> Faculty of Computer Science, Engineering and Economics, Østfold University College, 1757 Halden, Norway; mostafa.aliyari@hiof.no

<sup>2</sup> Department of Civil and Environmental Engineering, Garrick Institute for the Risk Sciences, University of California, Los Angeles, CA 90024, USA; eald@g.ucla.edu

<sup>3</sup> Department of Risk, Safety and Security, Institute for Energy Technology, 1777 Halden, Norway

\* Correspondence: yonas.ayele@ife.no

**Abstract:** As bridge inspection becomes more advanced and more ubiquitous, artificial intelligence (AI) techniques, such as machine and deep learning, could offer suitable solutions to the nation's problems of overdue bridge inspections. AI coupling with various data that can be captured by unmanned aerial vehicles (UAVs) enables fully automated bridge inspections. The key to the success of automated bridge inspection is a model capable of detecting failures from UAV data like images and films. In this context, this paper investigates the performances of state-of-the-art convolutional neural networks (CNNs) through transfer learning for crack detection in UAV-based bridge inspection. The performance of different CNN models is evaluated via UAV-based inspection of Skodsborg Bridge, located in eastern Norway. The low-level features are extracted in the last layers of the CNN models and these layers are trained using 19,023 crack and non-crack images. There is always a trade-off between the number of trainable parameters that CNN models need to learn for each specific task and the number of non-trainable parameters that come from transfer learning. Therefore, selecting the optimized amount of transfer learning is a challenging task and, as there is not enough research in this area, it will be studied in this paper. Moreover, UAV-based bridge inspection images require specific attention to establish a suitable dataset as the input of CNN models that are trained on homogenous images. However, in the real implementation of CNN models in UAV-based bridge inspection images, there are always heterogeneities and noises, such as natural and artificial effects like different luminosities, spatial positions, and colors of the elements in an image. In this study, the effects of such heterogeneities on the performance of CNN models via transfer learning are examined. The results demonstrate that with a simplified image cropping technique and with minimum effort to preprocess images, CNN models can identify crack elements from non-crack elements with 81% accuracy. Moreover, the results show that heterogeneities inherent in UAV-based bridge inspection data significantly affect the performance of CNN models with an average 32.6% decrease of accuracy of the CNN models. It is also found that deeper CNN models do not provide higher accuracy compared to the shallower CNN models when the number of images for adoption to a specific task, in this case crack detection, is not large enough; in this study, 19,023 images and shallower models outperform the deeper models.

**Keywords:** UAV; bridge; inspection; convolutional neural networks (CNN); deep learning (DL); transfer learning; VGG; ResNet; Xception; inception; NASNet; DenseNet; EfficientNet



**Citation:** Aliyari, M.; Droguett, E.L.; Ayele, Y.Z. UAV-Based Bridge Inspection via Transfer Learning. *Sustainability* **2021**, *13*, 11359. <https://doi.org/10.3390/su132011359>

Academic Editor:  
George D. Bathrellos

Received: 11 August 2021  
Accepted: 5 October 2021  
Published: 14 October 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The importance of bridges as a key element in civil infrastructures is getting more attention due to many recent bridge collapses [1,2]. Regular inspections for high numbers of bridges in conventional inspection procedures are unsafe, costly, time-consuming, and labor-intensive [3,4]. UAVs are among the best alternatives for conventional inspections, predominantly due to the higher safety of workers, lower cost, and tangible scientific improvements. Several studies have shown the successful implementation of UAVs in

bridge inspection [2,5–8]. Moreover, recent advances in the field of artificial intelligence (AI), especially in machine and deep learning, have offered suitable solutions in many different practical fields such as bridge inspection. AI coupling with various data that are acquired with unmanned aerial vehicles (UAVs) could enable bridge inspections to become automated. The key to the success of automated bridge inspection is a model capable of detecting failures from UAV images and films.

UAV-based bridge inspection images are usually inspected by human inspectors to identify any possible failure. This process is time-consuming, costly, and dependent on the knowledge and experience of inspectors, which are prone to objectivity and inaccuracy. To address these issues, a great deal of research has been devoted to developing methods and techniques to extract information regarding any possible failure from visual inputs. These methods and techniques mostly fall into two categories: image processing and deep learning-based failure detection. A wide variety of image processing methods and techniques are used in crack detection, including morphological approaches, digital image correlation, wavelet transform, median filtering, threshold methods, random structured forests, the photogrammetric technique, the recognition technique, and edge detectors such as the Canny, Sobel, Gabor, and Prewitt [9–16]. Recently, convolutional neural networks (CNNs) have gained a lot of attention and are widely used in the crack detection task [15,17–20]. For instance, Shengyuan Li and Xuefeng Zhao designed a CNN architecture of binary-class output for crack detection by modifying the AlexNet architecture [17]. More recently, CNN models have been used via transfer learning, which is learning a new task through the transfer of knowledge from a related task that is already learned. Transfer learning is used in a wide range of classification tasks, such as synthetic aperture radar (SAR) target classification [21], plant classification [22], and molecular cancer classification [23]. Transfer learning is also used in crack detection; for example, Kasthurirangan Gopalakrishnan et al. [24] used the VGG16 pre-trained model for crack detection and compared its performance with other machine learning classifiers like random forest (RF) and support vector machine (SVM). Cao Vu Dung and Le Duc Anh [25] used three different pre-trained CNN models with the VGG16-based encoder for crack segmentation and classification.

Although these studies show the usefulness of transfer learning in a wide range of classification tasks, there are still specific challenges regarding how to use it. For example, there is a huge difference in the number of images that these CNN models are already trained in: there are more than 14 million images in the ImageNet dataset [26], and the number of images that the last part of these CNN models were trained on for adoption to the specific task, in this case the crack detection task, was almost 19,023. This is what makes transfer learning important; if there is no such difference, transfer learning seems irrelevant as the CNN models can be trained directly on that big crack dataset. Thus, there is a trade-off between the number of trainable parameters that CNN models need to learn for each specific task and the number of non-trainable parameters that come from transfer learning. Therefore, selecting the right number of trainable parameters is a challenging task, and as there is not enough research in this area, it will be studied in this paper. Moreover, in the previously mentioned studies, CNN models for crack detection are mostly built from scratch or on top of a well-known architectures such as AlexNet and VGG, and the potential of existing state-of-the-art CNN models in UAV-based bridge inspections via transfer learning is rarely exploited. In this study, state-of-the-art CNN models are used via transfer learning to explore their real potential for UAV-based bridge crack detection. UAV-based bridge inspection images require specific attention to establish a suitable dataset as the input of CNN models that are trained on homogenous images, but in the real implementation of CNN models in UAV-based bridge inspection images there are always heterogeneities in the images, such as natural and artificial effects like different luminosities, spatial positions, and colors of the elements in an image. Therefore, in this study the effects of such heterogeneities on the performance of the CNN models



via transfer learning are examined as well. This research study attempts to answer the following main questions:

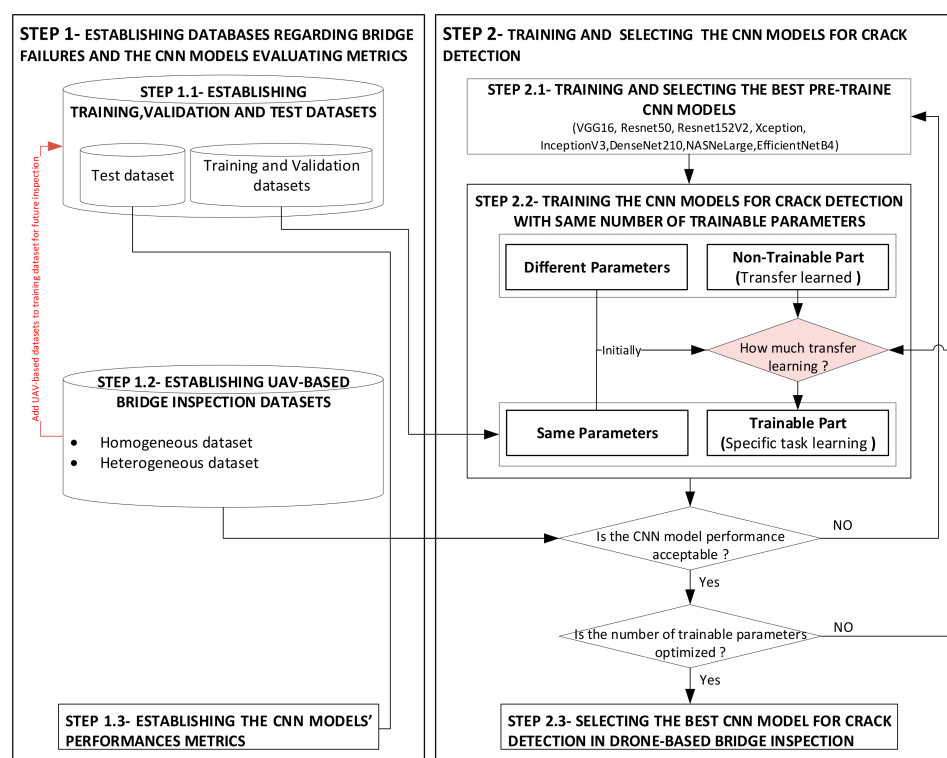
- How can drone-based bridge inspection benefit from the CNN models via transfer learning?
- How much do heterogeneities and noises between training and testing datasets affect the CNN models via transfer learning?
- How much transfer learning must be used?

To answer these questions, the weight of transfer learning in eight different CNN models was adjusted and evaluated in different datasets from drone-based bridge inspection considering the noises.

The remainder of this paper is structured as follows: in Section 2, the methodology and specific steps that help to identify the most suitable CNN model for UAV-based bridge inspection via transfer learning are discussed in detail; the results are presented in Section 3; Section 4 provides the discussion; and finally, Section 5 delivers the conclusions of this paper.

## 2. Methodology

The specific steps that help to identify the most suitable CNN model for UAV-based bridge inspection through transfer learning are shown in Figure 1 and explained in the following sections. Moreover, these specific steps establish the methodology to answer the research questions presented in Section 1 of this study.



**Figure 1.** Steps of using transfer learning in UAV-based bridge inspection.

### *STEP 1-Establishing Databases Regarding Bridge Failures and the CNN Model's Evaluating Metrics*

In this section, three datasets are established: (1) the SDNET2018 dataset, (2) heterogeneous UAV-based bridge inspection dataset, and (3) homogeneous UAV-based bridge inspection dataset. These datasets are different regarding the presence of some heterogeneities and noises, as will be explained.

### *STEP 1.1-Establishing Training, Validation, and Test Datasets*

The SDNET2018 is an annotated image dataset for training, validation, and benchmarking of artificial intelligence-based crack detection algorithms most suitable for concrete that contains over 56,000 cracked and non-cracked images [27]. Since there are many more non-cracked images than cracked images, to have a nearly balanced dataset in this study, 8484 crack images and 10,540 non-cracked images (total of 19,024) were used for training the trainable part of each CNN model and 20% of them were used for validation (1902) and testing (1902). Moreover, various transformations (such as rotation, zooming in, cropping, flipping, and so on) were performed on the original images in this study. The transformed images, also called data augmentation, were used for training, validation, and testing of the CNN models. A learned model may be more robust and accurate as it is trained on different variations of the same image, but the generated images are not different from the original ones. In this study, the number of training images was 15,219, with 5 different transformations of each original image (including shear range = 0.2, zoom range = 0.2, width shift range = 0.2, height shift range = 0.2, and horizontal flip), so the total number of unique images increased in the whole training across all epochs (not per epoch) to 76,000 images, and the same went for the validation and testing data.

### *STEP 1.2-Establishing UAV-Based Bridge Inspection Datasets*

The output of this step are the images that are categorized into two different datasets. One includes images with more noises and heterogeneities, and the other datasets include images with less noises. More details about establishing these UAV-based bridge inspection datasets are presented in the following sections.

