Natural Hazards
https://doi.org/10.1007/s11069-023-05991-2

ORIGINAL PAPER

Automatic post-tsunami loss modeling using deep learning CNN case study: Miyagi and Fukushima Japan tsunami

Shaheen Mohammed Saleh Ahmed[1] · **Hakan Güneyli**[2]

Received: 10 April 2022 / Accepted: 22 April 2023
© The Author(s), under exclusive licence to Springer Nature B.V. 2023

Abstract
Assessing the destruction caused by a tsunami is a challenging task that must be completed quickly with limited resources and information. To address this issue, we propose a method for accurate damage mapping using binary classification of high-resolution satellite imagery, where we enhance the performance of three pre-trained deep neural network models (Vgg19, Inception, and Xception). The pre-trained models are used, which have been previously trained on large datasets, and transferred to our tsunami problem. We also develop a custom network architecture specifically designed for tsunami damage detection using high-resolution remote sensing data, improving the accuracy of automated binary classification. We investigate the impact of various parameters and learning rates to detect small objects, demonstrating the suitability of our approach for tsunami damage assessment. Our network outperforms traditional and current deep learning-based approaches, as it shows low bias and high variance datasets that result in a skillful model. Specifically, we observe that Inception-v3 performs best on the dataset, exhibiting good behavior with low errors and achieving the best overall score with 24.11 min, while other models score between 30.50 min for Vgg19 and 45.33 min for Xception. Our study focuses on two important binary classification categories, tsunami-stricken and non-stricken areas, for which we train the proposed framework on a dataset comprising 30,000 small tiles of high-resolution satellite images obtained from Mexer satellite images. The model is validated on 8000 images using the Jupyter notebook of the Anaconda deep learning framework.

Keywords Tsunami · CNN · Deep learning · VGG · Pre-trained

✉ Shaheen Mohammed Saleh Ahmed
shaheengeolo@gmail.com

Hakan Güneyli
hguneyli@cukurova.edu.tr

[1] Geology Department, College of Science, Kirkuk University, Kirkuk, Iraq

[2] Geology Department, Faculty of Engineering, Cukurova University, Adana, Turkey

Published online: 07 May 2023

1 Introduction

After a massive earthquake, it is essential to carry out structural damage inspections as quickly as possible for emergency preparedness and to optimize recovery time. However, in the aftermath of such a catastrophe, time is of the essence in generating accurate damage mapping to swiftly deliver aid to the appropriate locations and ultimately save lives (Ohta et al. 2004). The Tohoku-Oki earthquake of 2011 was a 9.1 undersea mega-thrust earthquake that occurred on March 11, 2011, at 2:46 p.m. local time (JST). It caused the devastating Tohoku tsunami, which had waves as high as 40 m and wreaked havoc in the coastal towns of Iwate, Miyagi, and Fukushima prefectures. In the Sendai region, it traveled up to 5 km inland (Mori et al. 2011). Over the past two decades, tsunamis have caused significant death tolls and economic losses, particularly for countries bordering the Indian and Pacific Oceans, as highlighted by the United Nations Office for Disaster Risk Reduction (UNISDR) and Imamura et al. (2019). Available data from tsunami events show that from 1998 to 2017, these losses totaled 251,770 deaths and US$280 billion, out of a total of US$661.5 billion in recorded economic losses for earthquakes and tsunamis. In comparison, over the previous 20 years, tsunamis were responsible for 999 fatalities and US$2.71 billion in recorded losses, totaling US$410.91 billion in recorded economic losses from earthquakes and tsunamis (1978–1998).

Although there has been significant progress in machine learning, there remains a gap between current state-of-the-art techniques and their application in tsunami investigations. A lack of collaboration has been observed between the geoscience and computer science communities, particularly in the field of artificial intelligence (Sublime 2021). The following studies are the most important machine learning and deep learning works in the areas of seismic, structural, and tsunami damage mapping, presented with their original bracketed citations:

1. Different deep learning techniques have been employed in various civil engineering applications, including building assessment (Arnold and Tilton 2017), crack damage detection and classification (Cha et al. 2017; Lin et al. 2017), real-time traffic management (Hashemi ane Abdelghany 2018), seismic reliability (Nabian and Meidani 2018), and pixel-level detection for structures and pavements (Yang et al. 2018; Zhang et al. 2017).
2. Endo et al. (2018) proposed a multi-class damage classification method for buildings using TerraSAR-X images taken before and after the tsunami. Their study focused on the Sendai seaside region and utilized support vector machines (SVMs) to classify the retrieved information on damage.
3. Wieland et al. (2016) compared SVMs with simple thresholding methods for tsunami damage classification using TerraSAR-X and ALOS PALSAR images taken before and after the catastrophe. They aimed to demonstrate that basic machine learning algorithms such as SVMs perform better than commonly used image thresholding methods. Their approach is based on pre-segmentation using existing blueprints before transferring the data to the SVM algorithm.
4. Minghelli's et al. (2020) study differed from the others in that it focused on damage caused by the Tohoku tsunami to the shoreline itself, in the form of erosion or accretion, rather than mapping damage to structures or submerged regions. The authors applied an SVM to Worldview-2 images of the seashore in Fukushima Province and Sendai Channel taken before and after the tsunami.

5. Chini et al. (2013) utilized the *K*-Means algorithm (MacQueen 1967), the most well-known and simple unsupervised AI algorithm, to cluster data points into *K* groups. The algorithm is initialized with random *K* points and iteratively connects each datum to its nearest prototype, updating the prototypes as the mean of the linked data, until convergence is achieved.
6. Bai et al. (2018) proposed a supervised damage-detection method using the U-Net algorithm (Ronneberger et al. 2015), which is a family of deep convolutional neural networks commonly used for medical imaging. The U-Net segments images and classifies the segments into classes using a supervised learning techniqu(Ioffe and Szegedy 2015). Bai et al. made changes to the base U-Net design, including the use of stride 2 convolutions to replace max-pooling layers and a complete restructuring of the upper U-Net's branch. Their algorithm was implemented using the Microsoft Computational Network Toolkit (Seid and Agarwal 2016).
7. Fujita et al. (2017) examined how CNNs could identify structures that had been washed away and utilized aerial images, which offered higher resolution but limited coverage area compared to other studies. Their approach was found to be the most versatile for mapping tsunami damage.

Finally, CNN-based approaches such as Krizhevsky et al.'s (2012) AlexNet, Szegedy et al.'s (2015) GoogLeNet, and Simonyan & Zisserman's (2014) VGG16 represent the state-of-the-art in CNNs, which can be fine-tuned using transfer learning to detect seismic damage in a new dataset (Nabian and Meidani 2018).

All related works and previously studies used small dataset or synthetic dataset for training and validation. Generally, it is well known that a poor approximation is caused by a lack of training data or using fake synthetic data. A model's performance estimation will be overly optimistic and subject to high variance if there are insufficient validation data. This research improved pre-trained models' performance by increasing both training and validation data with low bias and high variance datasets and adjusted learning rate for all models by using Keras ***ReduceLROnPlateau*** which resulted in a skillful network. Also, we repurposed and fine-tuned the pre-trained model so that we might gain even better precision. We dropped the top layers since our recent problem isn't the same as the old one. By adding layers specific to our problems, we can achieve higher accuracy. We demonstrated in detail the suitability of our network for tsunami damage assessment by comparing our network with current deep learning-based approaches baseline and show that our strategy surpasses current methods in the result section. We have directly modeled the relationship between the measurement data and tsunami damage information by training adequate and huge dataset consisting of 30,000 high-resolution maxar satellite images for training and 8800 for validation for study area.

2 Methodology

2.1 Transfer learning and pre-trained models

Transfer learning involves utilizing an already trained model to solve related problems, thereby leveraging the knowledge acquired from one task to enhance the generalization performance on another task. By utilizing pre-trained weights or model architecture,

transfer learning allows for the adaptation of a model to solve a new problem by simply changing the output layer. This technique is time-efficient, improves performance in most cases, and reduces the need for extensive datasets. Although it is not a universal approach, it is highly effective in resolving related issues. Keras applications offer users access to numerous architectures, including Vgg19, Inception, Xception, and others, with a comprehensive list of their notable accuracies on ImageNet classifications. In this paper, we have focused on investigating the top specifications of three models from Keras applications, namely parameters, depth, and size, for effective image classification. These models are Vgg19, Inception, and Xception, and their respective specifications are presented in Table 1.

2.2 The advantages of CNN for tsunami loss assessment

In this study, we selected a Convolutional Neural Network (CNN) model as our preferred machine learning method for assessing Tsunami loss due to its several key advantages:

Firstly, CNNs are particularly effective in handling image data. Since Tsunami damage assessment often involves analyzing satellite or drone images, CNNs are well-suited for this task. Specifically, CNNs are able to learn spatial hierarchies of features in images, allowing them to automatically detect and classify different types of damage.

Secondly, CNNs have been shown to be effective even with relatively small datasets. This is because they are able to learn from a large number of parameters, which is particularly important for image classification tasks with limited data. *Furthermore*, CNNs have been shown to outperform other machine learning models in previous research studies for image classification tasks. This is due to their ability to learn from large and complex datasets, which is often the case in Tsunami loss assessment. By using a CNN model, we were able to leverage its superior performance to accurately classify and assess the extent of Tsunami damage in satellite images.

Overall, the advantages of CNNs for Tsunami loss assessment are clear. They are particularly effective in handling image data, able to learn from limited datasets, and have superior performance compared to other machine learning models. By leveraging these advantages, we were able to add new advanced information to the field of Tsunami loss assessment, providing valuable insights for disaster response and mitigation efforts.

