Convolutional neural networks for the crack diagnostics in concrete structures

Vladyslav Danishevskyy1*, Anastasia Gaidar2, and Bernd Markert3

1 Prydniprovska State Academy of Civil Engineering and Architecture, Structural and Theoretical Mechanics and Strength of Materials Dept., 49005 Architect Oleh Petrov Str. 24-a, Dnipro, Ukraine
2 Prydniprovska State Academy of Civil Engineering and Architecture, Construction Technologies Dept., 49005 Architect Oleh Petrov Str. 24-a, Dnipro, Ukraine
3 Institute of General Mechanics, RWTH Aachen University, 18 Eilfschornsteinstraße, 52062 Aachen, Germany

Abstract. New models of artificial neural networks are proposed for the identification and classification of cracks in concrete and reinforced concrete walls. The cloud tool Teachable Machine is used to develop a neural network with a pre-defined internal architecture. TensorFlow libraries allow us to develop a convolutional neural network with a tuneable architecture. The program code is written in Python and the learning was performed using the cloud environment Colab. The rational magnitudes of the learning parameters and the topology of the convolutional neural network are determined allowing to achieve the highest accuracy and the lowest losses of the model. The obtained results show a high efficiency of the artificial intelligence to solve problems of the health monitoring of building structures. The proposed models allow real-time automatic diagnostics by analysing photographs, images from smartphone or quadcopter webcams. The latter makes it possible to inspect buildings without the physical presence of humans at the site, which is especially important for working in dangerous places, such as tall buildings, partially destroyed buildings, mined areas, etc. The proposed methods can be further extended for the monitoring and classification of a wide range of defects in building structures.

1 Introduction

Millions of dollars are spent annually in the world on technical diagnostics of buildings and structures. Natural disasters such as floods and earthquakes, along with numerous negative man-made impacts lead to serious damage to building structures. The problem of diagnostics of the buildings and structures became extremely urgent after the aggression of the russian federation in Ukraine, which led to large-scale damage and destruction of industrial plants, housing stock and infrastructure projects such as roads, bridges, tunnels, etc.

Operations to technical diagnostics involve visual inspection of building structures, assessment of their condition, and ability to provide the necessary functionality and load-bearing capacity. The identification and assessment of structural defects is often a very time-
consuming task if it is performed by direct visual inspection or using classical diagnostic methods. There is also a possibility that due to subjective human factors, some defects may remain unnoticed. Many critical infrastructure projects need regular inspections that require significant human resources. These works can be dangerous for specialists’ health and life, for example, when they are carried out in places difficult to access, at height, in partially destroyed and emergency buildings. Thus, there is an urgent need to automate the processes of diagnostics of buildings and structures and develop new methods for identifying defects in building structures that would save human resources and reduce the dependence of survey results on subjective factors.

In recent years, there has been a rapid development of artificial intelligence methods in all areas of science and technology. One of the most common types of these methods is artificial neural networks (ANNs). By modelling the work of the human brain, after appropriate training, ANNs are able to solve effectively problems of recognition, classification, and analysis of various types of data and information. A large number of works have been devoted to the use of ANNs for identifying and evaluating defects in building structures. Thus, Perez, Tah and Mosavi [1] developed a technique for diagnosing damage caused by moisture using ANNs VGG-16 [2]. Rajadurai and Kang [3] adapted ANNs AlexNet [4] to detect cracks in concrete surfaces, while Dorafshan et al. [5] adapted it for the inspection of bridges and buildings using unmanned aerial vehicles. Chun, Yamane and Maemura [6] have developed a “machine vision” system for diagnosing the technical condition of bridges using ANN sand deep learning methods. Munawar et al. [7] give a detailed overview of new research in this area.

The purpose of this study is to develop ANNs to detect cracks in concrete walls and classify them by the direction (vertical or horizontal). The solution of this problem is complicated by the fact that cracks may be often visually similar to surface defects. The images of the investigated structures can vary greatly depending on the texture of the surfaces, paint, light intensity, photography angle, etc. Cracks can also be irregular. These factors cause significant difficulties in training and testing ANN models.