#### *STEP 1.2.1-Conducting UAV-Based Bridge Inspection*

UAV-based bridge inspection was conducted on a 2-lane vehicular bridge, Skodsberg Bridge, located at latitude 59.2063° or 59°12'22.6" north, longitude 11.6932° or 11°41'35.4" east, and with an elevation of 115 m (377 feet) near Aremark, Viken county, Eastern Norway (Figure 2).

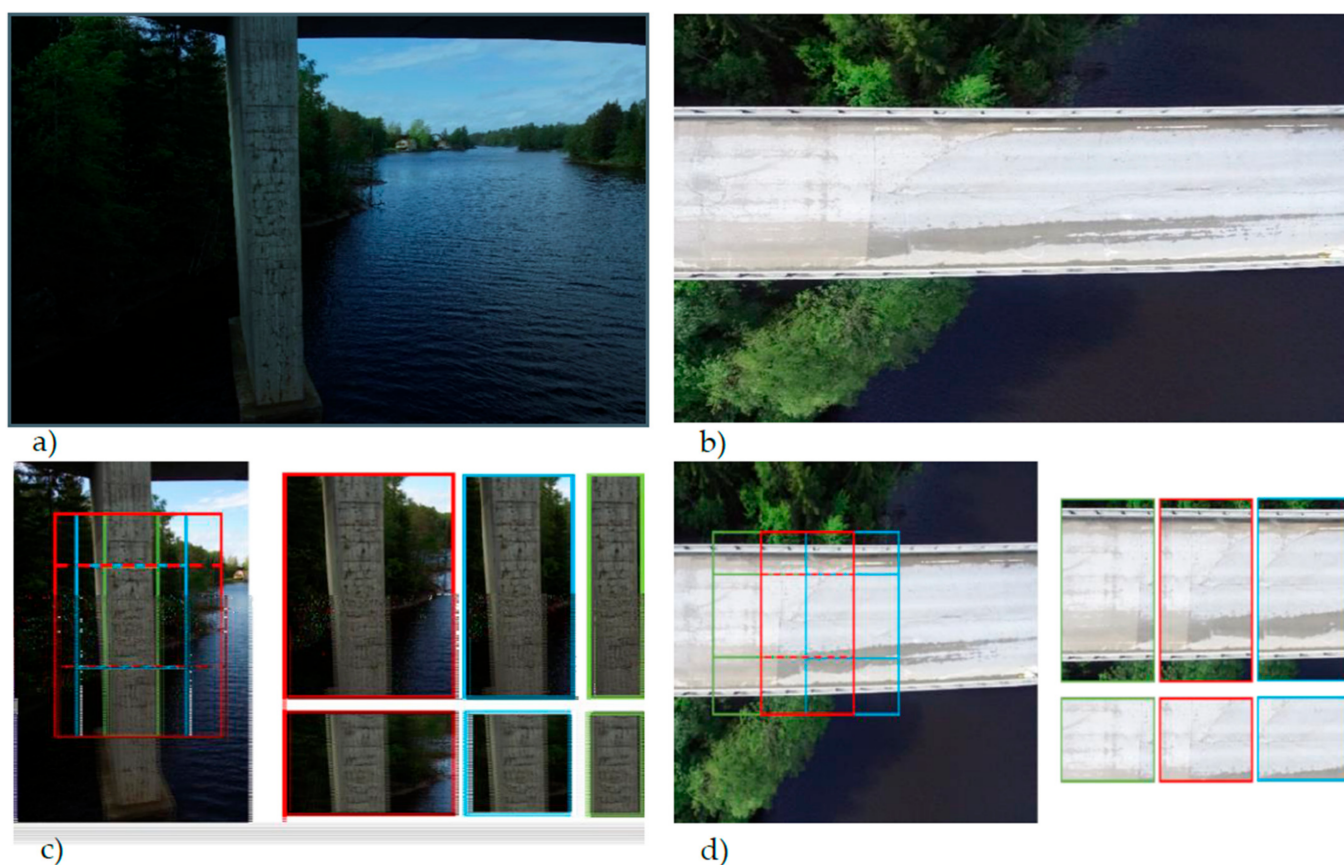


**Figure 2.** A perspective view of the inspected bridge.

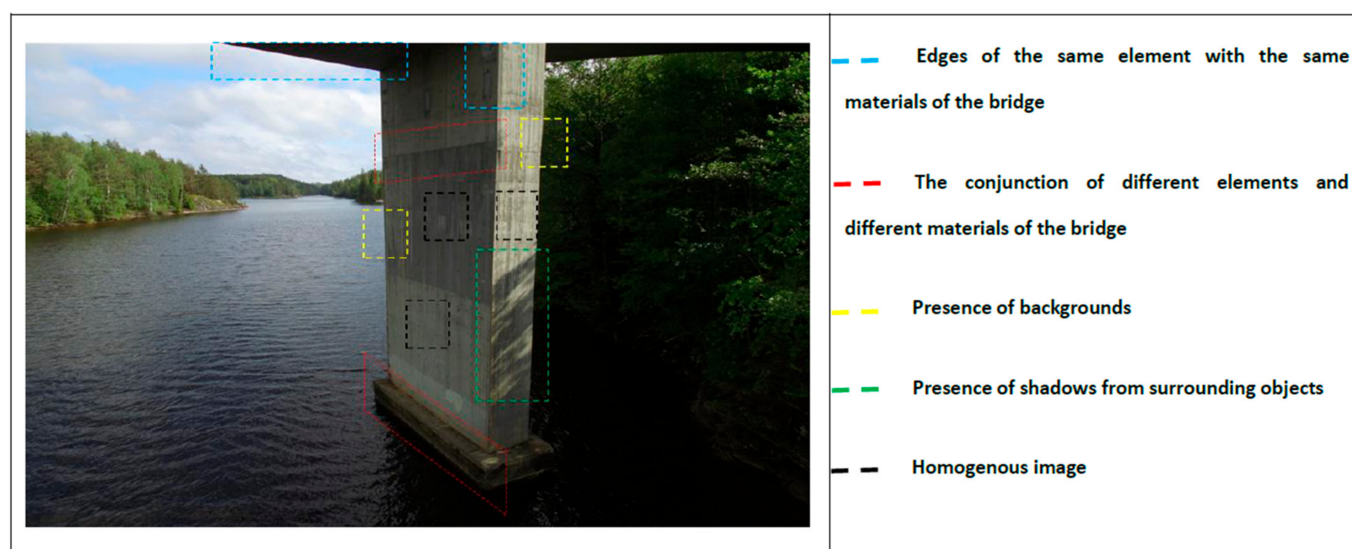
A DJI Matrice 100 with a Zenmuse Z3 aerial zoom camera and 7× zoom capacity was used to carry out the UAV-based bridge inspection; for more information, see [20]. As a result, images with 3000 × 4000 resolution were obtained; these high-resolution images allow for the capture of cropped images with high qualities (see images in Figure 3).

UAV-based bridge inspection images require specific attention to establish a suitable dataset as the input of CNN models that are trained on homogenous images. However, in the real implementation of CNN models for UAV-based bridge inspection images, there are always noises and heterogeneities in the images. Some sources of these heterogeneities are edges of the same element with the same materials, the conjunction of different elements or different materials of the bridge, the presence of backgrounds and shadows from

surrounding objects, and the shadows of other elements of bridge constructions as shown in Figure 4.



**Figure 3.** An Example of UAV Image from top deck and piers (a,b) and cropped images from them (c,d).

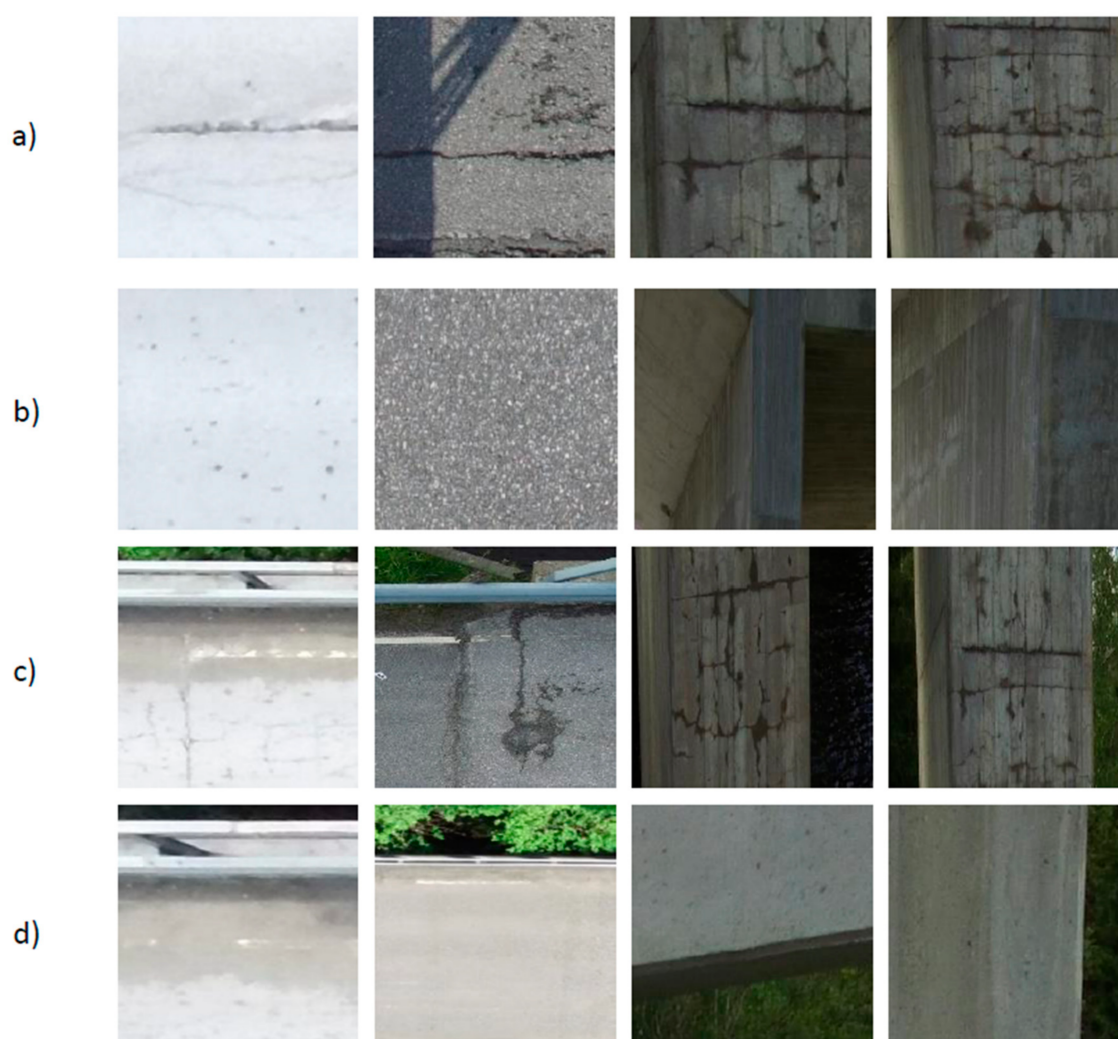


**Figure 4.** Some sources of heterogeneities in UAV-based bridge inception images.

A simple cropping strategy was performed to categorize the elements with identified cracks as a crack class, and cropped images from that element were captured and resized to the required input size for each CNN model. This simple cropping strategy was done with minimum effort; thus, there were more noises and this UAV-based bridge inspection



dataset was therefore called the “heterogeneous dataset”. As a result, 308 images belonging to 2 categories, crack and non-crack images (154 images for each category), were obtained in this heterogeneous dataset; a sample of this dataset is shown in Figure 5c,d. Then, the cropping was performed with more attention and effort to reduce heterogeneities in the images so all the backgrounds and the separation of different materials from the images were removed. Shadows and edges were impossible to remove through cropping alone; thus, there are a few images with shadows and edges in the second dataset, which is called the “homogeneous dataset”. As a result, 400 images belonging to the 2 categories, crack and non-crack images (200 images for each category), were obtained in this homogeneous dataset; a sample of this dataset is shown in Figure 5a,b. This allows for the evaluation of whether there is a meaningful difference in the performance of CNN models on UAV-based bridge inspection datasets and those images that CNN models are trained with.