Table 1 Concise summary of three pre-trained models (VGG19, Inception, and Xception)

Model	Size (MB)	Topl: Accuracy (%)	Top2: Accuracy (%)	Number of parameters	Depth
YGG-19	549	71.3	90.0	143,667,240	26
Inception v3	92	77.9	93.7	23,851,784	159
Xception	88 MB	79.0	94.5	22,910,840	126

The table provides information on the size, depth, number of parameters, and accuracy of each model. The data presented in this table were obtained from "Available models of Keras API reference/Keras applications" by Chollet, F., & others (2015), which can be found on GitHub at https://keras.io/api/applications/

2.3 Why we used VGG19, Inceptionv3, and Xception

In order to select a proficient CNN model from three different types of models, namely Deeper model, Wider Model, and depthwise separable convolutions Model, this paper chose VGG19, Inceptionv3, and Xception for specific reasons:

1. VGG19 was selected despite its relatively large network architecture weights because of its simplicity. VGG19 uses only 3×3 convolutional layers stacked on top of each other, increasing in depth. Despite being a deep network, VGG19 uses small kernels, which is a unique feature of this layer network.
2. The paper also attempted to use a wider model, Inceptionv3, which computes 1×1, 3×3, and 5×5 convolutions within the same network module. The Inception module serves as a "multi-level feature extractor," and the filters produced by this module are stacked along the channel dimension and added to the next layer of the network. Inceptionv3 replaces large filters with small filters, which is another unique feature of this model.
3. Xception, which is an extension of the Inception architecture, was chosen because it uses depthwise separable convolutions instead of the standard Inception modules. This unique feature of Xception involves depthwise convolution, followed by pointwise convolution.

2.4 VGG-19

VGG-19 is a CNN model with a depth of 19 layers, developed and trained by Karen Simonyan and Andrew Zisserman at the University of Oxford in 2014, as shown in Fig. 1. The

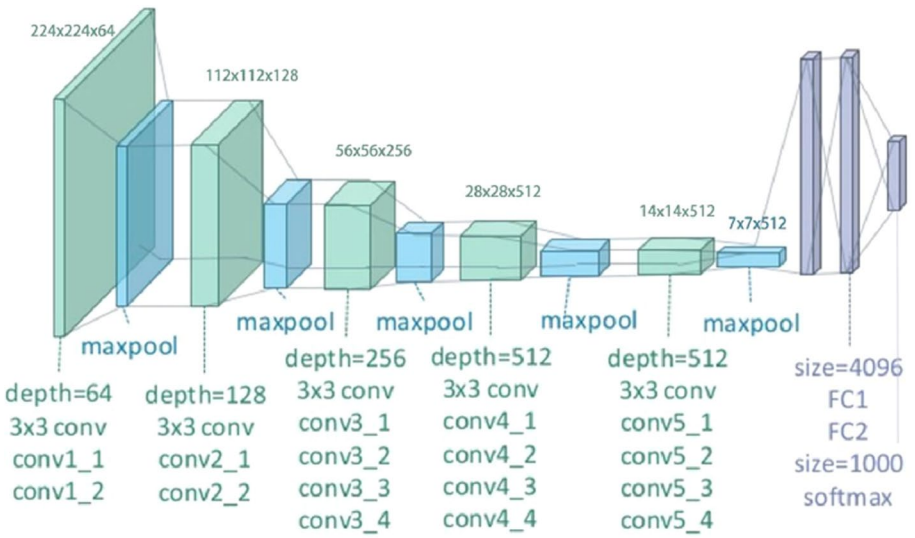


Fig. 1 Diagram of VGG-19, as depicted in "Breast cancer screening using convolutional neural network and follow-up digital mammography" by Zheng et al. (2018), published in Journal of X-ray Science and Technology. The article can be accessed at https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10669/1066905/Breast-cancer-screening-using-convolutional-neural-network-and-follow-up/10.1117/12.2309968.short?SSO=1

model was trained on over 1 million images from the ImageNet collection, and its details can be found in their paper. The VGG-19 network, which has been pre-trained to classify up to 1000 categories, can be imported from Keras applications with ImageNet trained weights. This network was trained on color images with a resolution of 224 × 224 pixels. Despite the large size of its network architecture weights, VGG-19 is characterized by its simplicity, using only 3 × 3 convolutional layers stacked on top of each other in increasing depth. The unique feature of the VGG-19 layer network is its use of small kernel sizes in a deep network. Table 1 provides a summary of the model's specifications and performance.

2.5 Inception-v3 (GoogLeNet)

Inception-v3 is a CNN model that consists of 50 layers, as shown in Fig. 2, and was developed and trained by Google (Szegedy et al. 2016a, b). The pre-trained version of Inception-v3 comes with ImageNet weights, which enables it to classify over 1000 features. Unlike VGG19, this model requires larger input images with a resolution of 299 × 299 pixels. In the ImageNet competition of 2014, Inception-v3 won first place, while VGG-19 was the runner-up.

2.6 Xception model

Chollet (2017) introduced the Xception model, an enhanced version of the Inception architecture that employs depthwise separable convolutions to replace the standard Inception modules as shown in Fig. 3. The data flow through the input, followed by eight cycles of the middle flow, and finally through the exit flow. Batch normalization is applied to all convolutional and separable convolutional layers.

2.7 Finding a skillful classification model and base model criteria

To find a skillful classification model, we tested various deep learning pre-trained techniques with varying learning rates and chose the baseline CNN model from three well-known pre-trained CNN models (VGG19, Inception3, and Xception) for our dataset using

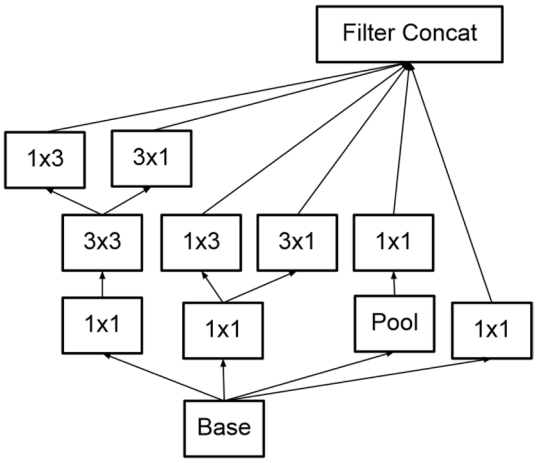


Fig. 2 Original Inception module, as presented in the research paper "Rethinking the Inception Architecture for Computer Vision" by Szegedy et al. (2016a, 2016b). The paper can be accessed from the IEEE Xplore Digital Library at https://arxiv.org/abs/1512.00567

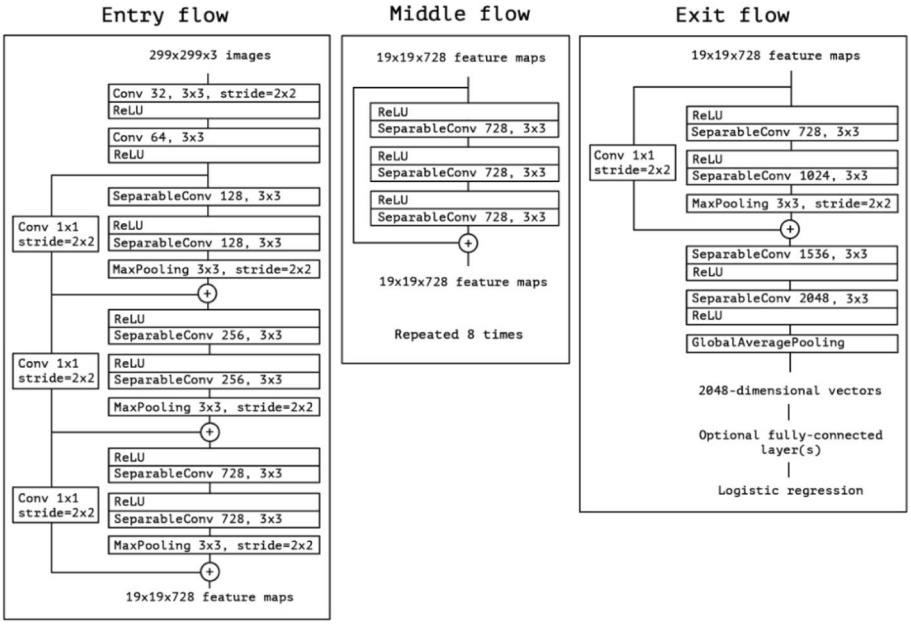


Fig. 3 Architecture of the original Xception model, as presented in Chollet's 2017 paper "Xception: Deep Learning with Depthwise Separable Convolutions," which employs depthwise separable convolutions in place of traditional Inception modules. The model consists of an entry flow, followed by eight cycles of a middle flow, and a final exit flow. Additionally, batch normalization is applied to all convolution and separable convolution layers. The paper was published in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) and can be accessed at https://ieeexplore.ieee.org/document/8099678

transfer learning. This established a minimum model performance to which all other models can be compared, as well as a model architecture that we could use as the basis for study and improvement.

In selecting the base model, we considered two competing criteria in machine learning tasks: model accuracy and speed of training and predictions. Additionally, we added three more criteria: the number of model parameters, validation loss, and model complexity. Our aim was to improve a model with the lowest validation loss and highest accuracy on validation data since it can make useful predictions. For production purposes, we needed a model that could predict and train quickly and easily to serve hundreds or thousands of predictions each second.

