International Journal of Pioneering Technology and Engineering

Research Article

Building Damage Detection Using Deep Learning Architecture with Satellite Images: The Case of the 6 February 2023 Kahramanmaraş Earthquake

Zeynep Aygün[a], **Merve Kocaman** [a], **Salih Berkan Aydemir** [a*] **Berkant Konakoğlu**[b]
[a] Department of Computer Engineering, Engineering Faculty, Amasya University, Amasya, Turkey
[a] Department of Architecture and Urban Planning, Vocational School of Technical Sciences, Amasya University, Amasya

Article Info

Article history

Received: 27/08/2024
Revised 1: 19/09/2024
Revised 2: 11/10/2024
Accepted: 21/10/2024

Keywords:
Earthquake;
Satellite image;
Deep learning;
Remote sensingCake

ABSTRACT

Turkey is located in a region with a high density of fault lines, which makes it susceptible to a significant earthquake risk. The Kahramanmaraş earthquake on February 6, 2023, was one of the most devastating in recent years, causing extensive damage and loss. This study aims to support post-disaster rapid response and rescue operations by using deep learning techniques to detect and classify damaged and intact buildings from satellite images. Satellite images of the Kahramanmaraş and Antakya regions, with a resolution of 8192x4537, were obtained via Google Earth Pro. The images were labeled as damaged or undamaged using the Labelme editor, which generated JSON format files for the labeled images. Using Google Colab, the JSON files and unlabeled images were merged, and buildings were cropped and categorized into two classes: damaged and undamaged. As a preprocessing step, interpolation was applied, resulting in 2211 images with a size of 128x128. A Convolutional Neural Network algorithm was created using TensorFlow, a Python library, via Google Colab. The performance metrics, including accuracy, loss, F1 score, ROC curve, precision, recall, and confusion matrix values, were compared based on the experiments.

1. Introduction

Natural disasters have been events that deeply affect both individuals and societies since the beginning of human existence. Disasters cause significant losses by affecting lives, property, cultural heritage, the economic structure, and the environment. The impact of natural disasters varies depending on the type and severity of the disaster, geographical conditions, and the preparedness level of the society. Due to its geographical location, Turkey is prone to disasters such as floods, earthquakes, landslides, and avalanches. Turkey is considered one of the most seismically active regions in the world. Major fault lines, such as the North Anatolian Fault Line, East Anatolian Fault Line, and West Anatolian Fault Line, run through various regions of Turkey. The activity of these fault lines leads to numerous earthquakes of varying magnitudes in the country. The frequency and intensity of these earthquakes highlight the fact that Turkey is at significant risk of seismic activity.

The earthquake that caused the greatest loss of life and property in Turkey occurred on February 6, 2023, in the province of Kahramanmaraş and affected surrounding cities as well. As a result of this earthquake, damage assessment studies conducted by the Ministry of Environment, Urbanization, and Climate Change in 13 provinces determined that, as of February 16, 2023, 263,800 independent units in 61,722 buildings were identified as collapsed, heavily damaged, or in urgent need of demolition. Following this major disaster, there emerged an urgent need to reassess disaster management and response strategies, enhance earthquake resilience, and raise public awareness. In this context, rapid and effective post-earthquake damage assessment and swift deployment of aid teams to the necessary locations are of vital importance. For this purpose, technological solutions such as detecting and classifying damaged and undamaged buildings using satellite images with various deep learning methods have come to the forefront. Information from satellite images can be processed using deep learning algorithms to detect damaged buildings and improve the coordination of aid teams more effectively. These technologies can contribute to reducing human casualties and enabling affected individuals to receive assistance more quickly by ensuring that relief efforts are carried out more swiftly and in a more coordinated manner.

Considering that traditional texture analysis methods for post-earthquake damage assessment are more time-consuming and yield less reliable results compared to deep learning-based approaches [1], this study focuses on the classification of images of damaged and undamaged buildings obtained from areas affected by the earthquakes that occurred in Kahramanmaraş on February 6, 2023, using Convolutional Neural Networks [2], a deep learning architecture. The aim is to facilitate rapid damage detection and ensure more effective and coordinated delivery of assistance. In this context, post-earthquake images were obtained using Google Earth Pro. A CNN model was developed for the classification of the acquired images. All coding and model training for the study were conducted on the Google Colab platform. The integration of deep learning methods with satellite imagery can enhance post-disaster response processes and contribute to the existing literature.

2. Literature Review

A review of the literature indicates that assessing damage, evaluating the impact of an earthquake, and quickly identifying areas requiring urgent intervention are crucial after an earthquake. Traditional methods for such planning are time-consuming and costly. However, using remote sensing technologies to detect

***Corresponding author:** Salih Berkan Aydemir
*E-mail address: salih.aydemir@amasya.edu.tr
https://doi.org/10.56158/jpte.2024.94.3.02

damage and implement necessary actions can be done more quickly and effectively. During the comparison of pre- and post-event images, automatic change detection can be challenging due to different imaging parameters. Therefore, visual interpretation of pre- and post-event data is generally preferred for determining building damage. Many studies have shown that visual interpretation is effective in this regard [3]. There are also methods that reveal damage through mathematical operations comparing pre- and post-event data. For example, methods such as ratioing and differencing focus on comparing brightness values and color differences between pre- and post-event images [4, 5]. However, these methods require precise spatial co-registration, which can sometimes be challenging [6].

In recent years, the use of deep learning-based techniques has shown great potential in post-earthquake damage assessment. Deep learning methods, trained on large datasets, can automatically extract complex features from the data and accurately identify subtle changes.

In the study titled "A Comparative Study of Texture and Convolutional Neural Network Features for Detecting Collapsed Buildings After Earthquakes Using Pre-and Post-Event Satellite Imagery," [1] Ji et al. utilized pre-event (pre-earthquake) and post-event (post-earthquake) satellite imagery for damage assessment following earthquakes. The study compared the effectiveness of the two methods by applying a traditional texture analysis method alongside a deep learning-based approach using Convolutional Neural Networks [2]. The comparison observed that CNN features outperformed traditional texture features, achieving an accuracy rate of 87.6%.

Ghaffarian et al., in their research titled “Updating Post-Disaster Building Database Using Automatic Deep Learning: Integration of Pre-Disaster OpenStreetMap and Multi-Temporal Satellite Data” [7], developed a database by integrating OpenStreetMap (OSM) data with multi-temporal satellite images. A deep learning based model was developed for the detection of buildings and updating their post-disaster status. CNN architecture was used to determine the damage levels of buildings by analyzing the changes in multi-temporal satellite images. The highest success values in the study were obtained with a test image among the event time images. This image gave the best results with an F1 score of 88.3%.

In the article titled "Automatic Detection of Damaged Buildings in Post-Disaster Scenarios: A Case Study of the Earthquakes Occurred in Kahramanmaraş (Turkey) on February 6, 2023" [8], a deep learning-based approach is proposed to automatically detect damaged and undamaged buildings using aerial or satellite images of earthquake-affected areas. In this study, the highest accuracy was stated as 0.973 and this success was achieved in the FPN (Feature Pyramid Network) model.