**Figure 5.** A sample of crack (a,b) and non-crack (c,d) images in homogeneous dataset (a,b) and heterogeneous dataset (c,d).

### STEP 1.3-The CNN Models' Performance Metrics

The following metrics were used in this study to compare the performance of different CNN models. Accuracy is the ratio of correctly predicted observations to the total observations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (1)$$

where true positives (TP) are the correctly predicted positive values, true negatives (TN) are the correctly predicted negative values, false positives (FP) are when the actual class is no and the predicted class is yes, and false negatives (FN) are when the actual class is yes but the predicted class is no.

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

Recall (sensitivity) is the ratio of correctly predicted positive observations to all the observations in the actual class:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

The F1 score is the weighted average of precision and recall and takes both false positives and false negatives into account:

$$\text{F1 Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (4)$$

Since there was a little unbalancing in the dataset, as there were 8484 crack images and 10,540 non-cracked images (cracked images are 0.805 less), each performance metric was weighted accordingly as follows:

$$\text{Weighted Score} = 0.805 \times \text{Cracked class score} + 0.195 \times \text{Noncracked class score} \quad (5)$$

where cracked class score is the score that the crack class gets in term of accuracy, precision, recall, and F1 score and non-cracked class score is the score that the non-crack class gets in term of accuracy, precision, recall, and F1 score

However, macro average is not weighted and therefore can be calculated as follows:

$$\text{Macro Score} = 0.5 \times \text{Cracked class score} + 0.5 \times \text{Noncracked class score} \quad (6)$$

Moreover, the performance of the CNN models on the test datasets can be presented by a confusion matrix that, in the case of the two classes, crack and non-crack, consists of two rows and two columns that report the number of false positives, false negatives, true positives, and true negatives. This allows for a more detailed analysis than the mere proportion of correct classifications and all metrics can be calculated from these confusion matrices.

### ***STEP 2-Selecting and Training the CNN Models for Crack Detection***

Before selecting a pre-trained CNN model, for training the trainable part of CNN models, two issues need to be considered. Firstly, adopting complete transfer learning for the specific task, in this case crack detection, without any trainable parameters causes CNN models to over-parameterize to the point that it is just interpolating the data. Secondly, the low number of UAV-based bridge inspection images encourages the use of data augmentation such as rotation, vertical and horizontal flipping, and zoom in to create a bigger dataset. However, many images were produced as the result of data augmentation from the low number of UAV-based bridge inspection images. Therefore, there are similarities in these images; by fitting the models a few hundred times, each time with a different realization of the same image, and then predicting a new image, CNN models are not able to correctly predict these new inputs. In other words, these issues also cause overfitting and lead to extremely high-variance predictions that can be seen when training accuracy reaches 100% and its accuracy in the validation and test datasets is much lower. To deal with these issues, and since low-level features are extracted in the last layers of networks, the last part of each CNN model was trained on the established training dataset. As a result,



during training, models were not saturated, and the pre-trained models were adopted for the crack detection in UAV-based bridge inspections.

### STEP 2.1-Selecting the CNN Models

Architecture engineering is a significant part of neural network development, starting from the seminal work of AlexNet [28], which is often regarded as the starting point of deep learning (DL). Since then, many improvements have been made in the development of neural networks such as ZFNet [29], VGG [30], Inception [31], ResNet [32], DenseNet [33], network in network (NiN) [34], wider residual network [35], networks with stochastic depth [36], Xception [37], fractural networks [38], squeeze networks [39], and so on. A survey of the recent architectures of deep convolutional NEURAL networks can be seen in [40]. These architectures are continuously developed manually by human experts, but recently the architecture engineering of neural networks is crossing into the area of dynamic architecture engineering. In this area, neural networks can build themselves from basic building blocks (different combinations of different layers such as a dense layer, convolutional layer, de-convolutional layer, max pooling, batch normalization, and so on) based on the number of input images and their resolution, mostly through reinforcement learning. NASNet [41], MnasNet [42], and EfficientNet [43] have used neural architecture search (NAS), which utilizes a recurrent neural network (RNN) as the controller to search variable-length architecture space to compose neural network architectures. Using dynamic architect engineering for building CNN models for crack detection is time-consuming and expensive; for instance, in NASNet a search through several convolutional cell candidates takes some days with the use of 500 GPUs, resulting in 2000 GPU-hours [41]. Therefore, the developed CNN models can be used via transfer learning for real problem solving such as crack detection in UA-based bridge inspections. The pre-trained CNN models, trained on more than 14 million images from ImageNet, that were chosen in this study are shown in Table 1 along with some of their specifics (the architecture of some of these CNN models is presented in Appendix A).

**Table 1.** Trainable, non-trainable, and total number of layers for each model.

Model	Total Parameters	Trainable Parameters	Non-Trainable Parameters	Total Number of Layers	FLOPs Billion
VGG16	14,714,688	11,799,040	2,915,648	19	30.7129
Resnet50	24,177,954	10,580,322	13,597,632	175	7.7511
Resnet152v2	58,921,890	10,574,178	48,347,712	564	21.8756
Xception	21,451,722	12,758,482	8,693,240	132	9.1340
InceptionV3	22,392,834	12,848,546	9,544,288	311	5.6933
DenseNet210	18,875,362	10,220,514	8,654,848	707	8.6316
NASNetLarge	86,079,220	11,216,802	74,862,418	1039	47.7934
EfficientNetB4	17,673,823	10,581,832	7,091,991	474	3.0688

As can be seen from Table 2, Resnet152v2 and NASNetLarge have more total number parameters than the other CNN models; in this study, these two CNN models are called heavy models and the rest are called non-heavy models. In this study, the CNN models were initially trained with the same number of trainable parameters and then the effects of transfer learning and the noises were evaluated to find the optimized amount of transfer learning. Moreover, floating point operations (FLOPs) is a measure of how many addition and multiplication operations are needed to run the CNN model. It can be considered as a proxy of CNN model speed regardless of the system specifics in which it is training. In general, with an increase in input shape, the number of input channels, and the number and size of convolution filters, the FLOPs will increase as well, but it has an inverse relation to the number of strides [32,44]. For instance, the first layer of VGG16, which has 64 filters of  $3 \times 3$  size with a stride and padding of 1 on an input image size of  $224 \times 224$  with three channels (RGB) requires almost 3.7 billion FLOPs, and in total VGG16 requires 30.7129 billion FLOPs, which is almost 9 times more than ResNet50 requires [32,44]. (see Table 1 for FLOPs comparison of CNN models).

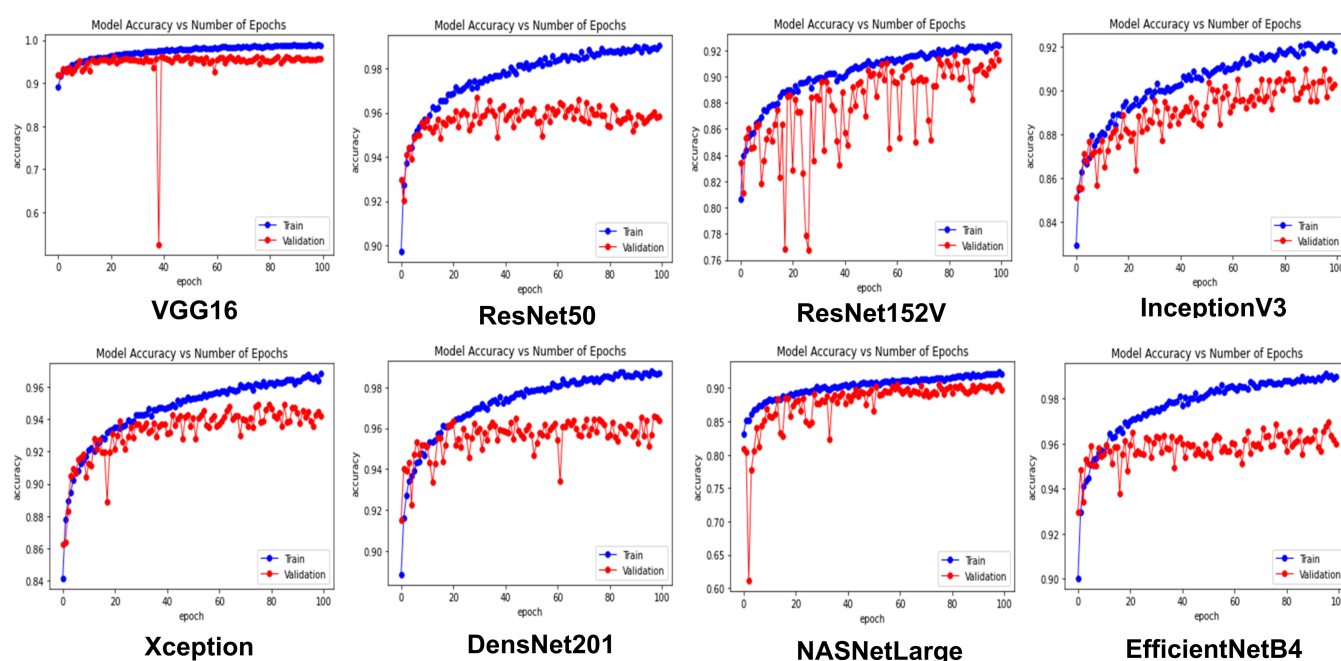
**Table 2.** Training and validation accuracy and training time of each model.

Model	Training Accuracy	Validation Accuracy	Training Time (s/100 Epochs)	Parameters	FLOPs Billion	Layers
VGG16	0.9881	0.9569	155,424	14,714,688	30.7129	19
Resnet50	0.9902	0.9585	71,700	24,177,954	7.7511	175
Resnet152v2	0.9120	0.8696	129,849	58,921,890	21.8756	564
Xception	0.968	0.9416	89,771	21,451,722	9.1340	132
InceptionV3	0.9181	0.9027	77,915	22,392,834	5.6933	311
DenseNet201	0.9867	0.9637	156,623	18,875,362	8.6316	707
NASNetLarge	0.9210	0.8975	337,903	86,079,220	47.7934	1039
EfficientNetB4	0.9901	0.96	107,984	17,673,823	3.0688	474

### STEP 2.2-Training the CNN Models

The basic notion is to keep trainable parameters for each model the same to evaluate firstly how CNN models are adopted for a specific task, in this case crack detection. Moreover, as the pre-trained models were based on ImageNet and were aimed to identify 1000 classes, in this study classification layers were adopted for the two classes (cracked and non-cracked) and classification layers were the same for all CNN models. Thus, the number of parameters was different from the original architectures due to adopting classification layers for the two classes. In addition, an Adam optimizer with the same parameters is used for all CNN models. All the CNN models were trained on a MacBook Pro 16-inch, with a 2.4 GHz 8 core 9th generation Intel Core i9 processor, 64 GB 2666 MHz DDR4 RAM, and an AMD Radeon Pro 5600 M graphics card with 8 GB of HBM2 memory.

Figure 6 shows the accuracy of the CNN models in the training and validation datasets in 100 epochs. The difference between training and validation accuracy was not significant in all CNN models and none of the models overfit or saturates during the training. The accuracy of CNN models on the validation dataset is a more reliable indicator of how the CNN models perform on unseen data. Looking at the accuracy of CNN models in the validation dataset, four CNN models with the highest accuracy in the training dataset also have the highest accuracy in the validation dataset, though in a different order. EfficientNetB4 and DenseNet210 have the highest accuracy, 96%, followed by ResNet50 and VGG16 with 95% accuracy. Interestingly the worst accuracies in the validation dataset belong to two heavy CNN models, ResNet152v2 and NASNetLarge.