2.8 Strategies for finding the best model

Our strategy is straightforward, and we have followed the following procedures as shown in Fig. 4:

2.8.1 Preparation and preprocessing of dataset

We have utilized the Python image slicer library to slice more than 500 large high-resolution satellite images obtained from Maxar technologies (Fig. 5). Firstly, each

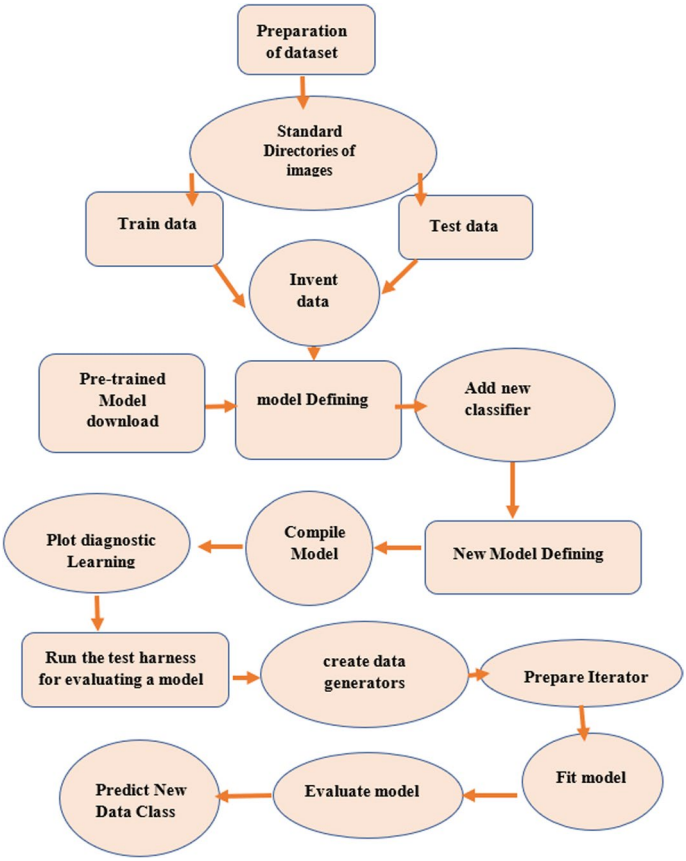


Fig. 4 Theoretical schematic model that illustrates the strategies proposed in this paper for finding the best model. The figure provides a visual representation of different stages involved in the process and the key factors that need to be considered to achieve optimal performance

high-resolution image was sliced into many small tiles using the Python slicer library, with the same number of image pixel rows and columns provided for consistency. For this binary classification task, we have two classes—tsunami-stricken and non-stricken areas. The dataset comprises 30,000 high-resolution satellite image tiles, which were split into a training set consisting of 15,000 tiles for each of the two classes and a validation set containing 8800 high-resolution satellite image tiles, which were split into a validation set consisting of 4400 tiles for each of the two classes. To load and preprocess the images efficiently, we used the Keras *ImageDataGenerator* class along with the *flow_from_directory() API*. This API requires the data to be organized into separate train/ and validation/ directories, with a sub-directory for each class under each directory, such as train/stricken/ and train/non-stricken/. The same organization was used for the validation dataset as well. This allowed us to group the images into their respective subdirectories and label them accordingly.

Before modeling, we reshaped all images to a fixed size of 224×224 pixels using the Keras image processing API. We also converted the images to grayscale to reduce

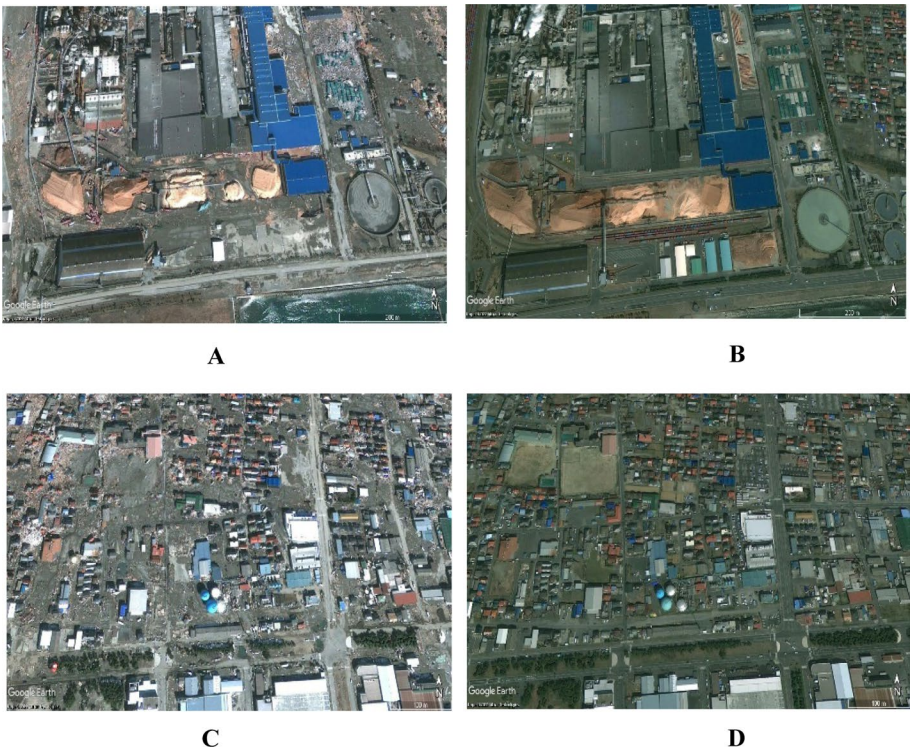


Fig. 5 High-resolution satellite images of the town of Fukushima and its surrounding area. Panels A and C show the region after the 2011 tsunami, while panels B and D show the same area before the disaster occurred. The images provide a visual comparison of the changes in the landscape caused by the tsunami

computational complexity and enhance model performance. The image preprocessing step also included normalization to ensure all pixel values fell between 0 and 1.

To make the training and validation datasets easily accessible, we saved the preprocessed images as *numpy* arrays using the *numpy.save()* function. The resulting files, 'stricken.npy' and 'nonstricken_labels.npy,' contain all the resized images and their associated class labels, making them quick and easy to load.

To ensure reproducibility, we created an Anaconda (Anaconda Software Distribution 2020) environment with the necessary python packages, including *tensorflow (*Abadi et al. 2015*), matplotlib (*Hunter 2007*), pandas (McKinney* 2010*) and tqdm (Casper da Costa et al.* 2020*).* We used the *pip install* command to install these packages in the *tf-gpu* deep learning environment.

Overall, our preprocessing pipeline ensured that the data were organized, resized, normalized, and saved in a way that would facilitate efficient and effective model training.

2.8.2 Pre-trained models preparation for transfer learning

We downloaded pre-trained models from Keras applications and removed the output layers. Our approach differs from previous works in that we made changes to all pre-trained models in the following manner:

Firstly, we froze the weights of the pre-trained models to prevent them from being modified during training. *Secondly*, we constructed an empty Sequential model and added the pre-trained model to it. Next, we added a single output Dense layer with sigmoid activation and compiled it with *categorical_crossentropy loss*. *Finally*, we trained the model for 20 epochs using the *model.fit function.*

To evaluate the performance of the models, we recorded the number of parameters, which represents the size of the original model, and the final accuracy of each model. We stored this information in a dictionary to visualize the results.

Unlike previous works, we only made changes to the last layer of the pre-trained models. This is because our paper focuses on a binary classification problem, whereas these models are designed to handle up to 1000 classes. Since we do not need to train all the layers, we made them non-trainable.

Our approach provides a novel way of preparing pre-trained models for transfer learning by modifying only the last layer, which is particularly useful for binary classification tasks. Additionally, our method of logging the number of parameters and final accuracy allows for easy comparison and selection of the best-performing models.

2.8.3 Defining the model

We constructed a function called *define_model()*, which was used to specify various baseline models, such as Inception, Exception, and Vgg19, to be fit on the data. Each model was fitted with the Adam optimizer, a popular optimization algorithm used in deep learning (Diederik and Jimmy 2014), with a conservative learning rate of 0.001. Since our challenge is a binary classification problem, we set the "*class_mode*" argument to specify the forecasting of one of two values: 0 or 1. To accomplish this, we utilized a single node in the output layer with a sigmoid activation and improved the models by utilizing the binary cross-entropy loss function.

To prepare the images for training, we implemented the *ImageDataGenerator*, which scaled the pixel values to the 0–1 range. We then built iterators for both the train and validation datasets using the *data generator flow_from_directory()* function on the data generator. One iterator was created for each of the train/ and validation/ directories, and the images were loaded and reshaped to the size of 224×224 pixels via the "target_size" argument.

We began with a conservative batch size of 8 and then fitted the model using the train iterator (train_it) and used the valid iterator (valid_it) as a validation dataset during training. The maximum number of batches in one epoch was determined by dividing the total number of images in the train and valid directories by the batch size (8), and the model was fitted for 20 epochs.

After fitting the final model, it was tested against the validation set, and the classification accuracy was recorded. We constructed a history plot, which was gathered during training and saved in the "history" directory restored from the call to fit_*generator()*. We extracted the accuracy and loss on the valid and training dataset from the history at the end

of each epoch. Learning curves were obtained using line plots of these metrics over training epochs to get an idea about the model overfitting or underfitting or having a good fit.

Executing this model first obtains the size of the train and validation datasets, proving that the dataset was loaded successfully. The model's output is used as the input for a new classifier model. The pre-trained model's weights have been frozen to prevent them from being changed when our new model is trained.