3. Material and Method

The research was conducted by processing satellite images of regions damaged by the earthquakes centered in Kahramanmaraş on February 6, 2023, utilizing the Convolutional Neural Network [2] deep learning architecture. This investigation is quantitative and has a cross-sectional design.

The analysis was carried out using numerical data and statistics in the damage assessment and classification processes. The satellite images were obtained and processed in digital format, with damage conditions categorized into specific groups and evaluated using numerical metrics. This approach ensured an objective and measurable analysis of the data.

The study assessed the aftermath of the earthquake occurring on February 6, 2023, within a defined time frame. This cross-sectional design was used to determine the extent of the damage by analyzing satellite images obtained post-earthquake. Data collection and analysis were performed at a single point in time following the earthquake, reflecting the study’s cross-sectional nature. This design enabled a rapid and comprehensive assessment in the aftermath of the disaster.

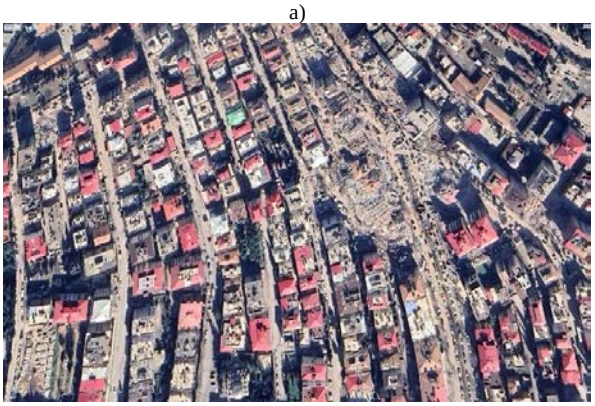
This type of research was deemed appropriate as it allows for the numerical analysis of the data and facilitates a swift assessment following a disaster. Additionally, its cross-sectional nature enabled the rapid and effective determination of the damage status across large areas.

3.1. Data Collection

Pre- and post-earthquake images of the regions affected by the earthquakes centered in Kahramanmaraş on February 6, 2023, were obtained in the highest resolution using Google Earth Pro. Google Earth Pro is a software that provides geographic data from around the world through high-resolution satellite imagery. Users can examine changes in a specific region over time and compare images from different dates. Due to the earthquake occurring during the winter season, weather conditions made it challenging to obtain clear satellite images in some areas. Therefore, the focus was placed on the Kahramanmaraş and Hatay regions, where the clearest images were available. A total of 64 clear images, covering both pre- and post-earthquake periods, were successfully acquired.

Specifically, 28 images were obtained from 14 regions affected by the earthquake in Kahramanmaraş, where the epicenter was located. These included 14 pre-earthquake images taken on February 4, 2023, and 14 post-earthquake images taken on February 16, 2023. Additionally, from Hatay, another province most affected by the earthquake, a total of 34 images were obtained from 17 regions. These included 17 pre-earthquake images taken on December 12, 2022, and 17 post-earthquake images taken on February 9, 2023. The resolution of the images was set to 8192 x 4537. These high-resolution images are crucial for achieving high accuracy in post-earthquake damage assessment. Figure 1 shows the pre- and post-earthquake satellite images obtained using Google Earth Pro.

a)



b)

Fig. 1. Pre(a)- and post(b)-earthquake satellite images

3.1.1. Image labeling

The buildings in the obtained satellite images were manually labeled using the LabelMe Studio editor. LabelMe is an open-source image annotation tool that allows users to label and classify objects within images. During this process, pre- and post-earthquake images were compared, and buildings were labeled as either damaged or undamaged, creating two distinct classes. After completing the labeling process, the data was saved in JSON format. These JSON files contain the annotation information and are used for accurate image classification. An example of post-earthquake images labeled as damaged and undamaged using the LabelMe editor is shown in Figure 2.

Fig. 2. Post-earthquake image labeled as damaged and undamaged.

3.1.2. Image cropping

Using labels from JSON files, damaged and undamaged buildings were identified and cropped from the unlabelled images obtained after the earthquake. As a result of this cropping process, the buildings were categorized into two separate classes. The cropped versions of the labeled satellite images, according to the damaged and undamaged classes, are presented in Figure 3.

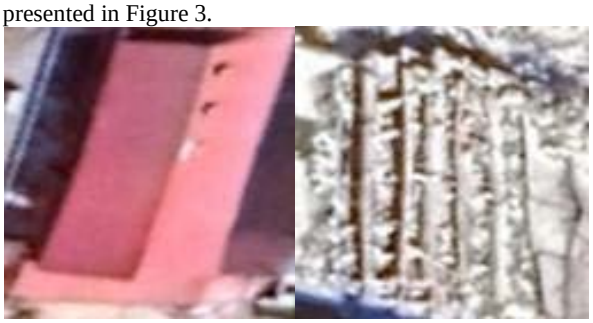


Fig. 3. Cropped images of the labeled data

3.1.3. Image resizing

The cropped images, being of different sizes, could negatively impact the model's training; therefore, these images need to be resized to a fixed size. The resizing process was carried out using interpolation methods.

Interpolation is a commonly used method in digital image processing. This technique is used to estimate the values of new pixels in operations such as image enlargement, reduction, and geometric transformation. It is a fundamental tool when images need to be resampled and resized in computer vision applications. There are three main interpolation methods. Nearest Neighbor Interpolation is often preferred for rapid prototyping and real-time image processing due to its simplicity and speed, but it may produce sharp edges and a 'blocky' appearance. Bilinear Interpolation provides smoother images by performing linear interpolation on both axes and is suitable for smooth transitions. Bicubic Interpolation, on the other hand, uses a weighted average of 16 surrounding pixels to produce more natural and smooth results, making it suitable for situations where high-quality image outputs are required [9].

The Nearest Neighbor interpolation method has not been used due to its tendency to produce a blocky appearance and cause sharp transitions in the image, while the Bicubic interpolation method was excluded because of its processing time and computational cost. By employing the Bilinear interpolation method, a balance between processing speed and output quality has been achieved, allowing for the generation of resized images that maintain high quality with minimal degradation.

The cropped images were resized using the Bilinear interpolation method through OpenCV, an open-source library in Python developed for computer vision and image processing applications, via Google Colab. The initial image in Figure 4 is a cropped image with dimensions of 708x976 before the interpolation method was applied. After interpolation, the same image was resized to 128x128 dimensions.

(a) (b)

Fig. 4. The cropped image (a) and its resized version (b)

3.2. Building and Training a Deep Learning Model

3.2.1. Convolutional neural networks [2]

Convolutional Neural Networks (CNNs) were first introduced by Yann LeCun and his team in a seminal paper published in 1998. These networks function similarly to the human visual system, extracting meaning from images. One of the key advantages of CNNs is their ability to learn specific features from complex and high-dimensional datasets, such as images, by focusing on smaller and meaningful portions rather than processing every single pixel. This approach enables the model to operate more efficiently and with reduced computational cost. Particularly effective in image recognition tasks, CNNs learn different levels of features at each layer, gradually developing a more abstract understanding of the input data. LeCun and his team emphasized that the use of local connectivity and weight sharing in CNNs contributes to their efficiency, allowing them to perform well on complex data structures [10].