**Figure 6.** Training and validation accuracy of each CNN model in 100 epochs.

Regardless of the available number of images for training, system specifics, CPU, and GPU power that CNN models train on, the number of parameters (trainable and non-trainable), FLOPs, and the number of layers are three key factors affecting the training time of CNN models. As can be seen from Table 2, the NASNetLarge is the slowest CNN model with the highest number of parameters, layers, and FLOPs. However, in the case of Resnet models (Resnet50 and Resnet152v2), it seems they are faster due to the presence of a skip connection that lets the gradient signal travel back directly to early layers via these skip connections. ResNet50 has the lowest training time with 71,700 s for 100 epochs of training and the second-fastest CNN model is Xception with 77,915 s.

#### STEP 2.2.1-The Performance of the CNN Models on the Sdnet2018 Test Dataset

The results confirm that the four CNN models (EfficientNetB4 and DenseNet210 with 96% accuracy and ResNet50 and VGG16 with 95% accuracy) with the highest validation accuracy also have the highest accuracy in the SDNET2018 test dataset (DenseNet210 with 97% accuracy and EfficientNetB4, ResNet50, and VGG16 with 95% accuracy). Figure 7 shows the confusion matrices for each CNN model; it can be seen that DensNet201 predicts 1040 non-crack images correctly out of 1054 non-crack images, and it correctly predicts 797 crack images out of 848 crack images, while it incorrectly predicts 51 crack images as non-crack images. More details about the performance of each CNN model on the SDNET2018 test dataset, such as the precision, recall, and F1 score of each model, can be seen in Appendix B.

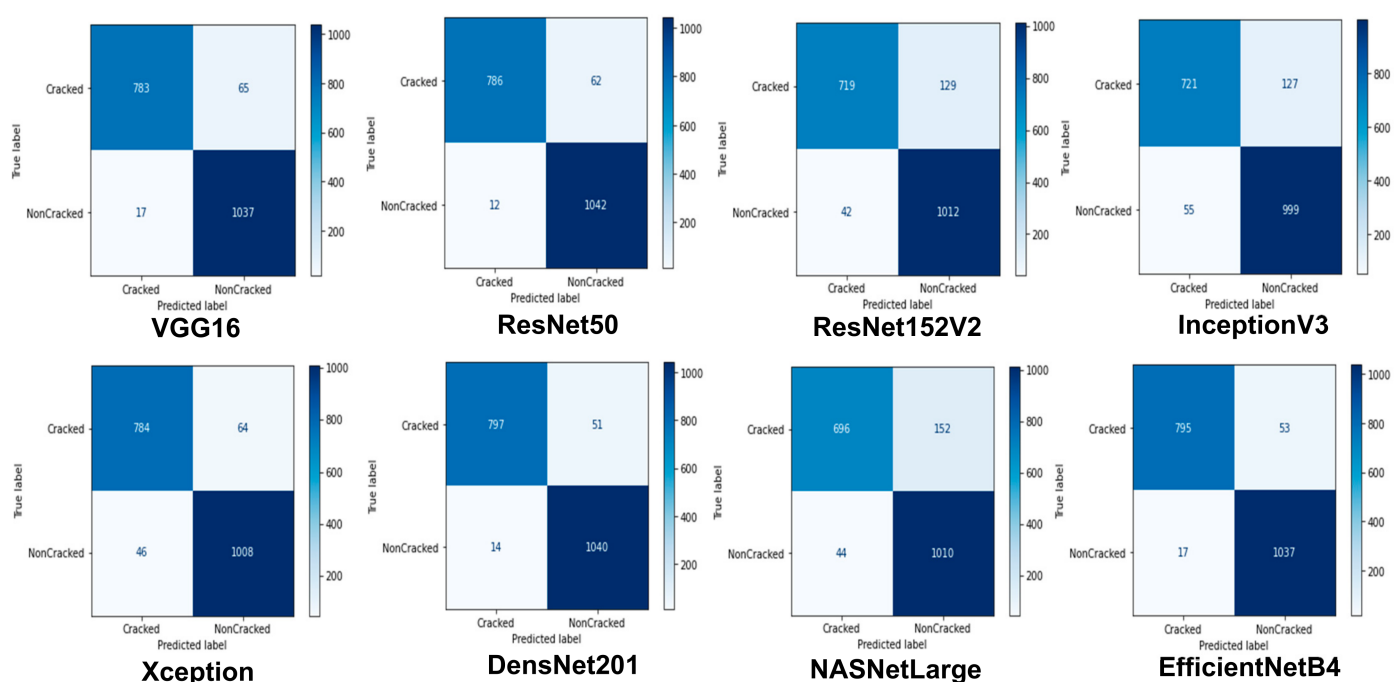


Figure 7. Confusion matrices of CNN models tested on SDNET2018 test dataset.

#### STEP 2.2.2-The Performance of the CNN Models on the UAV-Based Bridge Inspection Datasets

In this section, the performance of these CNN models will be examined in two UAV-bridge inspection datasets to see how they perform in practice. Table 3 summarizes the recall metric comparison of the CNN models on both of the UAV-based bridge inspection datasets. More details about the performance of each CNN model on both of the UAV-based bridge inspection datasets, such as the precision, recall, and F1 score of each model, can be seen in Appendix B.

**Table 3.** CNN models' recall for two UAV-based bridge inspection datasets.

Model	Heterogeneous UAV-Based Bridge Inspection Dataset			Homogeneous UAV-Based Bridge Inspection Dataset			% $\nabla$ Accuracy UAV-Based Bridge Inspection Datasets		
	Crack Recall	Non-Crack Recall	Average Recall	Crack Recall	Non-Crack Recall	Average Recall	% $\nabla$ Crack Recall	% $\nabla$ Non-Crack Recall	% $\nabla$ Average Recall
VGG16	0.90	0.49	0.69	0.85	0.52	0.68	−5%	3%	−1%
Resnet50	0.75	0.51	0.63	0.86	0.53	0.70	11%	2%	7%
Resnet152v2	0.66	0.73	0.69	0.58	0.81	0.69	−9%	8%	0%
Xception	0.88	0.33	0.60	0.80	0.69	0.74	−8%	36%	14%
InceptionV3	0.56	0.56	0.56	0.46	0.72	0.59	−10%	16%	3%
DenseNet201	0.75	0.39	0.57	0.88	0.42	0.65	13%	3%	8%
NASNetLarge	0.59	0.64	0.61	0.51	0.74	0.62	−8%	10%	1%
EfficientNetB4	0.75	0.27	0.51	0.81	0.35	0.58	6%	8%	7%

In the heterogeneous dataset, VGG16 and ResNet152v2 performed better in comparison to the other models, with an average accuracy of 69% followed by Resnet50 with 63% average accuracy and NASNetLarge with an average accuracy of 61%. As can be seen from Table 4, the CNN models' performance decreased substantially in the heterogeneous dataset compared to the SDNET2018 test dataset, an average accuracy decrease of 32.6% for all the CNN models. This is due to the heterogeneities existing in the UAV-based bridge inspection dataset. It is expected that the accuracy of the CNN models will be much higher on the homogeneous UAV-based bridge inspection dataset. However, the CNN models' performance does not increase substantially in this dataset compared to the heterogeneous dataset, with a 4.875% increase on average for all the CNN models.

**Table 4.** CNN models' accuracy for SDNET2018 test dataset and two UAV-based bridge inspection datasets.

Model	Training Accuracy	Validation Accuracy	SDNET 2018 Accuracy	UAV Heterogeneous Dataset Accuracy	UAV Homogenous Dataset Accuracy	Trainable Parameters	Non-Trainable Parameters	Total Number of Layers
VGG16	0.9881	0.9569	0.96	0.69	0.68	11,799,040	2,915,648	19
Resnet50	0.9902	0.9585	0.96	0.63	0.70	10,580,322	13,597,632	175
Resnet152v2	0.9120	0.8696	0.88	0.69	0.69	10,574,178	48,347,712	564
Xception	0.968	0.9416	0.94	0.60	0.74	12,758,482	8,693,240	132
InceptionV3	0.9181	0.9027	0.90	0.56	0.59	12,848,546	9,544,288	311
DenseNet201	0.9867	0.9637	0.97	0.57	0.65	11,216,802	74,862,418	707
NASNetLarge	0.9210	0.8975	0.90	0.61	0.62	10,220,514	8,654,848	1039
EfficientNetB4	0.9901	0.96	0.96	0.51	0.58	10,581,832	7,091,991	474

The best-performing CNN models on the heterogeneous UAV-based bridge inspection dataset were VGG16, ResNet152V2, and ResNet50 with 69%, 69%, and 63% accuracy, respectively. Also, the best-performing CNN models on the homogenous dataset were Xception, ResNet50, and ResNet152V2 with 74%, 70%, and 69% accuracy, respectively. Moreover, the best-performing CNN models on the SDNET2018 test dataset were DenseNet201 with 97% accuracy, and VGG16, ResNet50, and EfficientNetB4 with the same 96% accuracy. It can be realized from these results that in each specific dataset the performance of the CNN models was different; for example, DenseNet201, which outperformed the other CNN models in the SDNET2018 test dataset, was not successful in the UAV-based bridge inspection datasets, especially in the heterogeneous dataset. Therefore, regarding the datasets in this study deeper networks did not provide higher accuracy, and shallower networks like VGG16, ResNet50, and Xception performed better than heavier or deeper networks such as NASNetLarge with 1039 layers.

### STEP 2.3-Select the Best Model for Future Inspections

Due to lack of research in selecting the optimized amount of transfer learning, a well-established methodology for finding the optimized amount of transfer learning is missing. Therefore, we found the range of the optimized amount of transfer learning by trial and error as will be explained in the next section. After finding the optimized amount of transfer learning for the available training dataset and evaluating the performance of the CNN models, the best model can be selected for the crack detection task in future drone-based bridge inspections. At the end, the UAV-based bridge inspections will be added to establish

a training dataset so the next time the training dataset is richer, and more failure modes of the bridges can be identified. For example, in each inspection, failures other than cracks can be identified and classified, and therefore the CNN models can be trained to classify more and more failures.

### 3. Results

This study shows that, with the same number of trainable parameters regarding the available training dataset, the shallower networks like VGG16, ResNet50, and Xception perform better for the UAV-based bridge inspection datasets. To evaluate the effects of transfer learning and to find the optimized amount of transfer learning, more experiments were conducted, and the results are presented in the following sections.

#### 3.1. The Effects of Transfer Learning

It is not possible to have CNN models with the exact same number of trainable parameters because of their architectures. Therefore, it is not obvious if the difference between 2–3 million trainable parameters significantly affects the accuracy of CNN models. To explore such effects, the number of trainable parameters for ResNet50 was increased from 10,580,322 to 17,540,706 (65.8% increase).

Table 5 shows that ResNet50 with more than 65% more trainable parameters had a 0.0007 higher accuracy in training (0.07%) but 0.0053 lower accuracies (0.55%) in the validation. Thus, the effects of 65% more trainable parameters on training and validation accuracy in ResNet50 were negligible. The same was done for ResNet152v2 as a representative of the heavy CNN models; it can be seen from Table 5 that the effects are also negligible in this heavy model. Thus, it can be concluded with high confidence that changes in the range of three million trainable parameters are negligible for the training and validation accuracy of heavy and non-heavy models. Moreover, a lower number of trainable parameters means lower training time. Here, ResNet50 with 17,540,706 trainable parameters took 209.72 min longer to be trained. However, the change in the number of trainable parameters needs to be performed carefully and verified on the UAV-based inspection unseen dataset and test, which will be discussed in the following sections in more detail. The performance of these models was also evaluated on UAV-based datasets and the results are shown in Table 6.