2 Initial image dataset

We use the SDNET 2018 digital photo collection [8] for training and testing the ANNs. This collection consists of 56092 images of concrete structures with and without cracks, taken by a Nikon camera with a matrix resolution of 16 megapixels. To form the collection, 230 construction objects were used, which belong to three types: roads (104 objects), walls (72 objects) and bridges (54 objects). Images are scaled down to 256 by 256 pixels and have 3 colour channels with 256 luminance levels each (24-bit color).

Let us consider an ANN model that will diagnose cracks in concrete walls and classify them as vertical or horizontal. In practice, vertical cracks usually indicate foundation settlements. Horizontal cracks can occur due to insufficient bearing capacity or overloading of the structure. The selected initial set of images contains 1086 photographs evenly divided into three groups: vertical cracks, horizontal cracks, and undamaged structures. Accordingly, each group consists of 352 photos. Examples of the images are shown in Figures 1-3.

Subsequently, the initial image dataset is divided into two groups. 80-85% of images (training samples) are used to train the ANNs, that is, to establish optimal connections between neurons that allow the model to make correct decisions on data classification. The remaining 15-20% of images (test samples) are not involved in the training process, but are used to test the model and check how effectively it can process data that it encounters for the first time.
3 Development of ANNs using Teachable Machine

Teachable Machine (https://teachablemachine.withgoogle.com) is a free Google cloud tool that makes it easy to create machine learning models. Interaction with the tool is carried out through the web interface. Using the Teachable Machine, one can teach ANNs to recognize images, sounds and poses. Teachable Machine is based on JavaScript machine learning library TensorFlow.js (https://www.tensorflow.org/js).

Working with the tool is reduced to the following steps:

1. Selection of the project type: classification of images, sounds or poses. Different ANN architectures can be applied depending on this. It is worth noting that Teachable Machine does not allow the user to change independently or edit the internal architecture of the model.

2. Collection of initial data. One can upload files from a local computer, record images and poses via a webcam, and record sounds using a microphone.

3. Model training and testing. Checking how correctly the ANNs recognizes and classifies new examples.

4. Export of the model. The developed ANN model can be saved on the local computer and further used to create software applications or posted on the Internet.

It should be noted that the developed model may not always work as expected. Training a neural network is a heuristic procedure, the effectiveness of which depends on a large
number of parameters. In most cases, the optimal values of these parameters cannot be predicted and they must be determined experimentally for each individual task.

To develop the model, we use the set of images described in section 1. When training the ANNs, Teachable Machine allows the user to set the values of the following parameters:

1. The Epochs parameter determines the duration of model training. One epoch means that the model learned each training sample once. For example, if Epochs = 50 is specified, then during the training process the original data set will be processed 50 times. In general, the higher the value of this parameter, the better the model will learn to make decisions.

2. The Batch Size parameter determines the batch size. A package is a set of samples that are used in the course of one iteration during training. For example, Batch Size = 16 is specified, and the initial set of training samples contains 80 images. Then all data will be divided into 5 packets (80/16 = 5). Once the model has familiarized itself with all 5 packets, one epoch will end.

3. The Learning Rate coefficient determines the learning rate of the model, $0 \leq \text{Learning Rate} \leq 1$. It allows controlling the value of correction of the weighting coefficients of neurons at each iteration. Large values of Learning Rate $> 0.7$ correspond to large correction steps. At the same time, the model learns faster, but training errors also increase. Small values of Learning Rate $< 0.1$ correspond to small steps of correction of weight coefficients. This increases the duration of training, but allows achieving better accuracy of the model.

As noted above, in practice, the optimal values of training parameters are determined experimentally for each individual task. Figures 4-6 show the training results of the proposed model for different batch sizes of the Batch Size. In this case, Epochs = 50 and Learning Rate = 0.001 were accepted.

The Accuracy function is the proportion of correct decisions made by the model. So, if the model correctly recognizes and classifies 90 samples out of 100, then its accuracy is equal to $90/100 = 0.9$.