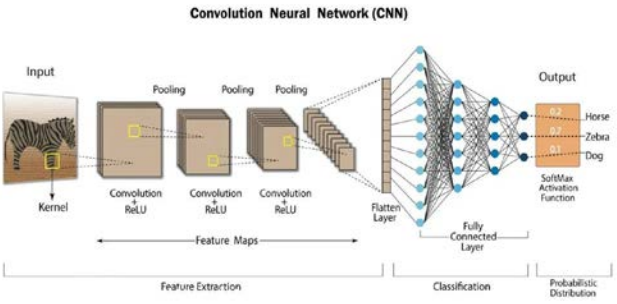


Fig. 5. Convolutional Neural Network [2]

3.2.2. Architecture of the deep learning model used

To classify the image data described under the data collection section, the deep learning model used is a Convolutional Neural

Network [2]. The model was built and all subsequent stages were conducted using the Google Colab platform. Google Colab is a cloud service that allows for writing and running Python code in a Jupyter Notebook environment. The main reasons for using Google Colab in the current work include: its free access to powerful computing resources such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), which enables faster training and testing of deep learning models; its ability to facilitate easy sharing and collaboration on project files, allowing multiple researchers to work together; and its cloud-based nature, which allows for quick setup and rapid prototyping of different models without requiring any installation.

The CNN model used to train the obtained images includes 2 convolutional layers, 2 pooling layers, and 2 fully connected layers, and was built using Python's TensorFlow library and its modules.

TensorFlow is an open-source machine learning library developed by Google. TensorFlow's flexibility and performance provide significant advantages in the development and training of deep learning models. It reduces computation time considerably with its powerful GPU and TPU integrations, especially when working with large datasets and complex models [11]. To build the model, the Keras library, which provides a high-level API for quickly prototyping and using deep learning models under TensorFlow, was used.

Initially, the Sequential class of Keras's Model module was used to create a sequential and layered model. The layers of the CNN model were then constructed using the layers module of Keras, which includes various types of layers for building deep learning models. The conv2D class from the layers module was used to create 2 convolutional layers, the maxpool2D class was used to create 2 max pooling layers, and the Dense class was used to create 2 fully connected layers. For both conv2D layers, 32 filters with a filter size of 3x3 were used. The first conv2D layer is where the images are input to the model for processing, so the shape of the images was set to 128x128 with 3 channels. For both maxpool2D layers, the filter size for pooling was set to 3x3, and the stride (the number of pixels the filter moves) was set to 3. For the first fully connected layer created using the Dense class, the number of neurons was set to 128. Since the image data is classified as either damaged or undamaged, the number of neurons/outputs for the final fully connected and output layer was set to 2. Figure 6 shows the schematic of the created CNN model.

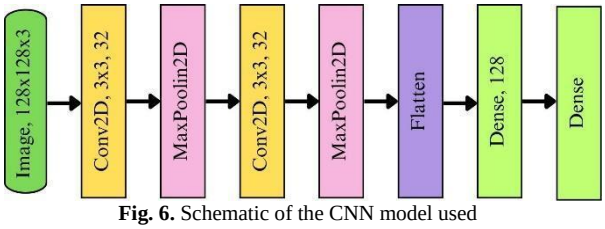


Fig. 6. Schematic of the CNN model used

This model was developed considering the complexity of the features that can be extracted from a dataset containing 1549 training samples of roof images from both damaged and undamaged buildings. Decisions regarding hyperparameters such as the number of layers, the number of filters in each layer, and the filter sizes were made with the goal of completing the training without overfitting and successfully performing the classification tasks.

3.2.3. Model training

For training the model, images that were resized after cropping were divided into training, validation, and test sets. The training set is used during the model's learning process to enable the model to recognize various damaged and undamaged building images. The validation set is used to evaluate the model's performance during training, while the test set measures the model's overall performance. This separation is critical for enhancing the model's generalization capability and accuracy. 70% of the images, consisting of 1549 images, were used for the training set; 15%, totaling 331 images, were allocated for the validation set; and the remaining 15%, also comprising 331 images, were used for the test set.

The preprocessing and loading of image data for model training were handled using Keras's ImageDataGenerator class. ImageDataGenerator is a powerful tool used for image data augmentation and preprocessing. This class helps in making the model more generalized by applying various random transformations to the dataset. It also performs essential preprocessing tasks such as loading and rescaling the training, validation, and test datasets. For the training set, data augmentation and preprocessing steps included rescaling pixel values from the range 0-255 to 0-1, random cropping (a type of geometric transformation where pixels in the image are shifted by a certain angle), random zooming, and random horizontal flipping to help the model better recognize symmetrical objects. No data augmentation techniques were applied to the validation and test datasets, as these are used to objectively evaluate the model's performance and assess the final performance respectively. The images were resized to 128x128, so during dataset loading, the dimensions were set to 128x128. The batch size was set to 32. The shuffle parameter was turned off for the test dataset to keep the dataset's order fixed, which helps in objectively evaluating the model's performance and better preparing it for real-world scenarios.

3.2.4. Activation functions used in the model

Activation functions are mathematical functions used in artificial neural networks to determine the output of a neuron. These functions process input signals to calculate the neuron's activation level, and this information is passed to subsequent layers in the network's forward propagation. Activation functions introduce non-linearity into neural networks, enabling the networks to learn complex functions. Since most real-world problems involve nonlinear structures, it is important for networks to be able to model nonlinear functions [12]. In the model, ReLU, ELU, Leaky ReLU, SiLU, Hard SiLU, and Mish were used as activation functions in the convolutional layers, while Sigmoid and Softmax were used as activation functions in the first fully connected layer. The graphs of the activation functions used are given in Figure 7.

ReLU (Rectified Linear Activation): ReLU is a simple activation function that returns the input value directly when it is positive and outputs zero when it is negative. It is widely used in deep learning models because it is easy to compute and helps deep networks learn more quickly and effectively [12].

ELU (Exponential Linear Unit) : ELU is a function designed to solve the problem of ReLU's zeroing of negative values. When the input is negative, it gradually approaches zero, and when it is positive, it directly outputs the value. This can increase the learning capacity of the network [13].

Leaky ReLU : Leaky ReLU is a variation of ReLU and uses a small slope for negative inputs. This helps avoid the dead ReLU problem. It uses a small negative value for negative inputs [14].

SiLU (Sigmoid-weighted Linear Unit): SiLU is an activation function that scales the effect of the input by using the sigmoid function. It is obtained by multiplying the input by the sigmoid function [15].

Hard SiLU (Hard Sigmoid-weighted Linear Unit) : Hard SiLU is a simpler and less computationally expensive version of SiLU. It uses a piecewise linear analog and is defined as x * ReLU6(x + 3) / 6. This makes it suitable for mobile devices and other computation-limited environments [16].