Decreasing transfer learning increased the precision of ResNet50 significantly from 63% to 79% in the UAV-based bridge inspection heterogeneous dataset. Conversely, ResNet152V2 precision was decreased significantly by increasing the number of trainable parameters from 70% to 60%. Interestingly, this difference is not as significant in the UAV-based bridge inspection homogenous dataset. The precision of ResNet50 increased from 72% to 77%, while the ResNet152V2 precision decreased by only 2% from 70% to 68% compared to a 10% decrease in the heterogeneous dataset. Therefore, it cannot be concluded that by increasing the number of trainable parameters (without transfer learning) the accuracy of CNN models will increase. In the case of ResNet152V2, its performance decreases in both UAV-based bridge inspection datasets; therefore, it can be concluded that the transfer learning is relevant in UAV-based bridge inspection.

**Table 5.** Comparison of ResNet50 and ResNet152v2 with different numbers of trainable parameters.

Model	Training Accuracy	Validation Accuracy	Training Time (s/100 Epochs)	Trainable Parameters	Non-Trainable Parameters
Resnet50 <sub>10,580,322</sub>	0.9902	0.9585	71,700	10,580,322	13,597,632
Resnet50 <sub>17,540,706</sub>	0.9909	0.9532	84,283	17,540,706	6,637,248
% ▽	0.07%	−0.55%	17.54%	65.8%	51.2%
Resnet152V2 <sub>10,574,178</sub>	0.9120	0.8696	129,849	10,574,178	48,347,712
Resnet152V2 <sub>16,942,434</sub>	0.9235	0.9127	137,043	16,942,434	41,979,456
% ▽	1.26%	4.96%	5.54%	60%	13.17%



**Table 6.** ResNet50 and ResNet152V2 with different trainable parameter performances.

Model		UAV-Based Bridge Inspection Heterogeneous Dataset			UAV-Based Bridge Inspection Homogenous Dataset		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
Resnet50 <sub>10,580,322</sub>	Cracked	0.60	0.75	0.67	0.65	0.86	0.74
	Non-cracked	0.67	0.51	0.58	0.8	0.53	0.64
	Weighted avg	0.63	0.63	0.62	0.72	0.70	0.69
Resnet50 <sub>17,540,706</sub>	Cracked	0.83	0.71	0.76	0.78	0.76	0.77
	Non-cracked	0.75	0.86	0.80	0.76	0.79	0.77
	Weighted avg	0.79	0.78	0.78	0.77	0.77	0.77
Resnet152V2 <sub>10,574,178</sub>	Cracked	0.71	0.66	0.68	0.75	0.57	0.65
	Non-cracked	0.68	0.73	0.71	0.65	0.81	0.72
	Weighted avg	0.70	0.69	0.69	0.70	0.69	0.68
Resnet152V2 <sub>16,942,434</sub>	Cracked	0.60	0.58	0.59	0.72	0.56	0.63
	Non-cracked	0.59	0.62	0.61	0.64	0.79	0.70
	Weighted avg	0.60	0.60	0.60	0.68	0.67	0.67

### 3.2. The Effects of Transfer Learning and Heterogeneities

The performance of these CNN models regarding decreasing the transfer learning part by increasing the number of trainable parameters and decreasing the noises in UAV images can be evaluated. Decreasing the heterogeneities and noises in UAV images gave a 7% increase of accuracy in ResNet50 and there was no gain in accuracy for ResNet152v2, as shown in Table 7. On the other hand, decreasing the noises in UAV images did not increase the accuracy in ResNet50 with 17,540,706 million trainable parameters. There was a 7% increase in accuracy of ResNet152v2 with 16,942,434 million trainable parameters by decreasing the heterogeneities. The results confirm that ResNet152v2 as a representative of heavy CNN models and ResNet50 as a representative of non-heavy CNN models have completely different behavior regarding noises in the datasets and the number of trainable parameters.

**Table 7.** Comparison of ResNet50 and ResNet152v2 with different numbers of trainable parameters on different datasets.

Model	Training Accuracy	Validation Accuracy	SDNET 2018 Accuracy	Heterogeneous UAV-Based Bridge Inspection Accuracy	Homogenous UAV-Based Bridge Inspection Accuracy	% $\nabla$ Accuracy UAV-Based Bridge Inspection Datasets
Resnet50 <sub>10,580,322</sub>	0.9902	0.9585	0.96	0.63	0.70	7%
Resnet50 <sub>17,540,706</sub>	0.9909	0.9532	0.96	0.78	0.77	−1%
% $\nabla$	0.07%	0.55%	0%	15%	7%	
Resnet152V2 <sub>10,574,178</sub>	0.9120	0.8696	0.88	0.69	0.69	0%
Resnet152V2 <sub>16,942,434</sub>	0.9235	0.9127	0.91	0.60	0.67	7%
% $\nabla$	1.26%	4.96%	3%	−9%	−2%	

As was shown, the performance of ResNet152v2 deteriorated with the increasing number of trainable parameters, and this had a more negative effect when the dataset was heterogeneous, with a 9% decrease in the accuracy, and a less negative effect when the dataset was homogenous, with only a 2% decrease in accuracy. On the other hand, decreasing transfer learning weight had a positive effect in ResNet50 when the UAV-based bridge inspection dataset was heterogeneous, with a 15% increase in the accuracy, and a 7% increase in accuracy when the dataset was homogenous. Therefore, one might speculate that transfer learning is irrelevant in non-heavy CNN models. To test if transfer learning is relevant or not in non-heavy CNN, models two of these CNN models, Resnet50 and VGG16, were trained without using transfer learning on the training dataset, and the results are shown in Table 8.

**Table 8.** Comparison of ResNet50 and VGG16 with and without transfer learning on three datasets.

Model	Training Accuracy	Validation Accuracy	SDNET 2018 Accuracy	UAV-Based bridge Inspection Heterogeneous Accuracy	UAV-Based Bridge Inspection Homogenous Accuracy	Training Time (s/100 Epochs)
<b>Resnet50</b> 10,580,322	0.9902	0.9585	0.96	0.63	0.70	71,700
<b>Resnet50</b> 17,540,706	0.9909	0.9532	0.96	0.78	0.77	84,283
<b>Resnet50</b> No transfer learning	0.9702	0.9574	0.95	0.57	0.62	218,990
% $\nabla$ 17,540,706-No transfer learning	−2.07%	0.42%	−1%	−21%	−15%	−134,707
<b>VGG16</b> 11,799,040	0.9881	0.9569	0.96	0.69	0.68	155,424
<b>VGG16</b> No transfer learning	0.9723	0.9485	0.94	0.81	0.83	351,676
% $\nabla$ 11,799,040-No transfer	−1.58%	−0.84%	−2%	12%	15%	−196,252

The results show that transfer learning is very important for UAV-based bridge inspection datasets in Resnet50 as its accuracy drops significantly by 21% and 15% in the heterogeneous and homogenous datasets, respectively. Therefore, it can be concluded that for CNN models with similar parameters to Resnet50, transfer learning improves their performance. On the other hand, VGG16 has only 19 layers, the shallowest CNN model, and transfer learning does not improve its performance and its accuracy improves 12% and 15% in the heterogeneous and homogenous datasets, respectively. Also, it is obvious in transfer learning that the higher the training parameters, the higher the training time; for instance, without transfer learning it takes 134.707 s (37.42 h) more time in Resnet50 and 196.252 s (54.51 h) more time in VGG16 training.

Due to the probabilistic nature of the CNN models' prediction, to see if there was significant variance, the predictions of the CNN models were repeated 10 times; it was confirmed that there were no such meaningful variances in the CNN models' predictions. For instance, the variance in prediction accuracy of Resnet152v2 in one of the UAV-bridge inspection datasets was 0.000068 which is not significant (for more details, see Appendix C).

### 3.3. Selecting the Optimized Amount of Transfer Learning

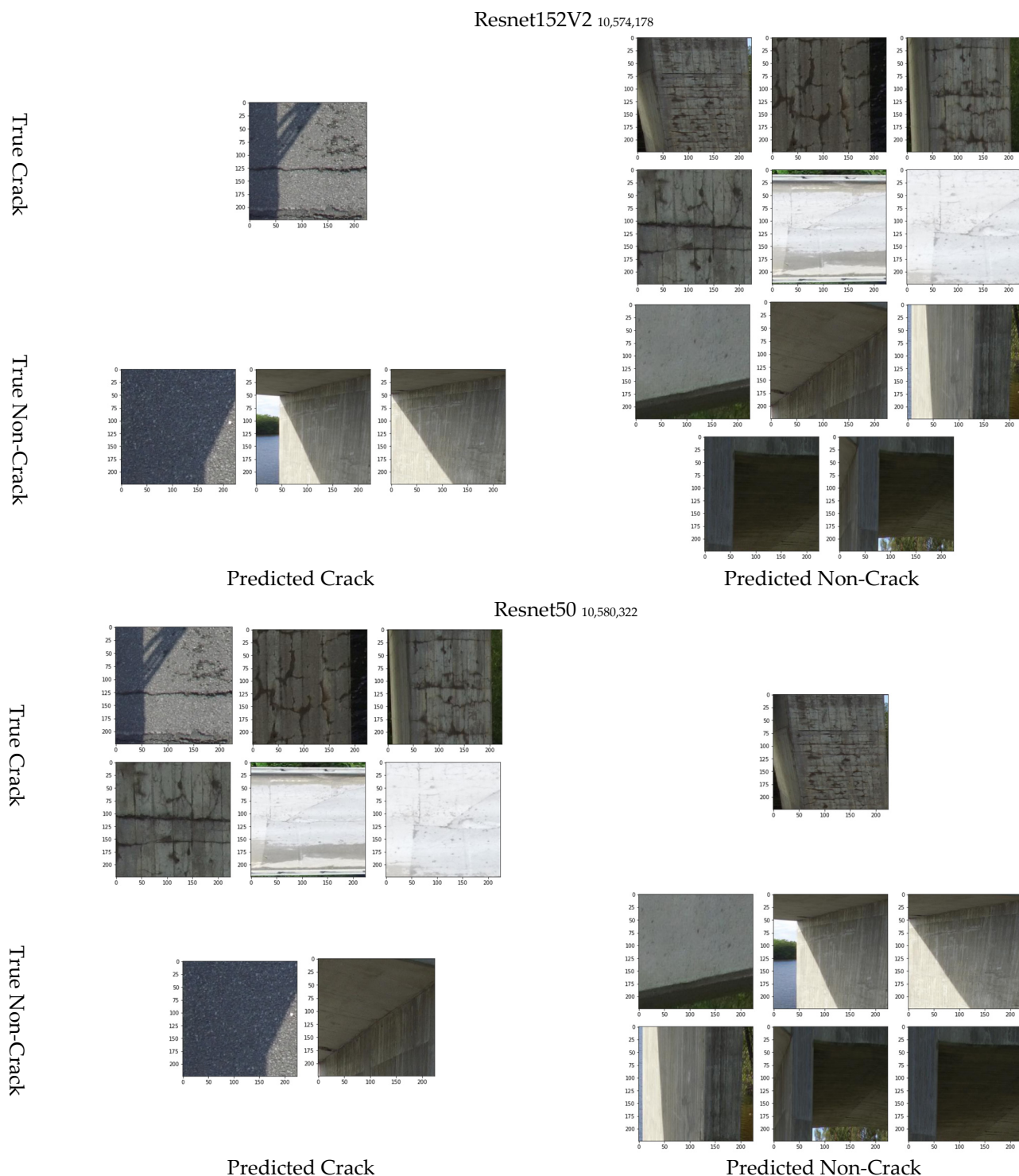
As was shown for Resnet152V2, going from 10,574,178 to 16,942,434 trainable parameters decreased its accuracy on average by 5.5% in both UAV-based bridge inspection datasets. Thus, the optimized number of trainable parameters for Resnet152V2 would be equal to or less than 10,574,178. For Resnet50, increasing the number of trainable parameters from 10,580,322 to 17,540,706 increased the accuracy by 11% in both UAV-based bridge inspection datasets. Further, increasing the number of its trainable parameters to 24,124,770, its accuracy decreased by 18% in both UAV-based bridge inspection datasets. Thus, the optimized number of trainable parameters for Resnet50 is between 17,540,706 and 24,124,770. On the other hand, for VGG16, the optimized number of trainable parameters is 14,862,498 without transfer learning.