The Loss function allows evaluating how reliably the model has learned to make decisions. If the model predicts correctly with 100% probability, the loss is zero. In other cases, the loss will be greater. For example, let model A correctly classified the sample with a probability of 60%, and model B with a probability of 90%. Even though the accuracy of the models is the same, the loss value of model B will be less.

![Batch Size = 32](image1.png) ![Batch Size = 64](image2.png)

**Fig. 4.** Model accuracy
Figure 5. Model losses

Blue curves correspond to training samples, and orange curves correspond to test samples in figures 4, 5.

The matrix of confidence provides data on how many test samples were recognized correctly and in which classes errors were made. For example, for Batch Size = 32, the model correctly recognized 53 vertical cracks (Vertical), 46 horizontal cracks (Horizontal) and 52 intact walls (Intact). But at the same time, 3 horizontal cracks and 1 intact wall were interpreted as vertical cracks, and photographs of 4 horizontal cracks were interpreted as images of walls without cracks.

When developing and training neural networks, one of the fundamental problems is model overfitting. The state of overfitting occurs when the accuracy relative to the test samples is lower, and the losses are greater than the accuracy and losses relative to the training samples. This means that the model correctly classifies the training data, but cannot correctly recognize the data it encounters for the first time. Overfitting is the result of adaptation of the model's neural connections to the information contained in the training dataset. This means that the model has adapted not to the general, main features of the image (in our case, the presence and orientation of cracks), but to secondary details (wall texture, colour, surface chips, etc.).
Such a model becomes unable to identify common features and cannot be used in practice, even if it demonstrates high accuracy relative to training samples.

To prevent overfitting, it is necessary to select the optimal architecture of the neural network (in particular, the number of layers and neurons), and, if necessary, remove some of the neural connections, reducing the number of weight coefficients of the model. Also, the effectiveness of the neural network is significantly influenced by the value of the training parameters.

We performed a series of numerical experiments which show that with Batch Size = 32 and Batch Size = 128 the model is approaching the overfitting state. However, the best results were achieved with Batch Size = 64. In the latter case, the accuracy and loss of the model relative to the training and test samples are very close (Table 1). The matrix of confidence also shows the smallest number of misclassifications with Batch Size = 64.

The developed model is posted on the Internet and is available for free use: https://teachablemachine.withgoogle.com/models/6RsGrbES2/. Figure 7 shows the results of the model when classifying images that were not used in the training process. Several examples of incorrect decisions are shown in figure 8. It is possible to make a conclusion about the high practical accuracy of the developed ANN.

**Table 1.** Accuracy and loss of the trained model with Batch Size = 64.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training samples</td>
<td>0.970</td>
<td>0.099</td>
</tr>
<tr>
<td>Test samples</td>
<td>0.971</td>
<td>0.081</td>
</tr>
</tbody>
</table>

**Fig. 7.** Examples of model operation and degree of probability of decisions made.
4 ANN development using TensorFlow

A significant limitation of the Teachable Machine is the inability to change the internal architecture of the ANNs, which is set automatically depending on the type of task. This section is devoted to the development of an ANN with its own architecture for classifying cracks in concrete walls. The open software platform TensorFlow (https://www.tensorflow.org), developed by Google for creating and training ANNs, is used for this purpose.

TensorFlow is an end-to-end open source machine learning platform with an open source code. It contains a complex and flexible ecosystem of tools, libraries and resources that allows researchers to implement cutting-edge machine learning technologies and developers to easily build and deploy AI-based applications.

The main application programming interface (API) of TensorFlow is implemented for the Python programming language using the Keras library. It should be noted that there are also APIs for other languages: R, C Sharp, C++, Haskell, Java, Go, and Swift. Here we use the API for Python. The main advantages of TensorFlow are as follows:

1. Simple model building. TensorFlow offers several layers of abstraction so a developer can choose one that suits their individual needs. To interact with TensorFlow, API of high-level Keras is used, which greatly simplifies beginning of work with machine learning. For greater flexibility, EagerExecution mode allows you to see the results of code execution immediately and customize it easily. For larger projects, you can use the parallel computing API to run the machine learning process on different hardware configurations without changing the specific model.