Our approach provides a best way of defining models for binary classification tasks that can be easily modified to suit different models and data. The use of the *ImageDataGenerator* ensures that the data are appropriately preprocessed, while the history plot provides a useful tool for visualizing the model's performance during training. Additionally, freezing the weights of the pre-trained models prevents them from being modified during training, ensuring that the models' performance is not negatively impacted.

2.8.4 Improving pre-trained models' performance

To improve the performance of pre-trained models, this study employed several techniques.

Firstly, the amount of training data was increased. The initial training run used only 16,000 images and the initial testing run used only 1000, resulting in underfitting. Therefore, an additional 14,000 new training images and 7800 new testing images were added to the original testing images. The quality of any model is constrained by the quality of training data. By using both old and new data, the performance of the models was improved.

Secondly, more data were invented using augmentation techniques, which created variations in the images to improve the ability of the fit models to generalize what they had learned to new images. Variations in the training set images that were likely to be seen by the model were added, such as vertical shifting, image rotation, zooming, and brightness adjustment. However, random rotation and random zooming augmentation were removed from the models' code as they underfit the pre-trained models.

Thirdly, different batch sizes and number of epochs were experimented with. Small batch sizes were used with a large epoch size, and batch sizes of 8, 16, and 32 were tried. The best performance was found with a batch size of 32 for the dataset.

Fourthly, the learning rate of the models was tuned using Keras *ReduceLROnPlateau*. This adjusted the learning rate when the model's performance reached a plateau, i.e., no variation for a specified number of training epochs, in order to fine-tune the model weights.

Finally, pre-trained models were repurposed and fine-tuned. The original classifier was removed, a new classifier was added that was appropriate for the study needs, and the model was adjusted using one of three methods: (a) training the model entirely, (b) training some layers while others are frozen, or (c) freezing the convolutional base. The layers, features, weights, and biases of these pre-trained models, which had a very large dataset as their base dataset, were repurposed. By fine-tuning the pre-trained model, the study aimed to gain even better precision and produce output in the properly updated architecture. The top layers were dropped since the recent problem was not the same as the old one, and layers specific to the problem were added to achieve higher accuracy. While the bottom and mid-level layers of a neural network normally represent generic information, the upper levels usually reflect problem-specific features.

2.8.5 Evaluation metrics for classification

The evaluation metrics used to assess the accuracy of the pretrained models in classifying tsunami-stricken and non-stricken areas include precision, recall, and $F1$_score.

Precision is calculated in binary classification with two classes as the ratio of true positives to the total of true positives and false positives. The resulting score ranges from 0.0 (no precision) to 1.0 (perfect precision), and it is calculated using the precision_score() function from scikit-learn.

Recall, on the other hand, measures the number of correct positive predictions made out of all possible predictions. Unlike precision, recall takes into account the positive predictions that were missed, providing an idea of the positive class's coverage. In a two-class imbalanced classification problem, recall is determined by dividing the number of true positives by the total number of true positives and false negatives. The resulting score ranges from 0.0 (no recall) to 1.0 (perfect recall).

F1_score is a measure that combines both precision and recall into a single score. Precision and recall alone may not provide a complete picture of the classification model's performance, as we could have high precision but poor recall, or vice versa. F1_score allows us to consider both metrics by calculating the harmonic mean of precision and recall. The conventional F1_score is calculated as follows:

$$F1_\text{score} = 2 * (\text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall})$$

Overall, these evaluation metrics provide valuable insights into the performance of classification models and can help guide model improvement and selection.

3 Result and discussion

We utilized a *NVIDIA RTX50i* Graphics processing unit (GPU) with the *TensorFlow* framework (Abadi et al. 2015) for our training. To provide a clear understanding of our data preprocessing steps and model configurations, we developed a schematic diagram of the data-preprocessing steps (Fig. 6).

Various batch sizes (8, 16, and 32) were evaluated for each of the pretrained models. Ultimately, a batch size of 32 was chosen for all models. Table 2 presents the highest accuracy, as well as the best training and validation losses achieved by each model using a batch size of 32. Notably, the VGG19 model exhibited a training loss of 0.302 and a validation loss of 0.302 with an accuracy of 87.205, while the Inception v3 model achieved a training loss of 0.167, a validation loss of 0.230, and an accuracy of 91.421. The Xception model achieved a training loss of 0.519, a validation loss of 0.480, and an accuracy of 87.318. The Adam optimizer with a learning rate of 0.0001 was used in the initial run for all models.

To ensure robustness, each network was trained five times with the test results averaged for the first and fifth run (see Table 3). The dataset of tsunami images was subjected to a combination of zoom, shift, and brightness augmentation techniques during training. The learning rate for all models was adjusted using Keras *ReduceLROnPlateau*, which detects plateaus in model performance and adjusts the learning rate accordingly. Dropout regularization, which randomly ignores neurons during training, forcing others in the layer to take on the delay, was also implemented. Dropout rates of 0.5 and 0.8 were evaluated, with the latter performing best.

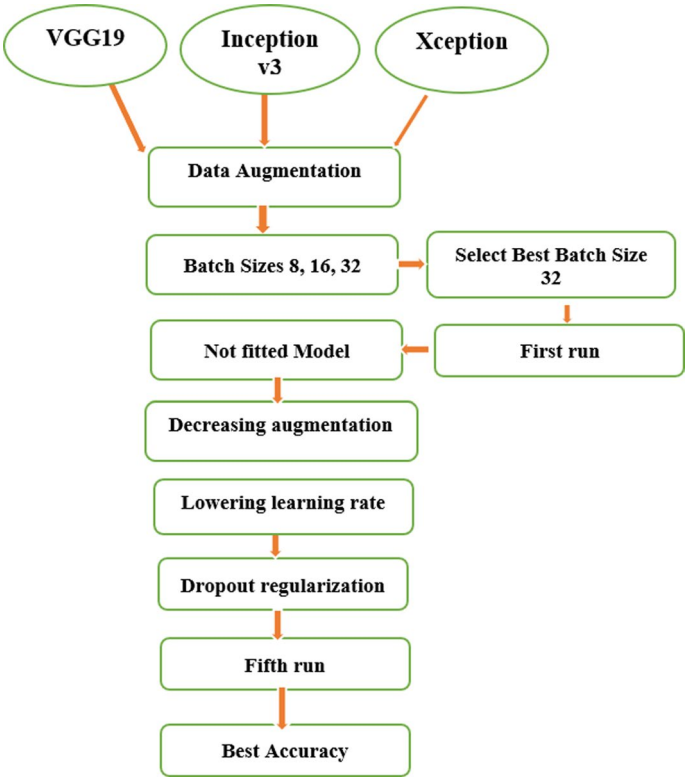


Fig. 6 Data preprocessing steps through a schematic diagram. The figure depicts different stages involved in preparing the raw data for machine learning

Table 2 Training results for various batch sizes (8, 16, 32) on multiple pretrained models, including train and validation loss and accuracy for each trial

Model	Batch size	Train loss	Val loss	Accuracy
VGG19	32	0.302	0.302	87.205
	16	0.306	0.301	86.352
	8	0.329	0.307	85.318
Iceptioii v3	32	0.167	0.230	91.421
	16	0.170	0.247	91.023
	8	0.187	0.249	91.068
Xception	32	0.519	0.480	87.318
	16	0.555	0.485	87.261
	8	0.556	0.486	87.240

Overall, the models displayed varying tendencies across the tsunami databases and exhibited different classification accuracies. Given the random nature of the algorithms or assessment procedure, or variations in numerical precision, our findings differed. To evaluate the models, we executed each five times and constructed line plots for the loss and accuracy of each model on both the train (green) and validation (blue) datasets for the first and fifth run (see Figs. 7 and 8).

Table 3 Averaged test results for the first and fifth run of experiments with the three pretrained models

Model	1st top accuracy	5th top accuracy	1st train loss	5th train loss	1st val loss	5th val loss
VGG19	82.057	85.307	0.302	0.267	0.300	0.291
Inception v3	87.739	89.477	0.167	0.155	0.240	0.230
Xception	87.545	89.170	0.519	0.501	0.490	0.430
	0.5	0.8	0.5	0.8	0.5	0.8
	Dropout					

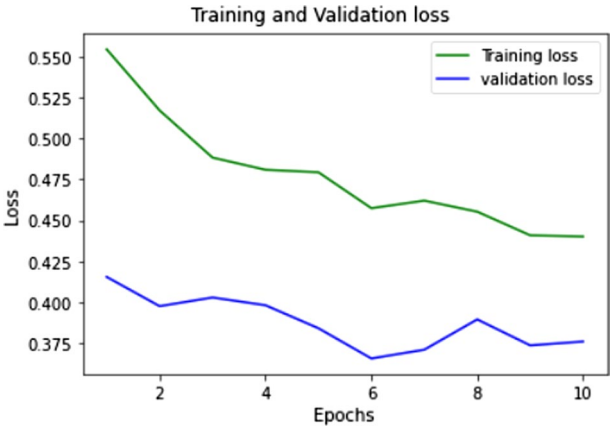


Fig. 7 First running training and validation loss curve of ***Vgg-9*** Model (2017)



Fig. 8 First running training and validation accuracy curve of ***Vgg-9*** Model

In the first run, the Vgg-19 model achieved an accuracy of approximately 82%, a first train loss of 0.302, and a first validation loss of 0.300 (see Table 3). The overall validation loss curve was better than the overall training loss curve, and the overall validation accuracy curve was better than the overall training accuracy curve. An obvious gap between the validation loss curve and the training loss curve, as well as the validation accuracy curve and the training accuracy curve, was observed, likely due to the data augmentation