The maximum performance of VGG16 is 81% and 83% in the two UAV datasets with no transfer learning. However, in the case of Resnet50, its highest performance was at between 17,540,706 and 24,124,770 trainable parameters. Thus, Resnet50 and other CNN models whose trainable parameters are not at the maximum possible number can reach at least the same performance as VGG without transfer learning. Therefore, it can be concluded, for small training datasets (19,023 in this study), the shallowest CNN model, VGG16, without transfer learning performs as well as Resnet50 with transfer learning, with 81% and 83% accuracy in the UAV-based bridge inspection heterogeneous and homogenous datasets, respectively.

### 3.4. Visualizing the Performance of the CNN Models

Examples of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) images for Resnet50 10,580,322 and Resnet152V2 10,574,178 models are shown in Figure 8 (for the rest of CNN models in this study see Appendix A2). As can be seen, Resnet152V2 incorrectly predicted six crack images as non-crack images while Resnet50 correctly identified five of them as a crack and was only unable to identify one of the crack images as it included edges and dark and light backgrounds; these kinds of images cannot be handled by most of the CNN models. Moreover, it seems that Resnet152V2

identifies shadows as cracks, as it incorrectly identified three images with shadows as crack images while they are non-crack elements, while Resnet50 identified two of them correctly. It also can be seen that Resnet50 incorrectly identified a 90° edge without shadows as a crack, but the existence of some lines in conjunction with some parts of the concrete might be the reason.



A feature map, or activation map, is the output activations for a given filter, as going further in the network reduces the size of filters; therefore, more details are captured from the input. The first feature map could, for example, look for curves, then the next feature map could look at a combination of curves that build circles, and the next feature map could detect an object from lines and circle features.

Table 9 shows the feature maps of CNN models from the first, middle, and last parts of the network. For example, DensNet201 has 707 layers (including all layers such as concatenation, ReLU, batch normalization, and convolutional): 64 feature maps of layer 67 (conv3-block2-concatenation) are shown as the first part, 64 feature maps of layer 477 (conv4-block48-concatenation) are shown as the middle part, and finally, 64 feature maps of layer 705 (conv5-block32-concatenation) are shown as the last part of the network. This image includes most of the heterogeneities, like the presence of edges, different materials, and dark and light backgrounds, and most of the CNN models were not able to handle this many heterogeneities and incorrectly classified it.

**Table 9.** Feature maps of first, middle, and last parts of each model.















































































Input		Feature Maps (First Part of Networks)	Feature Maps (Middle Part of Networks)	Feature Maps (Last Part of Networks)
VGG16				
				
				
				
Resnet50				
				
				
				
Resnet152V2				
				
				
				
Xception				
				
				
				



Table 9. Cont.

Input		Feature Maps (First Part of Networks)	Feature Maps (Middle Part of Networks)	Feature Maps (Last Part of Networks)
InceptionV3				
				
				
				
DenseNet210				
				
				
				

#### 4. Discussion

This study can be categorized in the transfer learning from pre-trained models by fine tuning in homogenous transfer learning methods based on the categorization of Weiss et al. [45]. As the source domain, trained on ImageNet for object detection, which is target and are similar as this study for crack detection. In this category, research in a wide range of tasks has achieved high accuracy using transfer learning. For example, Prajapati et al. used 251 images to train a VGG16 pretrained model for dental disease classification and they achieved an accuracy of 88.46% [46]. In related research, a dataset consisting of 582 crack images and 458 non-crack images was used by Kucuksubasi and Sorguch to train an InceptionV3 pretrained model in GPS-denied environments for autonomous UAVs crack detection [47]. Their results confirm that InceptionV3 can reach 97.382% accuracy with an unseen test dataset. These results agree with this study's results, as it was shown for small datasets (less than 20,000) that non-heavy models have better performance by finding the optimized amount of transfer learning. In these studies, the importance of transfer learning is confirmed for each specific task (as per the first research question in this study proposed in Section 1) as follows:

- How can drone-based bridge inspection benefit from the CNN models via transfer learning?
- How much do heterogeneities and noises between training and testing datasets affect CNN models via transfer learning?
- How much transfer learning must be used?

More than, showing the usefulness of transfer learning in drone-based bridge inspection. This study was an attempt to evaluate the effects of heterogeneities and noises between training and testing datasets and find the exact amount of transfer learning required regarding each specific task and the available datasets. Therefore, by having different noises and available training datasets, more experiments can help to find the thresholds in which shallow CNN models outperform the deeper CNN models to select the most suitable CNN model for real applications.

#### 5. Conclusions

The lack of UAV-based bridge inspection images makes transfer learning important in automatic failure detection as a key element of civil infrastructure. It was shown that there is a relationship between the number of available training images for each task and the optimized amount of transfer learning. Based on the available 19,023 datasets in this



study, the optimized range of transfer learning for some CNN models was achieved. For example, for Resnet152V2, it would be equal to or less than 10,574,178, and for Resnet50 it is between 17,540,706 and 24,124,770. VGG16, with only 19 layers, performs better without transfer learning. By choosing the optimized amount of transfer learning, at least 81% accuracy is achievable via transfer learning in non-heavy models such as Resnet or InceptionV3. On the other hand, the results confirm that the noises inherent to UAV-based bridge inspection significantly affect their performance, with an average accuracy decrease of 32.6% in all CNN models. With a simple cropping strategy on UAV-based bridge inspection images with minimum effort and without preprocessing images, CNN models can identify cracks element from non-crack elements with 81% accuracy. Moreover, with a better cropping strategy and a little more effort, CNN models can identify cracks elements from non-crack elements with 83% accuracy. Therefore, by simply choosing the optimized amount of transfer learning, the advantages of CNN can be utilized with minimum effort to reduce human involvement and increase the accuracy of conventional visual inspections of bridges. An automated cropping strategy combined with CNNs' transfer learning models enables UAV-based bridge inspections to be partially automated. As the number of images increases, they can be used in training CNN models to boost their performance and include more bridge failure modes.

**Author Contributions:** Conceptualization, M.A.; data curation, M.A. and Y.Z.A.; formal analysis, M.A., E.L.D. and Y.Z.A.; funding acquisition, M.A. and Y.Z.A.; investigation, M.A., E.L.D. and Y.Z.A.; methodology, M.A., E.L.D. and Y.Z.A.; project administration, E.L.D. and Y.Z.A.; resources, Y.Z.A.; software, E.L.D.; supervision, E.L.D. and Y.Z.A.; validation, M.A., E.L.D. and Y.Z.A.; visualization, M.A.; writing—original draft, M.A.; writing—review and editing, M.A., E.L.D. and Y.Z.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study received on external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

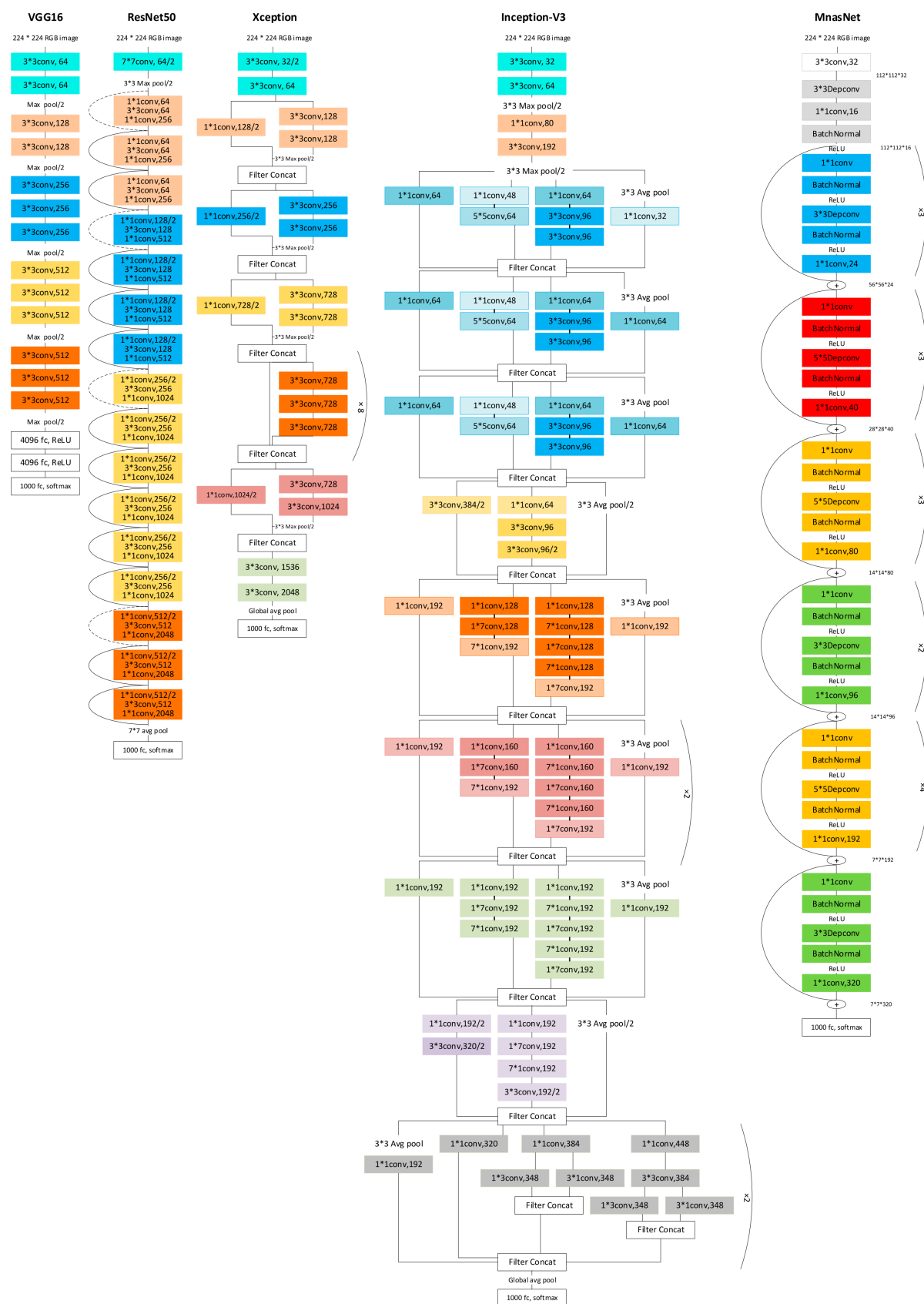


Figure A1. Some CNN Models' Architecture.