Mish: Mish is an activation function calculated by multiplying the input with a variation of the sigmoid function. It provides a smooth and continuous curve [17].

Sigmoid: The sigmoid function is a nonlinear function that converts the input into a value between 0 and 1, effectively normalizing the activation level of the neuron. However, when the input values are large, either positive or negative, the function's gradient can become very small, leading to the 'vanishing gradient' problem, which can slow down or halt the learning process of the network [18]. The sigmoid function transforms the model's linear output into a probability value constrained between 0 and 1, thereby representing the likelihood of an object belonging to a specific class. In this context, the sigmoid function is suitable for binary classification tasks [19].

Softmax: Softmax is used in multi-class classification problems and transforms the output values into a probability distribution [20].

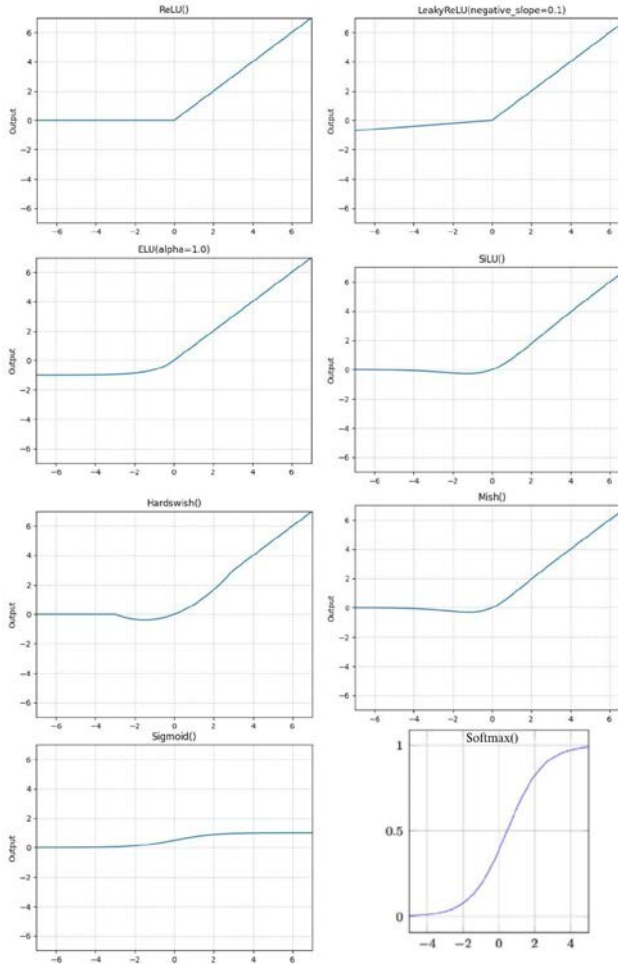


Fig. 7. Activation function graphs used

3.2.5. Optimization algorithms used in the model

Optimization algorithms are used to update the weights of a neural network during the learning process. These algorithms aim to achieve the best performance by minimizing the model's loss function [12]. In the model's final fully connected layer, optimization algorithms are included using the 'optimizers' module from the Keras library, which allows for optimization during model training. The optimization algorithms used include Adam, Adamax, Adagrad, Adadelta, SGD, and RMSprop.

Adam (Adaptive Moment Estimation): Adam is a generalized version of the stochastic gradient descent (SGD) algorithm. It provides an adaptive learning rate by using moving averages of the moments and squared gradients. Momentum and adaptive learning rate are combined. It keeps estimates of the first moment (mean) and the second moment (variance). It is particularly effective for time series data and deep neural networks [21].

Adamax: Adamax is a variation of the Adam algorithm. Adamax is a generalized version of Adam using the infinity norm (max norm) It requires less memory and has a simpler structure compared to Adam. Adamax is more resistant to the slowdown issue that can sometimes occur with Adam on large datasets [21].

Adagrad (Adaptive Gradient Algorithm): Adagrad is a generalized version of SGD where each parameter has its own learning rate. It keeps a running total of the squared gradients and uses this sum to calculate an adaptive learning rate for each parameter. It tends to have a lower learning rate for frequently used parameters. However, as training progresses, the learning rate can become very small, which may slow down the training process [22].

Adadelta : Adadelta is a generalized version of Adagrad. Instead of storing the squared gradients, Adadelta uses the moving average of the squared changes in the gradients. This approach addresses the disadvantage of Adagrad's decreasing learning rate over time. Adadelta is memory-efficient and can perform better in models with a large number of parameters [23].

SGD (Stochastic Gradient Descent):SGD calculates the gradient for each training example and updates the model parameters in the negative direction of the gradient. The term "stochastic" implies that it uses a randomly selected subset of the data to calculate the gradient for each training example. It is computationally inexpensive and a widely used optimization algorithm. However, it can exhibit slow convergence in irregular datasets and high-dimensional parameter spaces [24].

RMSprop (Root Mean Square Propagation):RMSprop is a generalized version of Adagrad. Instead of continuously updating the sum of squared gradients like Adagrad, RMSprop uses a moving average of the squared gradients [25].

In this model, all activation functions and optimization algorithms have been evaluated in combination, resulting in 72 different scenarios (Figure 8).

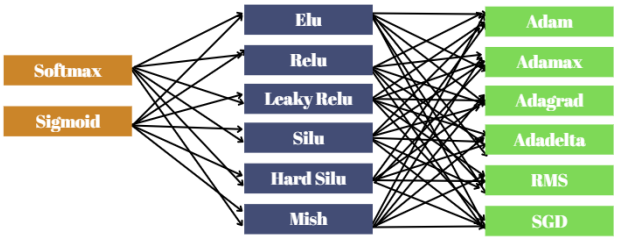


Fig. 8. Combinations of Activation Functions and Optimization Algorithms Used in the Model

3.3. Evaluation of Deep Learning Model

The correct selection and application of evaluation metrics is a critical step in deep learning projects. To objectively measure and compare a model's success, it is essential to select the appropriate metrics. Each model has different strengths and weaknesses, and these metrics help in understanding these aspects. Moreover, continuously monitoring and analyzing these metrics is important for understanding how the model performs over time. The model evaluation process not only involves analyzing the results but also understanding the circumstances under which the model performs well or poorly. This process provides the necessary feedback for continuously improving the model and achieving better outcomes [12].

3.3.1. Evaluation metrics for deep learning models

There are various evaluation metrics used to assess the performance of deep learning models. These metrics provide

information about the model's predictive ability, accuracy, and overall performance. The commonly used evaluation metrics for deep learning models and their explanations are as follows:

Accuracy is the ratio of the number of correctly predicted instances to the total number of instances. The loss function serves as an objective function that measures the difference between the model's predictions and the actual values. It expresses this difference as a numerical value and is used as a target to be minimized during the model's learning process. Precision measures how many of the items that the classification model has labeled as positive are actually positive. In other words, it indicates how many of the examples identified as positive by the model are correct. Recall measures how many of the truly positive examples the classification model has correctly identified. F1 score is the harmonic mean of precision and recall, and it is used to evaluate the overall performance of a model by balancing these two metrics. The ROC Curve is a graph used to visualize the performance of a classification model. The curve shows the True Positive Rate against the False Positive Rate. A confusion matrix is a table that summarizes the performance of classification models. The matrix shows the number of actual and predicted classes and includes the counts of true positives, false positives, true negatives, and false negatives [19].