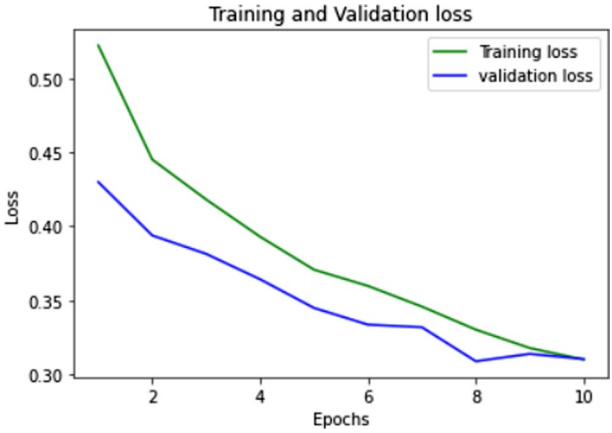


Fig. 9 Fifth running training and validation loss curve of ***Vgg-9*** Model fifth running training and validation loss curve of ***Vgg-9*** Model (2017)

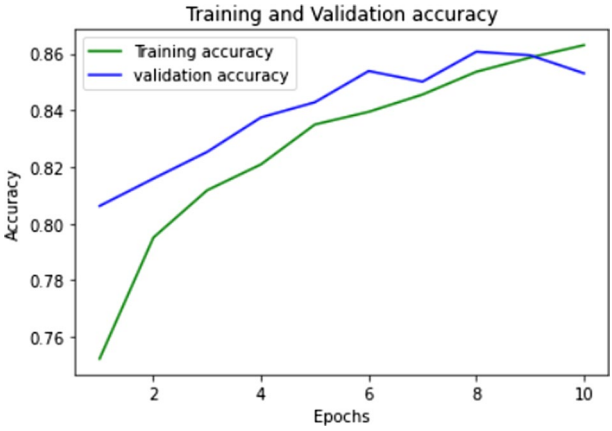


Fig. 10 Fifth running training and validation accuracy curve of ***Vgg-9*** Model (2017)

techniques applied to the training data, which made the task significantly harder for the network.

To improve the accuracies of our pretrained models, we removed random rotation and random zooming from our models' code and instead used adjusted brightness of images and a horizontal flip of images. The brightness adjustment made sense because the original images may have had less brightness, and the horizontal flip made sense because the image could have been taken from the left or right. This process was repeated five times, and the results were compared on both the train (green) and validation (blue) datasets. We observed that the validation loss curve and training loss curve in Fig. 9, as well as the validation accuracy curve and training accuracy curve in Fig. 10, became closer together, and the model achieved an accuracy of 85%.

In the fifth run, the model achieved a fifth train loss of 0.267 and a fifth validation loss of 0.291 (see Table 3). Figure 11 shows the final predicted feature maps of Myagi City for non-tsunami-stricken and tsunami-stricken areas in the fifth run.

Figure 12 shows that the Xception model was unable to achieve a suitably low error value on the validation set in the first model run, and the validation loss was greater than the training loss. The accuracy learning curve in Fig. 13 suggested that the model had

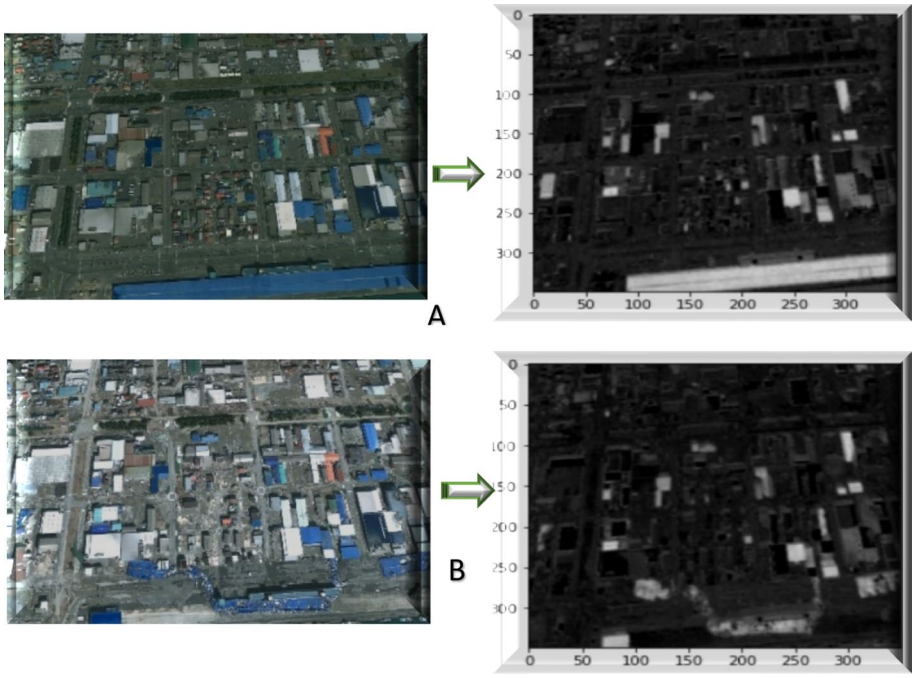


Fig. 11 Ground truth images and final predicted feature map s of Myagi City for **A** non-tsunami stricken and **B** tsunami stricken of VGG19

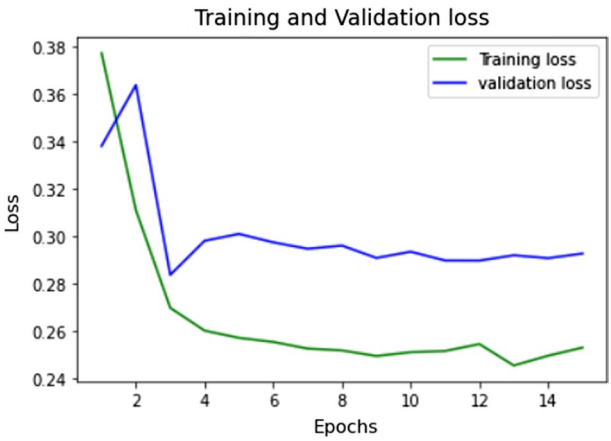


Fig. 12 First running training loss and validation loss curve of ***Xception*** Model (2017)

a high training accuracy compared to the validation accuracy, indicating that the model became unable to generalize to new data, resulting in an increase in generalization error. The model's averaged first run accuracy was about 87.5%, and Table 3 shows that the first train loss achieved 0.519 and the first validation loss achieved 0.490. This phenomenon often occurs because the model's capacity is greater than what is necessary for the

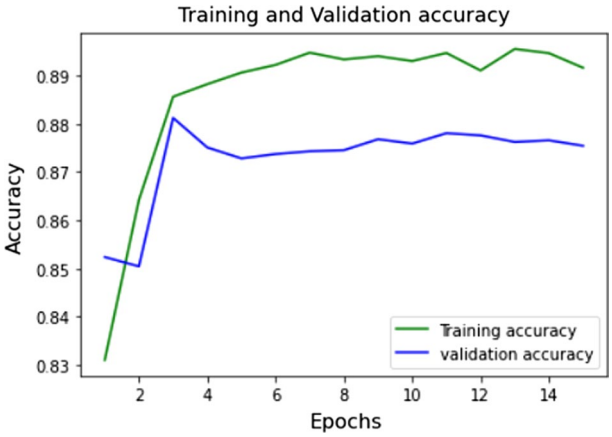


Fig. 13 First running training accuracy and validation accuracy curve of ***Xception*** Model (2017)

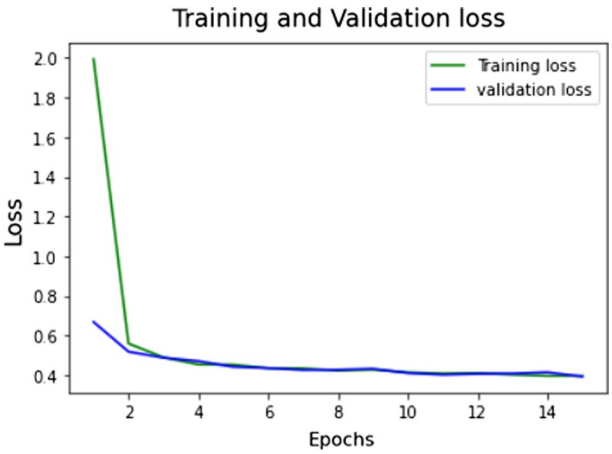


Fig. 14 Fifth running training and validation loss curve of ***Xception*** Model (2017)

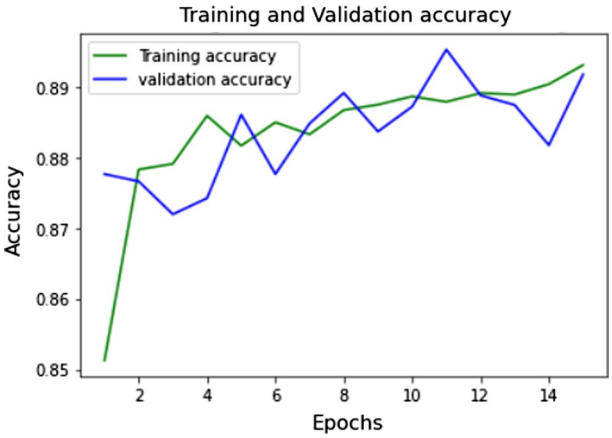


Fig. 15 Fifth running training accuracy and validation accuracy curve of ***Xception*** Model