2. Reliability of machine learning anywhere. TensorFlow provides a direct path to production on servers, peripheral devices, or the Internet. TensorFlow makes it easy to train and deploy a model, no matter what languages or devices are used. The TensorFlow Extended platform is designed for building large production machine learning environments. The TensorFlow.js library allows training and deploying neural networks in browsers. The TensorFlow Lite library enables machine learning models to run on mobile devices, microcontrollers, and various peripheral devices.

3. Space for experimental research. TensorFlow provides extensive opportunities for creation and training new models without limitations in speed and productivity. The Keras
Functional API and ModelSubclass API provide high flexibility and deep control when developing ANNs with complex architectures. TensorFlow also supports an ecosystem of powerful plugin libraries and models for experimentation, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor, and BERT.

Nowadays, the world's leading companies widely use TensorFlow to solve their application problems in various industries (Airbus, Airbnb, Coca-Cola, PayPal, Spotify, Twitter and many others).

To create and train the ANN in Python using TensorFlow libraries, we use the Colabary cloud environment (Colab for short) (https://colab.research.google.com). Colab has been developed by the Google Research division and it allows one writing and running Python programs through a web browser interface. From a technical point of view, Colab is a cloud hosting service for Jupyter notebooks (https://jupyter.org). The developed program code is produced on virtual machines with powerful graphics processors. Colab is free to use for research and educational purposes.

The optimal ANN architecture should be determined depending on the characteristics of each specific task and the initial data set. For the problem of recognizing and classifying cracks in concrete walls, we conducted a series of experiments and found the following rational network architecture, which allows us to achieve a high accuracy and to minimize model losses.

The proposed model is a multilayer convolutional ANN with a sequential connection of the layers. The first layer is the input image layer of the size 256x256x3, where 256x256 is the pixel size of the analysed images, and 3 is the number of colour channels for one pixel. This layer also normalizes the input data and brings all images to the same brightness level.

Next, there are 6 blocks located sequentially, each of which includes:
- 2D convolution layer tf.keras.layers.Conv2D, which detects the main features of the image;
- subsampling layer tf.keras.layers.MaxPooling2D, which reduces the size of the previous layer by compressing the image and discarding minor details;
- exclusion layer tf.keras.layers.Dropout, which randomly excludes a certain proportion (20% in our project) of the neurons of the previous layer. This reduces the total number of weight coefficients and prevents the model from retraining.

The tf.keras.layers.Flatten layer then reduces the dimensionality of the data by converting the 2D matrix input into a 1D vector output.

Next there are three conventional fully connected neural layers tf.keras.layers.Dense, which classify the received data. The size of these layers decreases gradually. The last one has a size of 3, which equals to the number of classes that the model recognizes (vertical cracks, horizontal cracks, intact walls). The detailed architecture of the ANNs is given in Table 2. The total number of model parameters, which magnitudes are determined during the training process, is 67875.

An important parameter of ANNs is the activation function. It determines what the value of the neuron's output signal is depending on the weighted average of the signals at its input. In this work, a nonlinear ReLU function is used for each layer.

The model was trained for the following parameters: Epochs = 50, Batch Size = 64, Learning Rate = 0.001. The accuracy and loss of the model are shown in figures 9, 10. Blue curves correspond to training samples, and orange curves correspond to test samples. The obtained training results indicate high accuracy and low losses of the model, which confirms the ability to recognize the data correctly that the model encounters for the first time.

Figure 11 shows examples of the model’s performance in classifying images that were not used in the training process. In these examples, the ANNs developed using TensorFlow demonstrates approximately the same high accuracy as the Teachable Machine model.
Table 2. ANN architecture.