3.3.2. Evaluation of the Model Used

The model was tested in 72 different scenarios using various activation functions and optimization algorithms, with comparisons made using accuracy and loss metrics. Hyperparameters were evaluated across four cases, where the batch size parameter was set to 4, 8, 16, and 32. The analysis showed that a batch size of 32 provided better outcomes than other values. Following these evaluations, two models with similar performance remained. To further compare these models, each was run 30 times, and metrics such as accuracy, loss, F1 score, precision, recall, and the ROC curve were calculated for every run. Additionally, averages for these metrics were computed over the 30 runs. The model achieving the highest accuracy after 30 runs was saved, and a confusion matrix was created for this model. The evaluation metrics for this final model are presented in the results section.

4. Results

In the combinational trials of the activation functions and optimization algorithms used in the model, 72 different scenarios have emerged. Among these 72 scenarios, there are 36 different cases for softmax and 36 different cases for sigmoid in the 2nd dense layer. When the results of the two activation functions were compared, softmax yielded better results in terms of loss and accuracy evaluations.

Among the types of learning algorithms, Adam has given the best result in 12 different scenarios tested with the silu activation function. The accuracy graph is shown in Figure 9, and the loss graph is shown in Figure 10.

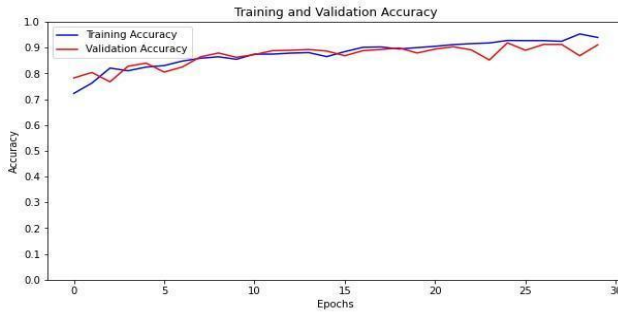


Fig. 9. The accuracy graph of the model using the Adam optimization algorithm and the silu activation function

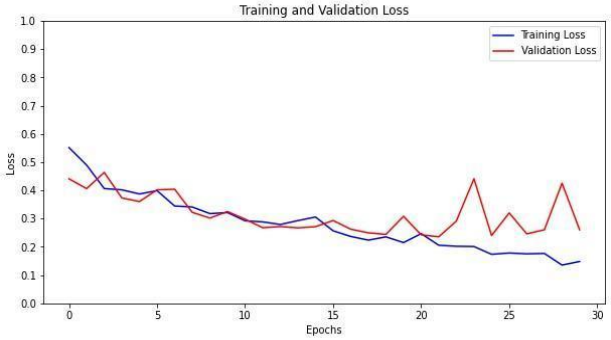


Fig. 10. The loss graph of the model using the Adam optimization algorithm and the silu activation function

Among the types of learning algorithms, Adagrad has given the best result in 12 different scenarios tested with the elu activation function. The accuracy graph is shown in Figure 11, and the loss graph is shown in Figure 12.

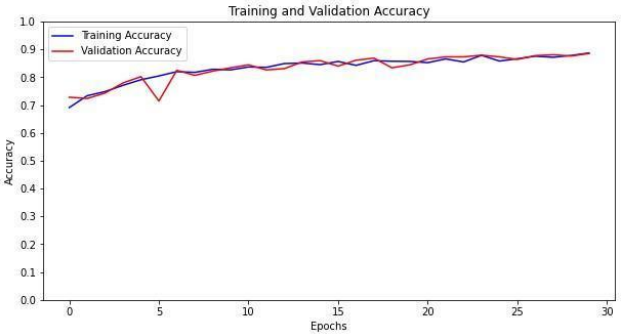


Fig. 11. The accuracy graph of the model using the Adagrad optimization algorithm and the elu activation function

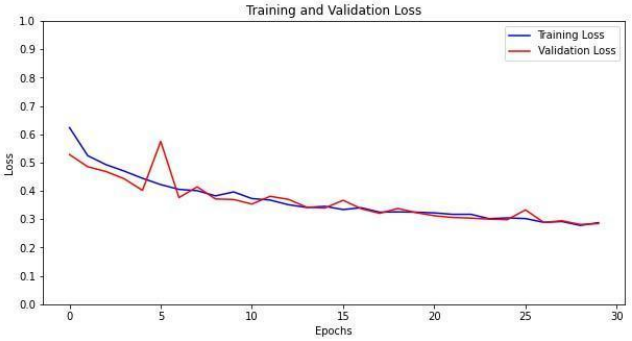


Fig. 12. The loss graph of the model using the Adagrad optimization algorithm and the elu activation function

Among the types of learning algorithms, SGD has given the best result in 12 different scenarios tested with the elu activation function. The accuracy graph is shown in Figure 13, and the loss graph is shown in Figure 14.

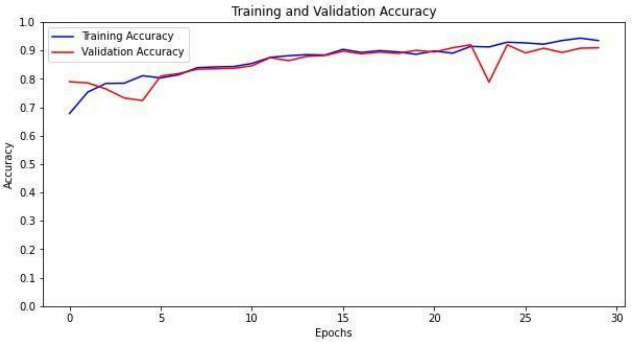


Fig. 13. The accuracy graph of the model using the SGD optimization algorithm and the elu activation function

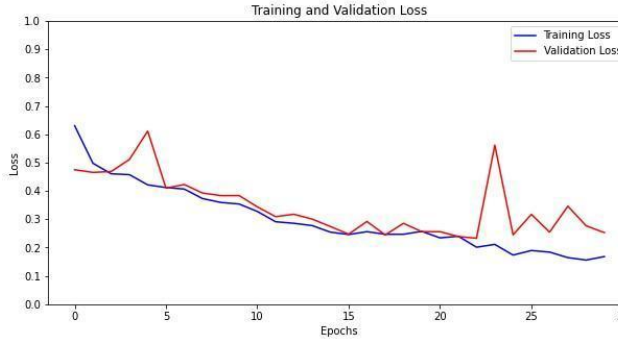


Fig. 14. The loss graph of the model using the SGD optimization algorithm and the elu activation function

Among the types of learning algorithms, Adamax has given the best result in 12 different scenarios tested with the elu activation function. The accuracy graph is shown in Figure 15, and the loss graph is shown in Figure 16.