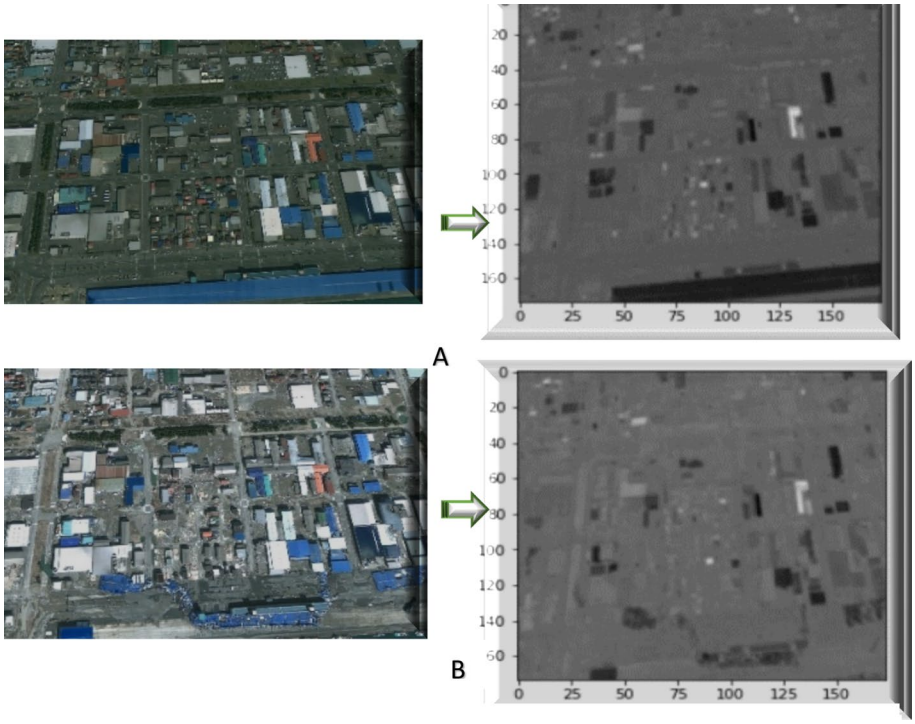


Fig. 16 Ground truth images and final predicted feature map of Myagi City for **A** non-tsunami stricken and **B** tsunami stricken for **Exception** model

problem, leading to far too much leniency. Additionally, the model may have been trained for an excessive amount of time.

In order to address the overfitting issue, we made several changes to our Xception model. First, we switched the optimizer from SGD to ADAM and lowered the learning rate to 0.0001. We also used the same *ImageDataGenerator* for both training and validation to ensure consistency. As a result, the gap between the training and validation loss curves was significantly reduced, as shown in Fig. 14. Similarly, the gap between the training and validation accuracy curves was also reduced, as shown in Fig. 15. The fifth run of the model achieved an accuracy of approximately 89.1%, with a fifth train loss of 0.501 and a fifth validation loss of 0.480, as presented in Table 3. The final predicted feature maps of Myagi City for non-tsunami-stricken and tsunami-stricken areas in the fifth run are displayed in Fig. 16.

The Inception-v3 model encountered a challenge with the interference between the training loss curve and validation loss curve, as depicted in Fig. 17. Additionally, Fig. 18 shows interference between the training accuracy curve and validation accuracy curve. This interference occurs because the model's complexity is insufficient for the neural network to understand the underlying mathematical mapping from input features to output labels. The model learns from the training data's noise and details to the extent that it reduces the model's performance on new data. The first accuracy achieved by the model was approximately 87.7%, with the first train loss of 0.167 and first validation loss of 0.240, as shown in Table 3. To address this issue, we employed a combination

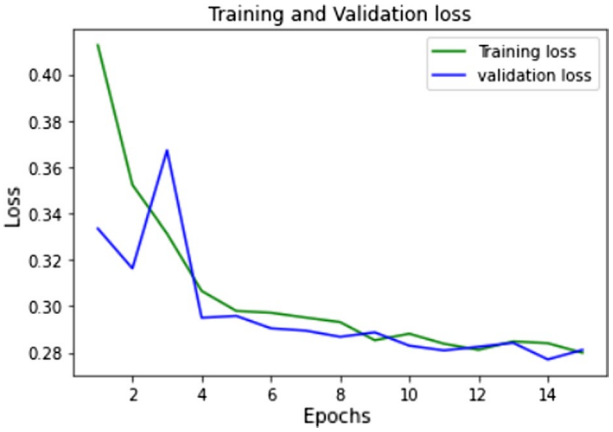


Fig. 17 First running training and validation loss curve of ***Inception-v3***Model (2017)

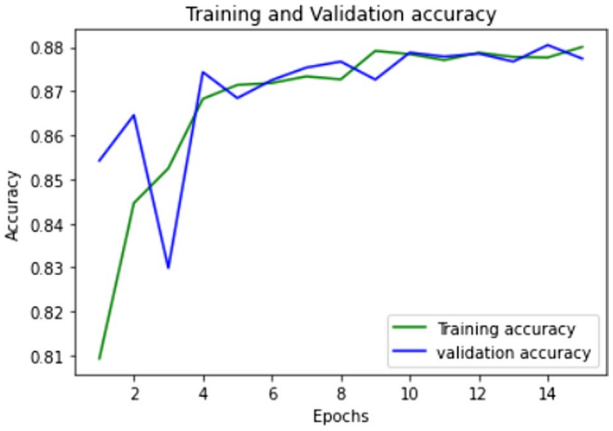


Fig. 18 First running training accuracy and validation accuracy curve of ***Inception-v3*** Model (2017)

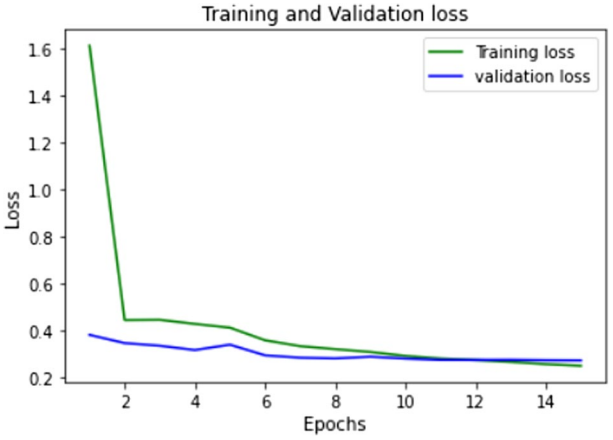


Fig. 19 Fifth running training and validation loss curve of ***Inception-v3*** Model (2017)

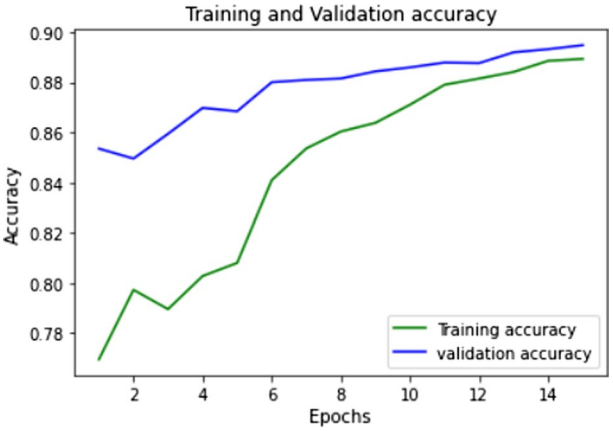


Fig. 20 Fifth running training accuracy and validation accuracy curve of ***Inception-v3*** Model (2017)

Table 4 Final run number of parameters and the execution time of the three pretrained models

Model	Ex cut km Time	Number of parameters
VGG19	30.5	32,870,465
Inception v3	24.1	74,233,633
Xception	45.3	72,242,729

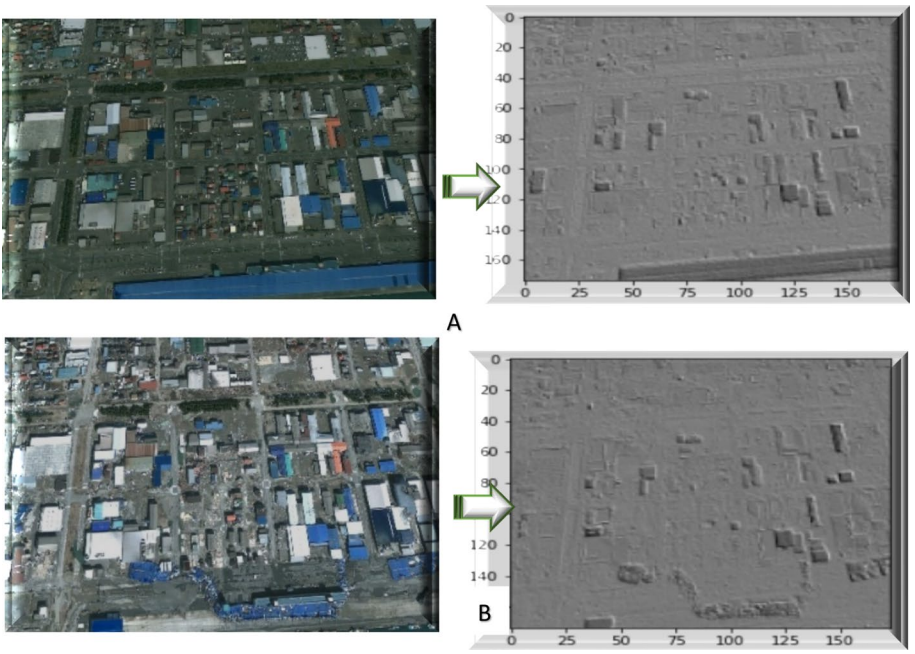


Fig. 21 Ground truth images and final predicted feature map of Myagi City for **A** non-tsunami stricken and **B** tsunami stricken for Inception v3