<table>
<thead>
<tr>
<th>Layer number</th>
<th>Layer type</th>
<th>Output data size</th>
<th>Number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input Rescaling</td>
<td>256, 256, 3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Convolution 2D</td>
<td>254, 254, 32</td>
<td>896</td>
</tr>
<tr>
<td>3</td>
<td>MaxPooling2D</td>
<td>127, 127, 32</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Dropout</td>
<td>127, 127, 32</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Convolution 2D</td>
<td>125, 125, 32</td>
<td>9248</td>
</tr>
<tr>
<td>6</td>
<td>MaxPooling2D</td>
<td>62, 62, 32</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Dropout</td>
<td>62, 62, 32</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Convolution 2D</td>
<td>60, 60, 32</td>
<td>9248</td>
</tr>
<tr>
<td>9</td>
<td>MaxPooling2D</td>
<td>30, 30, 32</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Dropout</td>
<td>30, 30, 32</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Convolution 2D</td>
<td>28, 28, 32</td>
<td>9248</td>
</tr>
<tr>
<td>12</td>
<td>MaxPooling2D</td>
<td>14, 14, 32</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Dropout</td>
<td>14, 14, 32</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Convolution 2D</td>
<td>12, 12, 32</td>
<td>9248</td>
</tr>
<tr>
<td>15</td>
<td>MaxPooling2D</td>
<td>6, 6, 32</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Dropout</td>
<td>6, 6, 32</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Convolution 2D</td>
<td>4, 4, 32</td>
<td>9248</td>
</tr>
<tr>
<td>18</td>
<td>MaxPooling2D</td>
<td>2, 2, 32</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Dropout</td>
<td>2, 2, 32</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Flatten</td>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Dense</td>
<td>128</td>
<td>16512</td>
</tr>
<tr>
<td>22</td>
<td>Dense</td>
<td>32</td>
<td>4128</td>
</tr>
<tr>
<td>23</td>
<td>Dense</td>
<td>3</td>
<td>99</td>
</tr>
</tbody>
</table>

Let us consider examples where the Teachable Machine model made mistakes (Fig. 8). The results of the developed model using TensorFlow are shown in figure 12. It can be observed that the present model classifies correctly and with a high accuracy the images that were not recognized by the Teachable Machine. This allows us to conclude that the proper individual tuning of a rational ANN architecture for each individual task allows achieving significantly better results than the use of general universal solutions.
Fig. 9. Accuracy of the model

Fig. 10. Losses of the model

Fig. 11. Examples of the model’s operation and the degree of probability of made decisions
5 Conclusions

In this work, two ANN models were developed for detecting and classifying cracks in concrete walls. The Teachable Machine cloud tool and the libraries of the TensorFlow software platform were used for this purpose. We investigated how the values of the training parameters affect the accuracy of the models and determined the optimal values of these parameters. When developing the ANN using TensorFlow, a series of computational experiments were carried out and it was studied how the network architecture (number and types of layers, number of neurons, etc.) affects the performance of the model. As the result, the rational internal architecture of the ANN was determined, which allows achieving the highest accuracy and the lowest losses.

A comparative analysis of the practical effectiveness of the models developed using Teachable Machine and TensorFlow was carried out. Both models show approximately the same accuracy. But at the same time, the model created on the basis of TensorFlow avoids the classification errors made by the Teachable Machine model. This allows us to conclude that individual selection of a rational ANN architecture for each individual task allows achieving significantly better results than the use of general universal solutions.

The developed ANN models and the obtained results demonstrate the high efficiency of the methods of artificial intelligence for solving problems of building constructions diagnostics. Unlike traditional examination methods, the proposed models allow real-time automatic diagnostics by analysing photographs, images from a computer web camera, smartphone or quadcopter. The latter allows performing the inspection of structures without the physical presence of humans at the site, which is especially important when working in dangerous places: at heights, in emergency and partially destroyed buildings, in mined and war areas, etc.

The proposed methods for developing ANNs can be further extended to diagnose and classify a wide range of defects in building structures and constructions.

Acknowledgement. This work is supported by the European Commission under HORIZON 2020 project “A novel decentralized edge-enabled PREsCriptivE and ProacTive framework for increased energy efficiency and well-being in residential buildings”, grant id 958284.

References

   https://doi.org/10.3390/s19163556
https://arxiv.org/abs/1409.1556

https://doi.org/10.3390/app11115229

https://doi.org/10.1145/3065386

https://doi.org/10.1109/ICUAS.2018.8453409

https://doi.org/10.1111/mice.12793

https://doi.org/10.3390/infrastructures6080115

https://doi.org/10.15142/T3TD19