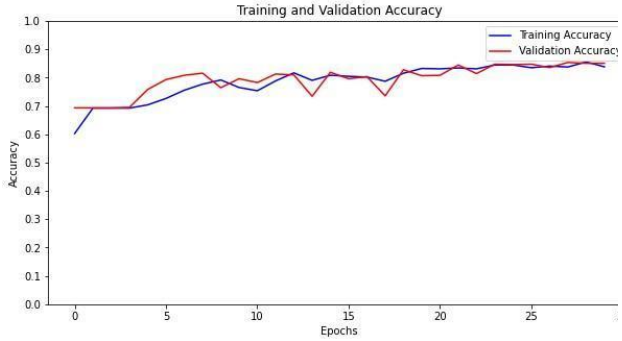


Fig. 15. The accuracy graph of the model using the Adamax optimization algorithm and the elu activation function

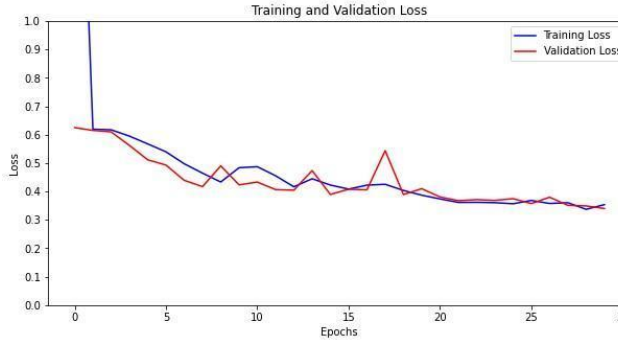


Fig. 16. The loss graph of the model using the Adamax optimization algorithm and the elu activation function

Among the types of learning algorithms, Rmsprop has given the best result in 12 different scenarios tested with the relu activation function. The accuracy graph is shown in Figure 17, and the loss graph is shown in Figure 18.

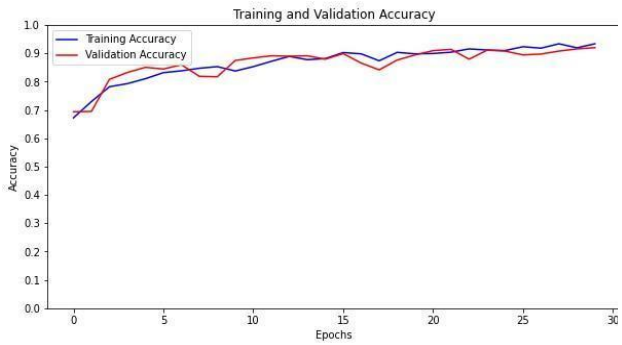


Fig. 17. The accuracy graph of the model using the Rmsprop optimization algorithm and the relu activation function

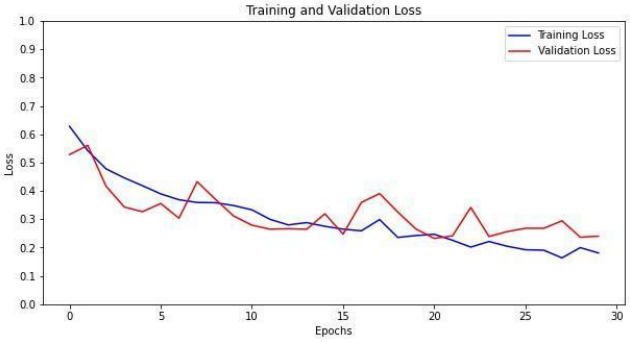


Fig. 18. The loss graph of the model using the Rmsprop optimization algorithm and the relu activation function

Among the types of learning algorithms, Adadelta has given the best result in 12 different scenarios tested with the leaky relu activation function. The accuracy graph is shown in Figure 19, and the loss graph is shown in Figure 20.

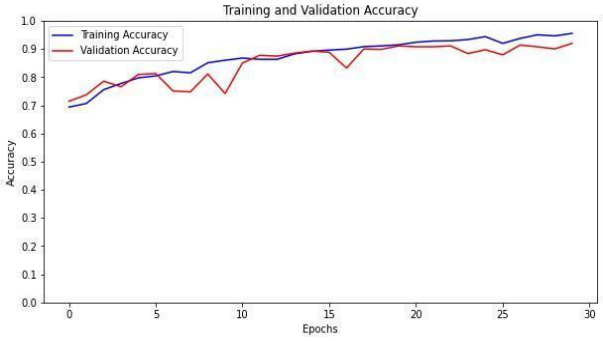


Fig. 19. The accuracy graph of the model using the Adadelta optimization algorithm and the leaky relu activation function

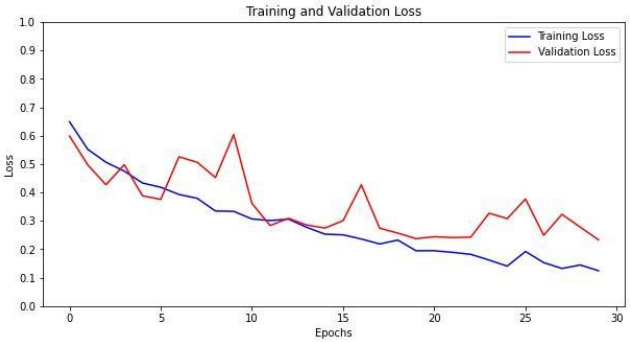


Fig. 20. The loss graph of the model using the Adadelta optimization algorithm and the leaky relu activation function

The train loss, train accuracy, validation loss, and validation accuracy values of the combinations that give the best results in 72 cases are presented in Table 1.

Table 1. Results of optimization algorithms

Optimization algorithm	**Activation function**	**Loss**	**Accuracy**	**Val_Loss**	**Val-Accuracy**
RMSprop	Relu	0.1809	0.9335	0.2401	0.9199
Adadelta	Leaky relu	0.1245	0.9555	0.2334	0.9199
Adagrad	Elu	0.2879	0.8871	0.2845	0.8852
Adam	Silu	0.1481	0.9394	0.2603	0.9109
Adamax	Elu	0.3536	0.8381	0.3400	0.8505
SGD	Elu	0.1682	0.9348	0.2526	0.9094

In the comparison between Sigmoid and Softmax, Softmax achieved the best results. When examining the best results for Softmax, RMSprop and Adadelta optimizers yielded the best performance. To compare the optimization algorithms and obtain

the best results, 30 different runs were conducted. During the runs, the best model was saved, and the confusion matrix for this model was obtained. The F1 score, ROC curve, precision, and recall values were calculated by taking the average of the results from 30 runs.

The average metric values of the models that ran with RMSProp and Adadelta optimization algorithms as a result of these runs are provided in Table 2.