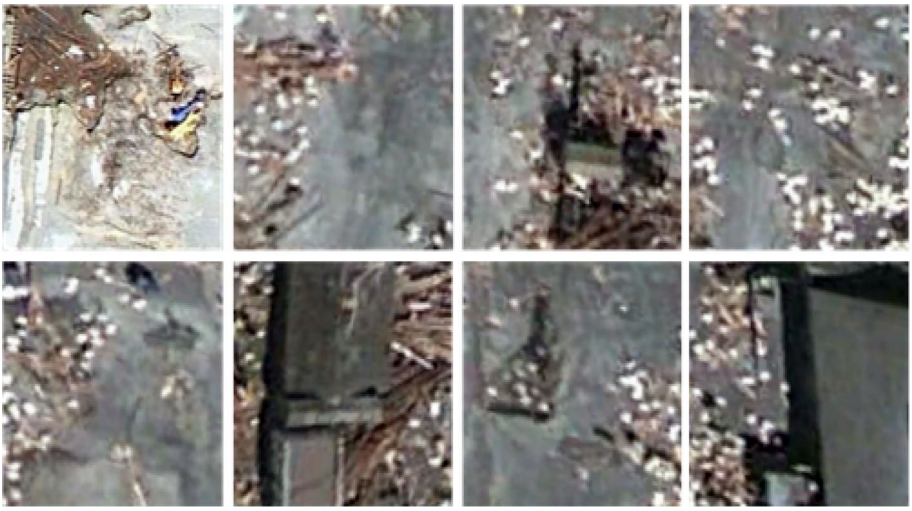


Fig. 22 Predicted images for ***stricken*** regions using ***Inception-v3*** model (2017)

of techniques, including data augmentation and adjusting the learning rate, to improve the model's performance. The fifth train loss was 0.213, and the fifth validation loss was 0.220. The model achieved an accuracy of about 91.5%, as shown in Fig. 19.

To optimize the performance of the Inception-v3 model, we made some modifications. First, we increased the number of nodes in the trainable fully connected layer from 256 to 1024, which improved the final validation accuracy. Additionally, we increased the dropout rate from 0.5 to 0.8, which reduced over-fitting and minimized the gap between training and validation accuracy. These modifications led to a training loss plot that stabilized and a validation loss plot that declined to a point of stability. The gap between training and validation loss decreased significantly as shown in Figure 19. The validation accuracy plot also declined to a point of stability and had a much smaller



Fig. 23 The predicted images for ***non-stricken*** regions using ***Inception-v3*** model (2017)

gap with training accuracy as shown in Figure 20. The Inception-v3 model achieved an excellent final accuracy of about 89.4% with fifth train loss of 0.155 and fifth Val loss of 0.230 as shown in Table 3.

The inference time of the fastest model, Inception-v3, is half as fast as the slowest model, Xception, and achieves an excellent compromise between quality and inference time. In fact, it performs better than the other models on the dataset and shows the best overall score of 24.11 minutes, while all other models scored between 30.50 min for Vgg19 and 45.33 min for Xception as shown in Table 4. Figure 21 shows the final predicted feature maps of Myagi City for non-tsunami-stricken and tsunami-stricken areas in the fifth run for Inception-v3's final run.

The final predicted categories of our chosen Inception-v3 model consist of binary classes for two types of regions—stricken and non-stricken, as shown in Figs. 22 and 23, respectively.

3.1 Evaluation of models

Upon analyzing the precision, recall, and *F*-1 scores of the three models, Inception v3 achieved the best overall performance. Table 5 shows that Inception v3 obtained a higher precision score of 0.92 for stricken, 0.94 for non-stricken, and a total average of 0.93 compared to Vgg19 and Xception. Vgg19 obtained the highest precision score of 0.85 for stricken, 0.87 for non-stricken, and 0.86 for the total average, while Xception obtained a precision score of 0.88 for stricken, 0.89 for non-stricken, and 0.89 for the total average.

Inception v3 outperformed the other two models in the Recall score as well, achieving a recall score of 0.94 for stricken, 0.95 for non-stricken, and a total average of 0.95. As *F*-1 score is calculated based on precision and recall, it naturally came better for Inception v3. Numerically, Inception v3 achieved a total average *F*-1 score of 0.94, outperforming Vgg19 and Xception, which achieved *F*-1 scores of 0.88 and 0.87, respectively.

Therefore, it can be concluded that Inception v3 achieved the best overall performance based on precision, recall, and *F*-1 scores. This analysis suggests that Inception v3 model is the best fit for this particular problem.

3.2 Deeper comparisons against previous researches

Real-time tsunami forecasting is crucial for mitigating the potential damage caused by such disasters, and ML algorithms can play a vital role in this regard. However, generating real training data for such algorithms is challenging, and most studies in this field rely on synthetic data. Firstly, Previous research in this field mostly relied on traditional numerical modeling based on individual scenarios, which has many drawbacks and uncertainties, including aleatory, epistemic, and computational costs. Our study offers a new approach using ML algorithms trained on real tsunami images. We investigate the impact of various parameters and learning rates to detect small objects, demonstrating the suitability of our approach for tsunami damage assessment.

To highlight the significance of our study, we compared it with others in the field as follows:

Table 5 Tsunami two destruction categories (Binary classes: Stricken and Non-stricken areas) performance metrics (precision, recall and *F*1_Score for all pertained models

Class	Precision			Recall			F1_Score		
	VGG19	Inception v3	Xception	VGG19	Inception v3	Xception	VGG19	Inception v3	Xception
Stricken	0.85	0.92	0.88	0.87	0.94	0.88	0.88	0.93	0.86
Non-stricken	0.87	0.94	0.89	0.88	0.95	0.90	0.88	0.95	0.89
Avg/total	0.86	0.93	0.89	0.88	0.95	0.89	0.88	0.94	0.87

1. Bhattacharya et al. (2019) proposed a method for identifying tsunami-affected areas using pre- and post-tsunami satellite images. They used Support Vector Machine (SVM) and Random Forest (RF) classifiers to classify the images. However, their approach was limited by the small size of their dataset, and they did not compare their results with other state-of-the-art methods. In contrast, our proposed method uses a larger dataset and state-of-the-art pre-trained deep neural networks, which outperform traditional machine learning algorithms.
2. Guo et al. (2020) used deep learning and high-resolution satellite imagery to assess the damage caused by the 2019 Typhoon Hagibis in Japan. They also used binary classification, but their approach was based on a convolutional neural network (CNN) and did not employ pre-trained models. We employ pre-trained models that have been previously trained on large datasets, allowing us to transfer the knowledge to our specific problem of tsunami damage detection Their method achieved an accuracy of 84.4%, which is lower than the accuracy achieved in our study.
3. Liu et al. (2021) focused on the detection of building damage caused by earthquakes using high-resolution satellite imagery and a CNN-based method. They achieved an accuracy of 89.8%, which is higher than the accuracy achieved by Guo et al. (2020) but lower than the accuracy achieved in our study. However, their method was not designed specifically for tsunami damage detection. we develop a custom network architecture that is specifically designed for tsunami damage detection using high-resolution remote sensing data, which improves the accuracy of automated binary classification.
4. Makinoshima et al. (2021) presented a tsunami inundation forecasting approach based on a CNN trained with synthetic tsunami scenarios, and they verified the approach against both synthetic tsunamis and actual tsunami observations from the 2011 Tohoku earthquake. They used a single CPU node and trained the model on 10,000 synthetic data cases + 1000 validations + 1000 tests, while we used 30,000 real tsunami images for training- 8800 validation and 2000 for testing, and completed the training process in 24.1 min using GPU nodes.
5. Ji et al. (2018) suggested an unsupervised method based on feature selection and building orientation to determine which buildings are affected or unaffected following a disaster. They contrasted their unsupervised approach with one involving supervised learning (Support Vector Machines). However, their study used radar images instead of optical ones, which cannot distinguish or resolve multiple targets and can be interrupted by other signals.
6. Kalinicheva et al. (2018) proposed a deep architecture based on CNN and auto-encoders (AE) for change detection in image time series. However, their method proved to be

extremely susceptible to variations in luminosity and seasonal changes, making it unsuitable for our case study.

7. Khiali et al. (2019) suggested a non-neural network-based, unsupervised method that depends on tracking segmented objects over time. While intriguing, this method is challenging to use in situations where there are significant differences between two images, as in natural disasters like tsunamis. Our study used a supervised method that tends to be more accurate than unsupervised learning models because of the labeled dataset.

4 The limitations of this study

While our approach offers many advantages, it is important to acknowledge the limitations of our study. One of the limitations is that we relied on a single dataset for training and testing the model, which may limit the generalizability of our results. This means that the performance of our model may not be as accurate when applied to new and unseen datasets. Therefore, it is important to evaluate the proposed approach in various real-world scenarios to assess its true effectiveness.

Another limitation of our study is the potential for training sample distribution imbalance. This is a common issue in preparing training samples and can affect the accuracy of the network model. Therefore, it is important to ensure that the training data are diverse and representative of the real-world scenarios to avoid any potential bias or overfitting.

Furthermore, the use of satellite images for tsunami damage assessment presents several challenges such as high cluttering, varying illumination, and scale variation caused by different perspectives. These factors can reduce the accuracy of the model and lead to distorted estimates. Therefore, it is important to consider these limitations and take appropriate steps to address them in future studies.