## Appendix B

**Table A1.** Performance Metrics for CNN Models on SDNET2018 Test Data Set.

Model		Precision	Recall	F1-Score	Support
<b>VGG16</b>	Cracked	0.98	0.92	0.95	848
	Non-cracked	0.94	0.98	0.96	1054
	Macro avg	0.96	0.96	0.96	1902
	Weighted avg	0.96	0.96	0.96	1902
<b>Resnet50</b>	Cracked	0.98	0.93	0.96	848
	Non-cracked	0.94	0.99	0.97	1054
	Macro avg	0.96	0.96	0.96	1902
	Weighted avg	0.96	0.96	0.96	1902
<b>Resnet152v2</b>	Cracked	0.97	0.75	0.85	848
	Non-cracked	0.83	0.98	0.9	1054
	Macro avg	0.9	0.87	0.87	1902
	Weighted avg	0.89	0.88	0.87	1902
<b>Xception</b>	Cracked	0.94	0.92	0.93	848
	Non-cracked	0.94	0.96	0.95	1054
	Macro avg	0.94	0.94	0.94	1902
	Weighted avg	0.94	0.94	0.94	1902
<b>InceptionV3</b>	Cracked	0.93	0.85	0.89	848
	Non-cracked	0.89	0.95	0.92	1054
	Macro avg	0.91	0.9	0.9	1902
	Weighted avg	0.91	0.9	0.9	1902
<b>DenseNet201</b>	Cracked	0.98	0.94	0.96	848
	Non-cracked	0.95	0.99	0.97	1054
	Macro avg	0.97	0.96	0.97	1902
	Weighted avg	0.97	0.97	0.97	1902
<b>NASNetLarge</b>	Cracked	0.94	0.82	0.88	848
	Non-cracked	0.87	0.96	0.91	1054
	Macro avg	0.9	0.89	0.89	1902
	Weighted avg	0.9	0.9	0.9	1902
<b>EfficientNetB4</b>	Cracked	0.98	0.94	0.96	848
	Non-cracked	0.95	0.98	0.97	1054
	Macro avg	0.97	0.96	0.96	1902
	Weighted avg	0.96	0.96	0.96	1902

**Table A2.** Performance Metrics for CNN Models on UAV-Based Bridge Inspection Heterogeneous Dataset.

Model		Precision	Recall	F1-Score	Support
<b>VGG16</b>	Cracked	0.64	0.9	0.75	154
	Non-cracked	0.83	0.49	0.61	154
	Macro avg	0.74	0.69	0.68	308
	Weighted avg	0.74	0.69	0.68	308
<b>Resnet50</b>	Cracked	0.6	0.75	0.67	154
	Non-cracked	0.67	0.51	0.58	154
	Macro avg	0.63	0.63	0.62	308
	Weighted avg	0.63	0.63	0.62	308
<b>Resnet152v2</b>	Cracked	0.71	0.66	0.68	154
	Non-cracked	0.68	0.73	0.71	154
	Macro avg	0.70	0.69	0.69	308
	Weighted avg	0.70	0.69	0.69	308

Table A2. Cont.

Model		Precision	Recall	F1-Score	Support
<b>Xception</b>	Cracked	0.57	0.88	0.69	154
	Non-cracked	0.73	0.33	0.46	154
	Macro avg	0.65	0.60	0.57	308
	Weighted avg	0.65	0.60	0.57	308
<b>InceptionV3</b>	Cracked	0.56	0.56	0.56	154
	Non-cracked	0.56	0.56	0.56	154
	Macro avg	0.56	0.56	0.56	308
	Weighted avg	0.56	0.56	0.56	308
<b>DenseNet201</b>	Cracked	0.55	0.75	0.64	154
	Non-cracked	0.61	0.39	0.48	154
	Macro avg	0.58	0.57	0.56	308
	Weighted avg	0.58	0.57	0.56	308
<b>NASNetLarge</b>	Cracked	0.62	0.59	0.60	154
	Non-cracked	0.61	0.64	0.62	154
	Macro avg	0.61	0.61	0.61	308
	Weighted avg	0.61	0.61	0.61	308
<b>EfficientNetB4</b>	Cracked	0.51	0.75	0.61	154
	Non-cracked	0.53	0.27	0.36	154
	Macro avg	0.52	0.51	0.48	308
	Weighted avg	0.52	0.51	0.48	308

Table A3. Performance Metrics for CNN Models on UAV-Based Bridge Inspection Homogenous Dataset.

Model		Precision	Recall	F1-Score	Support
<b>VGG16</b>	Cracked	0.64	0.85	0.73	200
	Non-cracked	0.77	0.52	0.62	200
	Macro avg	0.71	0.68	0.67	400
	Weighted avg	0.71	0.68	0.67	400
<b>Resnet50</b>	Cracked	0.65	0.86	0.74	200
	Non-cracked	0.80	0.53	0.64	200
	Macro avg	0.72	0.70	0.69	400
	Weighted avg	0.72	0.70	0.69	400
<b>Resnet152v2</b>	Cracked	0.75	0.57	0.65	200
	Non-cracked	0.65	0.81	0.72	200
	Macro avg	0.70	0.69	0.68	400
	Weighted avg	0.70	0.69	0.68	400
<b>Xception</b>	Cracked	0.72	0.8	0.76	200
	Non-cracked	0.78	0.69	0.73	200
	Macro avg	0.75	0.74	0.74	400
	Weighted avg	0.75	0.74	0.74	400
<b>InceptionV3</b>	Cracked	0.62	0.46	0.53	200
	Non-cracked	0.57	0.72	0.64	200
	Macro avg	0.60	0.59	0.58	400
	Weighted avg	0.60	0.59	0.58	400
<b>DenseNet201</b>	Cracked	0.6	0.88	0.72	200
	Non-cracked	0.78	0.42	0.55	200
	Macro avg	0.69	0.65	0.63	400
	Weighted avg	0.69	0.65	0.63	400
<b>NASNetLarge</b>	Cracked	0.66	0.51	0.57	200
	Non-cracked	0.6	0.74	0.66	200
	Macro avg	0.63	0.62	0.62	400
	Weighted avg	0.63	0.62	0.62	400

Table A3. Cont.

Model		Precision	Recall	F1-Score	Support
EfficientNetB4	Cracked	0.55	0.81	0.66	200
	Non-cracked	0.65	0.35	0.45	200
	Macro avg	0.60	0.58	0.56	400
	Weighted avg	0.60	0.58	0.56	400

## Appendix C

Table A4. Standard Deviations and Variance of One CNN Model in 10 Predictions.

	Prediction	Precision	Recall	f1-Score
1	Crack	0.75	0.58	0.66
	Non-crack	0.66	0.81	0.73
	Accuracy			0.69
	Weighted avg	0.71	0.69	0.69
2	Crack	0.73	0.57	0.64
	Non-crack	0.65	0.79	0.71
	Accuracy			0.68
	Weighted avg	0.69	0.68	0.68
3	Crack	0.76	0.56	0.65
	Non-crack	0.65	0.82	0.73
	Accuracy			0.69
	Weighted avg	0.71	0.69	0.69
4	Crack	0.75	0.57	0.65
	Non-crack	0.66	0.81	0.72
	Accuracy			0.69
	Weighted avg	0.7	0.69	0.69
5	Crack	0.76	0.56	0.65
	Non-crack	0.65	0.82	0.73
	Accuracy			0.69
	Weighted avg	0.71	0.69	0.69
6	Crack	0.72	0.54	0.62
	Non-crack	0.63	0.79	0.7
	Accuracy			0.67
	Weighted avg	0.68	0.67	0.66
7	Crack	0.76	0.56	0.65
	Non-crack	0.65	0.82	0.73
	Accuracy			0.69
	Weighted avg	0.71	0.69	0.69
8	Crack	0.76	0.58	0.66
	Non-crack	0.66	0.81	0.73
	Accuracy			0.7
	Weighted avg	0.71	0.7	0.7
9	Crack	0.75	0.56	0.64
	Non-crack	0.65	0.81	0.72
	Accuracy			0.69
	Weighted avg	0.7	0.69	0.68
10	Crack	0.75	0.55	0.63
	Non-crack	0.64	0.82	0.72
	Accuracy			0.68
	Weighted avg	0.7	0.68	0.68
Standard deviations	crack	0.013703	0.012517	0.012693
	non crack	0.009428	0.011547	0.010328



Table A4. Cont.

Prediction		Precision	Recall	f1-Score
Variance	crack	0.000188	0.000157	0.000161
	non crack	0.000089	0.000133	0.000107
Standard deviations	Weighted avg	0.010328	0.008233	0.010801
Variance	Accuracy			0.008233
	Weighted avg	0.000107	0.000068	0.000117

## Appendix D

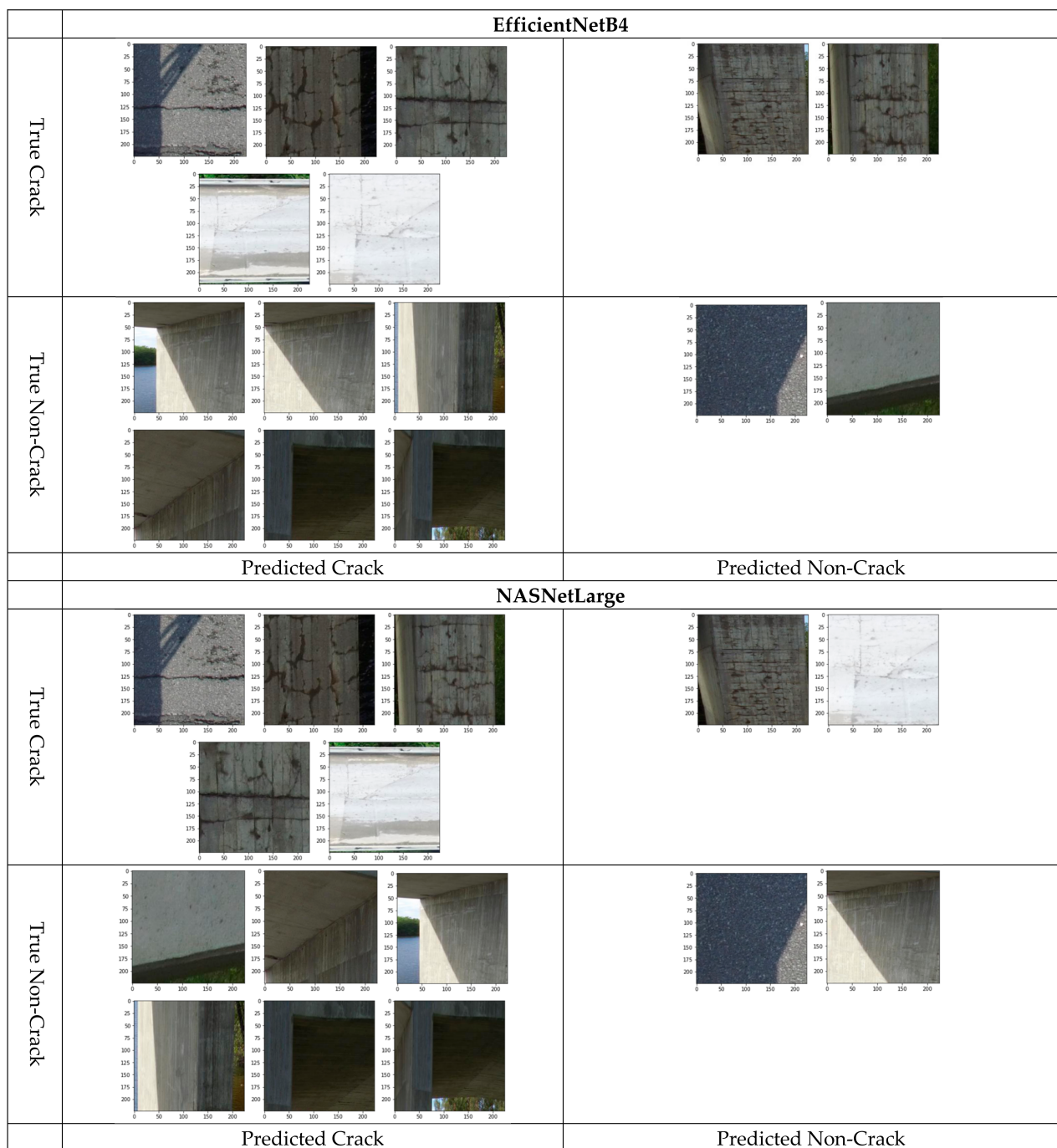


Figure A2. Cont.