Table 2. Evaluating model metrics

Optimization Algorithm	Activation Function	Average accuracy	Average loss	Average f1-score	Averı recall
Adadelta	Leaky relu	0.9041	0.2916	0.9091	0.904
RMSprop	Relu	0.8914	0.3531	0.8856	0.891

In Table 2, Adadelta with Leaky ReLU yielded the best average metric values. Table 3 shows the 30-epoch training results for the top-performing model after 30 run.

Table 3. Results of the model reaching the best accuracy and loss value

Epoch Number	Loss	Accuracy	Validation-Loss	Validation-Accuracy
Epoch 1	0.6580	0.6850	0.6703	0.6918
Epoch 2	0.5889	0.7088	0.6462	0.5952
Epoch 3	0.5232	0.7340	0.4869	0.7613
Epoch 4	0.4570	0.7811	0.5749	0.7372
Epoch 5	0.4497	0.7954	0.4256	0.8066
Epoch 6	0.4249	0.8063	0.4421	0.7915
Epoch 7	0.3893	0.8250	0.4063	0.8066
Epoch 8	0.3754	0.8283	0.4160	0.8157
Epoch 9	0.3482	0.8373	0.5660	0.7432
Epoch 10	0.3567	0.8405	0.3463	0.8248
Epoch 11	0.3265	0.8651	0.3364	0.8248
Epoch 12	0.3264	0.8509	0.3557	0.8489
Epoch 13	0.2887	0.8748	0.3101	0.8489
Epoch 14	0.2805	0.8715	0.2762	0.8912
Epoch 15	0.2586	0.8909	0.2979	0.8792
Epoch 16	0.2828	0.8812	0.2957	0.8882
Epoch 17	0.2731	0.8844	0.3857	0.8369
Epoch 18	0.2209	0.9090	0.2521	0.9003
Epoch 19	0.2059	0.9154	0.3089	0.8761
Epoch 20	0.2065	0.9174	0.3442	0.8580
Epoch 21	0.2250	0.9019	0.2635	0.8973
Epoch 22	0.1913	0.9258	0.2420	0.9063
Epoch 23	0.1895	0.9225	0.2423	0.9063
Epoch 24	0.1937	0.9245	0.2471	0.9003
Epoch 25	0.1723	0.9329	0.2353	0.9063
Epoch 26	0.1683	0.9251	0.2253	0.9124
Epoch 27	0.1384	0.9393	0.2847	0.9033
Epoch 28	0.1327	0.9451	0.2044	0.9124
Epoch 29	0.1744	0.9264	0.2026	0.9215
Epoch 30	0.1315	0.9484	0.1993	0.9245

After 30 runs, the loss graph of the best model is presented in Figure 22, and the accuracy graph is presented in Figure 23.

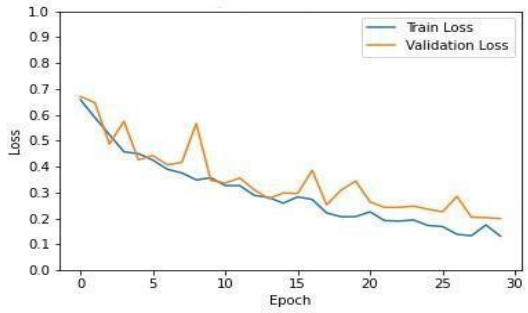


Fig. 22. Loss graph of the best model for the Adadelta optimization algorithm and leaky ReLU activation function

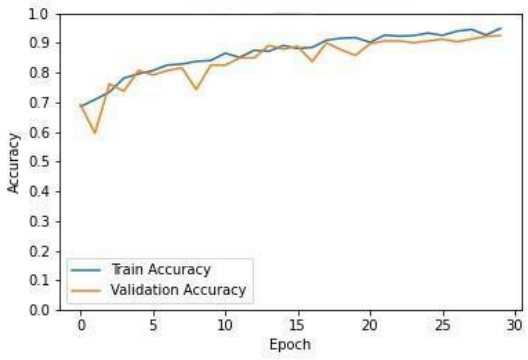


Fig. 23. Accuracy graph of the best model for the Adadelta optimization algorithm and leaky ReLU activation function

The average roc curve graph after 30 runs is shown in Figure 24.

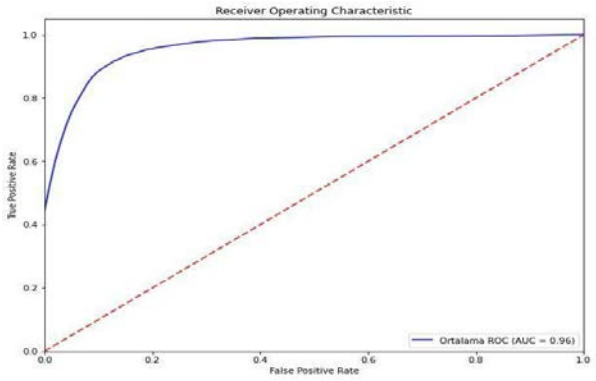


Fig. 24. Average ROC curve graph for the Adadelta optimization algorithm and leaky ReLU activation function

After 30 runs, the model with the highest accuracy was evaluated on the test data to assess its overall performance and generalization ability. The accuracy, loss, and confusion matrix were calculated for the test dataset. The model achieved an accuracy of 92% and a loss of 0.24 on the test data. The confusion matrix of the model is shown in Figure 3.17. Out of 331 damaged and undamaged test images, the model classified 84 damaged buildings correctly as damaged, 17 damaged buildings incorrectly as undamaged; 225 undamaged buildings correctly as undamaged, and 5 undamaged buildings incorrectly as damaged.

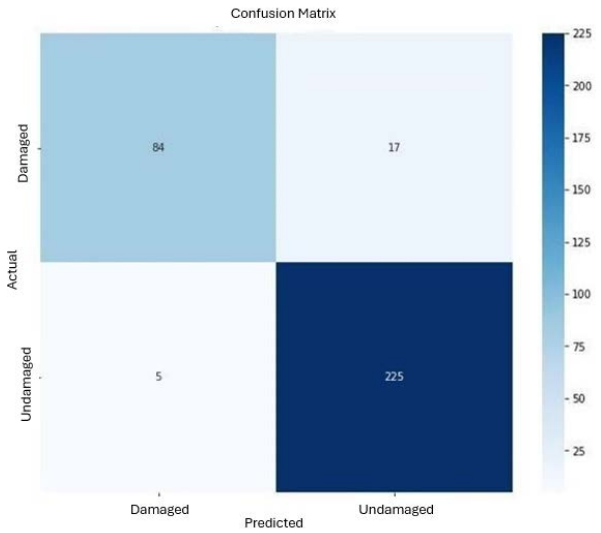


Fig. 25. The confusion matrix of the model with the best accuracy

Figure 26 shows 4 building images where the model classified the damaged building as damaged, the damaged building as undamaged, the undamaged building as undamaged and the undamaged building as damaged among the cropped images.