In the view of this study, the proposed approach for real-time tsunami forecasting using machine learning algorithms and real tsunami images is a significant step toward mitigating the devastating effects of tsunamis. The promising results obtained from this study demonstrate the potential of this approach in generating accurate and timely forecasts to help save lives and minimize property damage. However, it is important to acknowledge the limitations of the study, particularly the reliance on a single dataset and the negative characteristics of satellite images. Therefore, further research is needed to improve the generalizability and performance of the proposed approach in various real-world scenarios. Overall, this study provides valuable insights and sets the foundation for future advancements in real-time tsunami forecasting using machine learning techniques.

5 Conclusion

In this groundbreaking research, we have successfully evaluated the damage caused by tsunamis in a remarkably short period of time, just 24.11 min, with limited resources and information. Our study utilized advanced Deep Learning techniques, incorporating pre-trained Convolutional Neural Networks (CNN) features for binary classification of high-resolution satellite image tiles. Our research has demonstrated a significant improvement in accuracy rates compared to traditional numerical models and standard methods that rely on hand-crafted features.

We proposed a transfer learning approach using the Inceptionv3 pre-trained model, which outperformed other models, not only in terms of features but also in classification accuracy. Our pre-trained neural network model exhibited low bias and high variance, which was a result of the large and varied training datasets and its sensitivity to specific examples used to train the model, thus making minimal assumptions about the computational functional form of the mapping function.

Finally, we compared our network with traditional and current deep learning-based approaches, demonstrating the superiority of our strategy due to its low bias and high variance datasets, leading to a skillful model for tsunami damage assessment.

We highly recommend the use of our optimized pre-trained model in the event of a tsunami in a new area. Furthermore, we encourage the establishment of specialized machine learning centers worldwide to monitor earthquakes and tsunamis. Our challenges are interconnected, and a collaborative response is crucial in addressing these global threats.

Funding The authors have not disclosed any funding.

Declarations

Conflict of interest No conflict of interest exists. We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Craig Citro GS, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems, 2015. Google Brain Team. Software available from tensorflow.org

Anaconda Software Distribution (2020) Anaconda Documentation. Anaconda Inc. Retrieved from https://docs.anaconda.com/

Arnold T, Tilton L (2017) Automatic building damage assessment using deep learning and ground level image data. In: Applied sciences: school of computing science pp 113–129. Retrieved from http://link.springer.com.ezp-prod1.hul.harvard.edu/chapter/10.1007/978-3-319-20702-5_8.

Bai Y, Mas E, Koshimura S (2018) Towards operational satellite-based damage-mapping using U-net convolutional network: a case study of 2011 Tohoku earthquake-Tsunami. Remote Sens 10(10):1626. https://doi.org/10.3390/rs10101626

Bhattacharya D, Shrestha S, Riggan B, Salinas V (2019) Deep learning for automatic detection of earthquake damage using optical satellite imagery. IEEE Trans Geosci Remote Sens 57(10):7581–7596

Casper da Costa-Luis SK, Larroque SK, Mary H, Altendorf K, Yorav-Raphael N, Korobov M, Ivanov I, Bargull M, Rodrigues N, Chen G, Dektyarev M, Mjstevens M, Pagel D, Zugnoni M, Charles J, Newey T, Malmgren S, Umer A (2020) tqdm: a fast, extensible progress bar for Python and CLI (v4.46.0) [Software]. Zenodo. https://doi.org/10.5281/zenodo.3783558

Cha Y, Choi JW, Büyüköztürk O (2017) Deep learning-based crack damage detection using convolutional neural networks. Comput Aided Civ Infrastruct Eng 32(5):361–378. https://doi.org/10.1111/mice.12263

Chini M, Piscini A, Cinti FR, Amici S, Nappi R, DeMartini PM (2013) Tohoku (Japan) Tsunami inundation and liquefaction investigated through optical, thermal, and SAR data. IEEE Geosci Remote Sens Lett 10:347–351

Chollet F, Others. (2015). Keras [Computer software]. GitHub. https://github.com/fchollet/keras.

Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Diederik J, Jimmy D (2014) The effects of sleep deprivation on cognitive performance. J Sleep Res 23(5):535–542. https://doi.org/10.1111/jsr.12145

Endo Y, Adriano B, Mas E, Koshimura S (2018) New insights into multiclass damage classification of Tsunami-induced building damage from SAR images. Remote Sens 10:2059. https://doi.org/10.3390/rs10122059

Fujita A, Sakurada K, Imaizumi T, Ito R, Hikosaka S, Nakamura R (2017) Damage detection from aerial images via convolutional neural networks. In: Proceedings of the 2017 fifteenth IAPR international conference on machine vision applications (MVA), Nagoya, Japan, pp 5–8

Guo H, Wang X, Guo T, Liu Z (2020) Research on fast damage assessment of sudden natural disasters based on deep learning. J Ambient Intell Humaniz Comput 11(9):3739–3751

Hashemi H, Abdelghany K (2018) End-to-end deep learning methodology for real-time traffic network management. Comput Aided Civ Infrastruct Eng 33(10):849–863. https://doi.org/10.1111/mice.12376

Hunter JD (2007) Matplotlib: a 2D graphics environment. Comput Sci Eng 9(3):90–95

Imamura F, Boret SP, Suppasri A, Muhari A (2019) Recent occurrences of serious tsunami damage and the future challenges of tsunami disaster risk reduction. Progress Disaster Sci 1:100009. https://doi.org/10.1016/j.pdisas.2019.100009

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach FR, Blei DM (Eds), Proceedings of the 32nd international conference on machine learning, ICML 2015, vol 37, Lille, France, pp 448–456)

Ji Y, Sumantyo JTS, Chua MY, Waqar MM (2018) Earthquake/Tsunami damage assessment for urban areas using post-event PolSAR data. Remote Sens 10:1088

Kalinicheva E, Sublime J, Trocan M (2018) Neural network autoencoder for change detection in satellite image time series. In: Proceedings of the 25th IEEE international conference on electronics, circuits and systems (ICECS 2018), Bordeaux, France, pp 641–642

Khiali L, Ndiath M, Alleaume S, Ienco D, Ose K, Teisseire M (2019) Detection of spatio-temporal evolutions on multi-annual satellite image time series: a clustering-based approach. Int J Appl Earth Obs Geoinf 74:103–119. https://doi.org/10.1016/j.jag.2018.08.013

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Proceedings of the neural information processing systems conference, Stateline, NV

Lin Y, Nie Z, Zhang H, Ma HW (2017) Structural damage detection with automatic feature-extraction through deep learning. Comput Aided Civ Infrastruct Eng 32(12):1025–1046. https://doi.org/10.1111/mice.12313

Liu Z, Xu Y, Xu J, Li Y, Li H, Li J (2021) A deep learning approach for automatic building damage assessment from high-resolution remote sensing images after natural disasters. Remote Sens 13(3):491

MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol 1. pp 281–297

Makinoshima F, Oishi Y, Yamazaki T (2021) Early forecasting of tsunami inundation from tsunami and geodetic observation data with convolutional neural networks. Nat Commun 12:2253. https://doi.org/10.1038/s41467-021-22348-0

McKinney W (2010) Data structures for statistical computing in Python. In: Proceedings of the 9th Python in science conference vol 445, pp 51–56

Minghelli A, Spagnoli J, Lei M, Chami M, Charmasson S (2020) Shoreline extraction from WorldView2 satellite data in the presence of foam pixels using multispectral classification method. Remote Sens 12(16):2664. https://doi.org/10.3390/rs12162664

Mori N, Takahashi T, Yasuda T, Yanagisawa H (2011) Survey of 2011 Tohoku earthquake tsunami inundation and run-up. Geophys Res Lett. https://doi.org/10.1029/2011GL047440

Nabian A, Meidani H (2018) Deep learning for accelerated seismic reliability analysis of transportation networks. Comput Aided Civ Infrastruct Eng 33(6):443–458. https://doi.org/10.1111/mice.12359

Ohta Y, Murakami H, Watoh Y, Koyama M (2004) A model for evaluating life span characteristics of entrapped occupants by an earthquake. In: Proceedings of the 13th world conference on earthquake engineering, Vancouver, BC

Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In Navab N, Hornegger JMWIII, Frangi AF (Eds), Medical image computing and computer-assisted intervention—MICCAI 2015—18th international conference, Munich, Germany, 5–9 October 2015; Proceedings, Part III, Lecture Notes in Computer Science vol 9351, Springer, pp 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Seide F, Agarwal A (2016) CNTK: Microsoft's open-source deep-learning toolkit. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining pp 2135–2135. Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. In: Proceedings of the international conference on learning representations (ICLR), San Diego, CA

Sublime J (2021) The 2011 Tohoku Tsunami from the sky: a review on the evolution of artificial intelligence methods for damage assessment. Geosciences 11(3):133. https://doi.org/10.3390/geosciences11030133

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1–9. IEEE. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Rabinovich A (2016a) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016b) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Wieland M, Liu W, Yamazaki F (2016) Learning change from synthetic aperture radar images: performance evaluation of a support vector machine to detect earthquake and Tsunami-induced changes. Remote Sens 8(10):792

Yang X, Li H, Yu Y, Luo X, Huang T, Yang X (2018) Automatic pixel-level crack detection and measurement using fully convolutional network. Comput Aided Civ Infrastruct Eng 33(12):1090–1109. https://doi.org/10.1111/mice.12412

Zhang A, Wang KCP, Li B, Yang E, Dai X, Peng Y, Chen C (2017) Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. Comput Aided Civ Infrastruct Eng 32(10):805–819. https://doi.org/10.1111/mice.12297

Zheng Y, Yang C, Merkulov A (2018). Breast cancer screening using convolutional neural network and follow-up digital mammography, In: Proc. SPIE 10669, Computational Imaging III, p 1066905. https://doi.org/10.1117/12.2304564

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.