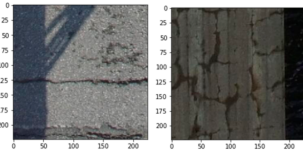
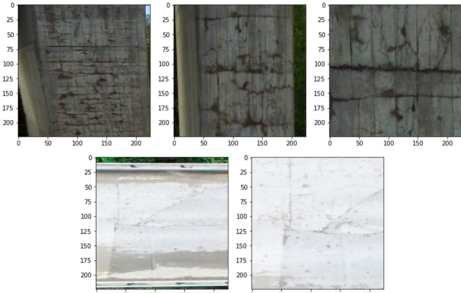
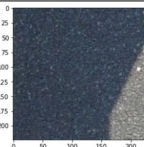
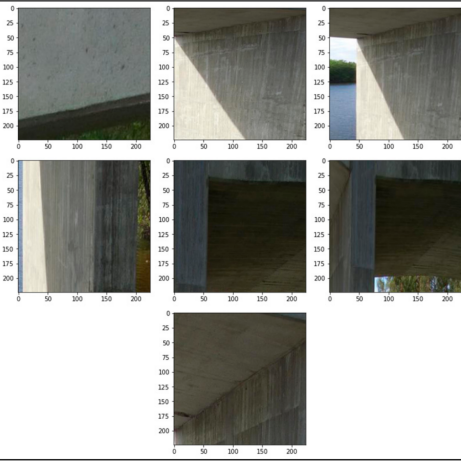
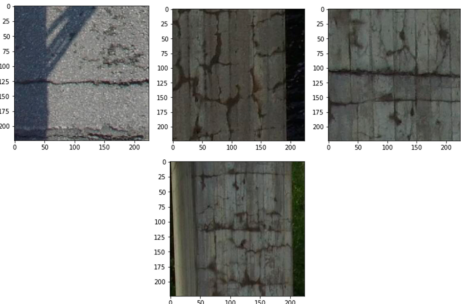
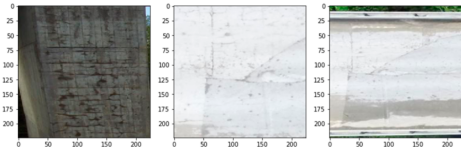

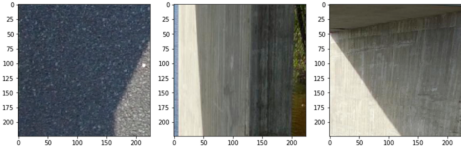
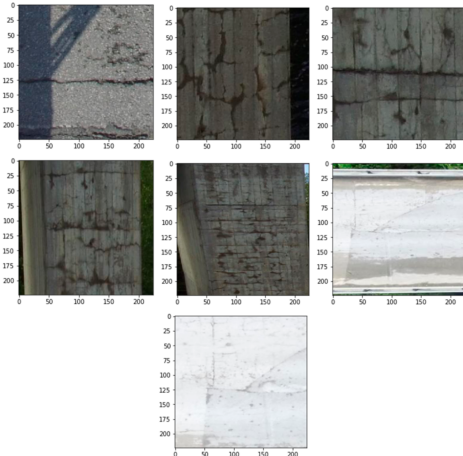
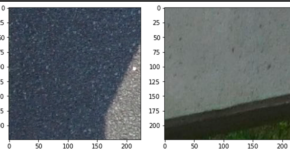
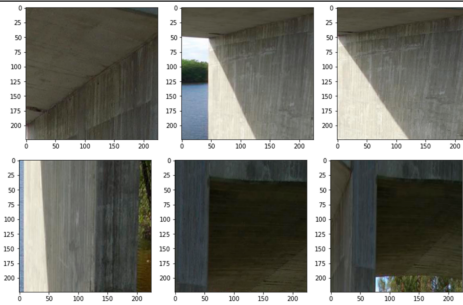
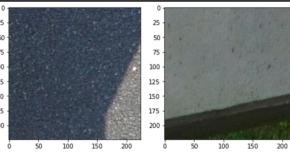
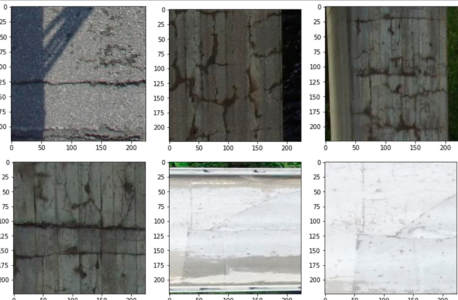
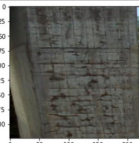
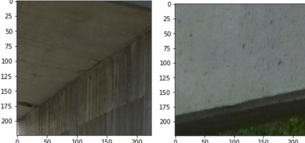
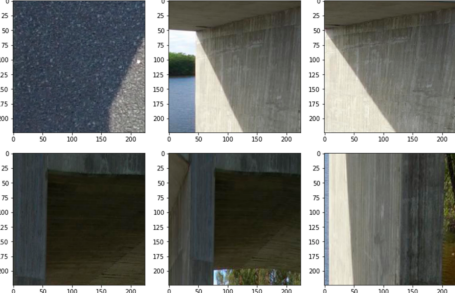
	DenseNet201	
True Crack		
True Non-Crack		
	Predicted Crack	Predicted Non-Crack
	Xception	
True Crack		
True Non-Crack		
	Predicted Crack	Predicted Non-Crack

Figure A2. Cont.

		InceptionV3	
True Crack			
True Non-Crack			
	Predicted Crack	Predicted Non-Crack	
VGG16			
True Crack			
True Non-Crack			
	Predicted Crack	Predicted Non-Crack	

**Figure A2.** Examples of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) Images for CNN Models.

## References

1. Ayele, Y. Drones for inspecting aging bridges. In Proceedings of the International Conference on Natural Hazards and Infrastructure, Chania, Crete Island, Greece, 23–26 June 2019; pp. 23–26.
2. Ayele, Y.Z.; Droguett, E.L. Application of UAVs for bridge inspection and resilience assessment. In Proceedings of the 29th European Safety and Reliability Conference, Hannover, Germany, 22–26 September 2019; pp. 22–26.

3. Puri, A. *A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance*; Department of Computer Science and Engineering, University of South Florida: Tampa, FL, USA, 2005; pp. 1–29.
4. Haddad, C.C.; Gertler, J. *Homeland Security: Unmanned Aerial Vehicles and Border Surveillance*; Congressional Research Service: Washington, DC, USA, 2010.
5. Aliyari, M.; Ashrafi, B.; Ayele, Y.Z. Hazards identification and risk assessment for UAV-assisted bridge inspections. *Struct. Infrastruct. Eng.* **2020**, 1–17. [\[CrossRef\]](#)
6. Metni, N.; Hamel, T. A UAV for bridge inspection: Visual servoing control law with orientation limits. *Autom. Constr.* **2007**, 17, 3–10. [\[CrossRef\]](#)
7. Chen, S.; Laefer, D.F.; Mangina, E.; Zolanvari, S.I.; Byrne, J. UAV bridge inspection through evaluated 3D reconstructions. *J. Bridge Eng.* **2019**, 24, 05019001. [\[CrossRef\]](#)
8. Achuthan, K.; Hay, N.; Aliyari, M.; Ayele, Y.Z. A Digital Information Model Framework for UAS-Enabled Bridge Inspection. *Energies* **2021**, 14, 6017. [\[CrossRef\]](#)
9. Landstrom, A.; Thurley, M.J. Morphology-based crack detection for steel slabs. *IEEE J. Sel. Top. Signal Process.* **2012**, 6, 866–875. [\[CrossRef\]](#)
10. Gehri, N.; Mata-Falcón, J.; Kaufmann, W. Automated crack detection and measurement based on digital image correlation. *Constr. Build. Mater.* **2020**, 256, 119383. [\[CrossRef\]](#)
11. Nigam, R.; Singh, S.K. Crack detection in a beam using wavelet transform and photographic measurements. *Structures* **2020**, 25, 436–447. [\[CrossRef\]](#)
12. Shi, Y.; Cui, L.; Qi, Z.; Meng, F.; Chen, Z. Automatic road crack detection using random structured forests. *IEEE Trans. Intell. Transp. Syst.* **2016**, 17, 3434–3445. [\[CrossRef\]](#)
13. Peng, L.; Chao, W.; Shuangmiao, L.; Baocai, F. Research on crack detection method of airport runway based on twice-threshold segmentation. In Proceedings of the 2015 5th International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC), Qinhuaingdao, China, 18–20 September 2015; pp. 1716–1720.
14. Fujita, Y.; Mitani, Y.; Hamamoto, Y. A method for crack detection on a concrete structure. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20–24 August 2006; IEEE: Piscataway, NJ, USA, 2006; Volume 3, pp. 901–904.
15. Mandal, V.; Uong, L.; Adu-Gyamfi, Y. Automated road crack detection using deep convolutional neural networks. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 5212–5215.
16. 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), Hague, The Netherlands, 6–9 October 2013.
17. Li, S.; Zhao, X. Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Adv. Civ. Eng.* **2019**, 2019. [\[CrossRef\]](#)
18. Mei, Q.; Gül, M.; Azim, M.R. Densely connected deep neural network considering connectivity of pixels for automatic crack detection. *Autom. Constr.* **2020**, 110, 103018. [\[CrossRef\]](#)
19. Liu, Y.; Yao, J.; Lu, X.; Xie, R.; Li, L. DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing* **2019**, 338, 139–153. [\[CrossRef\]](#)
20. Ayele, Y.Z.; Aliyari, M.; Griffiths, D.; Droguett, E.L. Automatic Crack Segmentation for UAV-Assisted Bridge Inspection. *Energies* **2020**, 13, 6250. [\[CrossRef\]](#)
21. Huang, Z.; Pan, Z.; Lei, B. Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data. *Remote Sens.* **2017**, 9, 907. [\[CrossRef\]](#)
22. Kaya, A.; Keceli, A.S.; Catal, C.; Yalic, H.Y.; Tekinerdogan, B. Analysis of transfer learning for deep neural network based plant classification models. *Comput. Electron. Agric.* **2019**, 158, 20–29. [\[CrossRef\]](#)
23. Sevakula, R.K.; Singh, V.; Verma, N.K.; Kumar, C.; Cui, Y. Transfer learning for molecular cancer classification using deep neural networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2018**, 16, 2089–2100. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Gopalakrishnan, K.; Khaitan, S.K.; Choudhary, A.; Agrawal, A. Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. *Constr. Build. Mater.* **2017**, 157, 322–330. [\[CrossRef\]](#)
25. Dung, C.V.; Anh, L. Autonomous concrete crack detection using deep fully convolutional neural network. *Autom. Constr.* **2019**, 99, 52–58. [\[CrossRef\]](#)
26. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khola, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, 115, 211–252. [\[CrossRef\]](#)
27. Dorafshan, S.; Thomas, R.; Maguire, M. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data Brief* **2018**, 21. [\[CrossRef\]](#) [\[PubMed\]](#)
28. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, 25, 1097–1105. [\[CrossRef\]](#)
29. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *Computer Vision—ECCV 2014, Proceedings of the 13th European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 818–833.
30. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.



31. 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.
32. 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, 27–30 June 2016; pp. 770–778.
33. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
34. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
35. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146.
36. Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; Weinberger, K.Q. Deep networks with stochastic depth. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 646–661.
37. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
38. Larsson, G.; Maire, M.; Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. *arXiv* **2016**, arXiv:1605.07648.
39. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
40. Khan, A.; Sohail, A.; Zahoor, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [[CrossRef](#)]
41. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710.
42. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2820–2828.
43. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. *Proc. Mach. Learn. PMLR* **2019**, *97*, 6105–6114.
44. Kundu, S.; Nazemi, M.; Pedram, M.; Chugg, K.M.; Beerel, P.A. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Trans. Comput.* **2020**, *69*, 1045–1058. [[CrossRef](#)]
45. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [[CrossRef](#)]
46. Prajapati, A.S.; Nagaraj, R.; Mitra, S. Classification of dental diseases using CNN and transfer learning. In Proceedings of the 2017 5th International Symposium on Computational and Business Intelligence (ISCBI), Dubai, United Arab Emirates, 11–14 August 2017; IEEE: Piscataway, NJ, USA, 2017.
47. Kucuksubasi, F.; Sorguc, A. Transfer learning-based crack detection by autonomous UAVs. *arXiv* **2018**, arXiv:1807.11785.