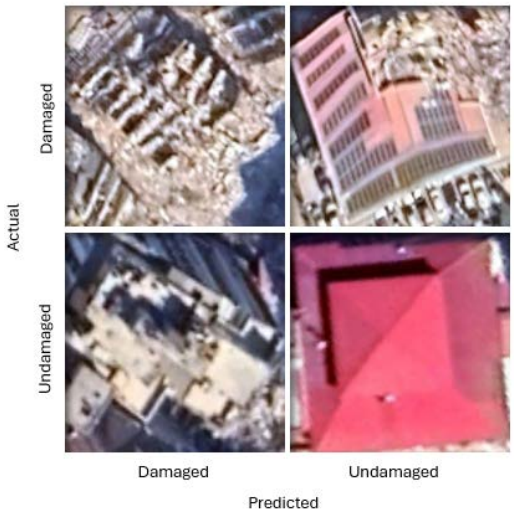


Fig. 26. Image matrix classified by the model

5. Conclusions

The performance of a Convolutional Neural Network [2]-based model in classifying a building as damaged or undamaged following a disaster was evaluated. The model achieved an accuracy rate of 92% on the validation dataset. This result aligns with similar studies in the literature, such as "A Comparative Study of Texture and Convolutional Neural Network Features for Detecting Collapsed Buildings After Earthquakes Using Pre-and Post-Event Satellite Imagery" [1], which reported an accuracy of 87.6%, and "Automatic Detection of Damaged Buildings in Post-Disaster Scenarios: A Case Study of the Earthquakes Occurred in Kahramanmaraş (Turkey) on February 6, 2023"[8], which achieved an accuracy of 97%. These findings further confirm the high accuracy of CNNs in image classification tasks.

The success of the CNN model is directly related to the quality and diversity of the constructed dataset. The model performed particularly well in cases where the general roof shape was preserved and edge lines were distinct for undamaged buildings, and where damage was clearly identifiable and building features (such as windows and collapsed roofs) were present in the debris for damaged buildings. However, accuracy decreased in the classification of undamaged buildings that deviated from the general roof shape and contained details such as solar panels and terraces, as well as buildings with low damage and no roof damage.

The outcomes demonstrate significant potential for applications in detecting damaged and undamaged buildings after a disaster. However, the model's performance has certain limitations. Notably, the size and diversity of the dataset are important factors affecting the model's overall success. Therefore, it is recommended that future studies utilize larger and more diverse datasets. Additionally, employing data augmentation techniques and optimizing the model's hyperparameters could further enhance classification accuracy.

In conclusion, findings indicate that CNN-based models offer an effective solution for the classification of buildings as damaged or undamaged. These insights may have important applications in optimizing post-disaster relief efforts and total damaged building detection. It is anticipated that with broader datasets and advanced modeling techniques, this approach can be further advanced in future research.

Declaration of conflicting interests

The authors declare no competing interests.

Funding

The author received 6000 TL financial support from TUBITAK 2209-A Project for this study.

References

[1]. Ji, M., Liu, L., Du, R., and Buchroithner, M.F.,2019, *A comparative study of texture and convolutional neural network features for detecting collapsed buildings after earthquakes using pre-and post-event satellite imagery*, J Remote Sensing, 11(10): 1202.

[2]. Shahriar, N.,2023, *What is convolutional neural network–CNN (Deep Learning)*, J Electronic resource.

[3]. Saito, K., Spence, R.J., Going, C., and Markus, M.,2004,*Using high-resolution satellite images for post-earthquake building damage assessment: a study following the 26 January 2001 Gujarat earthquake*, J Earthquake spectra, 20(1): 145-169.

[4]. Oommen, T., Rebbapragada, U., and Cerminaro, D.,2012, *Earthquake damage assessment using objective image segmentation: a case study of 2010 Haiti earthquake*, *GeoCongress 2012: State of the Art and Practice in Geotechnical Engineering*. 3069-3078.

[5]. Sugiyama, M. and Abe, H.S.K,2002, *Detection of earthquake damaged areas from aerial photographs by using color and edge information*. Proceedings of the 5th Asian Conference on Computer Vision, Melbourne, Australia.

[6]. Chen, Z. and Hutchinson, T.C.,2010, *Image-based framework for concrete surface crack monitoring and quantification*, J Advances in Civil Engineering, 2010(1): 215295.

[7]. Ghaffarian, S., Kerle, N., Pasolli, E., and Jokar Arsanjani, J.,2009, *Post-disaster building database updating using automated deep learning: An integration of pre-disaster OpenStreetMap and multi-temporal satellite data*, J Remote sensing, 11(20): 2427.

[8]. Serifoglu Yilmaz, C., Yilmaz, V., Tansey, K., and Aljehani, N.S.,2023, *Automated detection of damaged buildings in post-disaster scenarios: a case study of Kahramanmaraş (Türkiye) earthquakes on February 6, 2023*, J Natural Hazards, 119(3): 1247-1271.

[9]. Sonka, M., Hlavac, V., and Boyle, R.,2013, *Image processing, analysis and machine vision*: Springer.

[10]. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.,1998, *Gradient-based learning applied to document recognition*, J Proceedings of the IEEE, 86(11): 2278-2324.

[11]. Zaccone, G.,2016, *Getting started with TensorFlow*: Packt Publishing ISBN.

[12]. Goodfellow, I., Bengio, Y.,2016, and Courville, A., *Deep learning, vol. 29*, MIT Press.

[13]. Clevert, D.-A.,2015, *Fast and accurate deep network learning by exponential linear units (elus)*, J arXiv preprint arXiv:.07289.

[14]. Maas, A.L., Hannun, A.Y. , and Ng, A.Y.,2013, *Rectifier nonlinearities improve neural network acoustic models*. Proc. icml: Atlanta, GA.

[15]. Elfwing, S., Uchibe, E., and Doya, K.,2018, *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning*, J Neural networks, 107: 3-11.

[16]. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., and Vasudevan, V.,2019, *Searching for mobilenetv3*. Proceedings of the IEEE/CVF international conference on computer vision.

[17]. Misra, D.,2019, *Mish: A self regularized non-monotonic activation function*, J arXiv preprint arXiv:.08681.

[18]. LeCun, Y., Bottou, L.,2002, Orr, G.B., and Müller, K.-R., *Efficient backprop, Neural networks: Tricks of the trade*, Springer. 9-50.

[19]. Bishop, C.M. and Nasrabadi, N.M.,2006, *Pattern recognition and machine learning*, 4: Springer.

[20]. Bridle, J.,1989, *Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters*, J Advances in neural information processing systems, 2.

[21]. Kingma, D.P.,2014, *Adam: A method for stochastic optimization*, J arXiv preprint arXiv:.07289.

[22]. Duchi, J., Hazan, E., and Singer, Y.,2011, *Adaptive subgradient methods for online learning and stochastic optimization*, J Journal of machine learning research, 12(7).

[23]. Zeiler, M.D.,2012, *ADADELTA: an adaptive learning rate method*, J arXiv preprint arXiv.

[24]. Montavon, G., Orr, G., and Müller, K.-R.,2012, *Neural networks: tricks of the trade*, 7700: springer.

[25]. Tieleman, T.,2012, *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*, J COURSERA: Neural networks for machine learning, 4(2): 26.