International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I3P128

Volume 72 Issue 3, 318-330, March 2024

Original Article

Designing an Image Classification Model on Emergency Incident Images using a Convolutional Neural Network for iRESPOND

Freddie Prianes[1], Ichelle Baluis[2], Joseph Jessie Oñate[3], Challiz Omorog[4], Thelma Palaoag[5], Nancy Flores[6]

[1,2,3,4]*College of Computer Studies, Camarines Sur Polytechnic Colleges, Philippines.*
[5,6]*College of Information Technology and Computer Science, University of the Cordilleras, Philippines.*

[1]*Corresponding Author : fprianes@cspc.edu.ph*

Received: 28 November 2023 Revised: 15 February 2024 Accepted: 26 February 2024 Published: 17 March 2024

Abstract *- In the face of increasing natural disasters and emergencies, there is a growing need for effective geospatial information systems to process and classify emergency reports in real time. This work presents a new Convolutional Neural Network (CNN) model that is intended to classify emergency images taken and delivered to the iRESPOND system. Through the utilization of training phases and various tools, frameworks, and techniques, the authors effectively used deep learning to develop the CNN model. This model improves disaster response and mitigation by enabling the iRESPOND system to categorize emergency incidents rapidly. The results showcase the model's commendable performance, achieving a high accuracy of 95.02% on the test set. A comprehensive evaluation, including precision, recall, and F1-score metrics for individual classes, illuminates the model's strengths and areas for improvement. Noteworthy is the model's proficiency in classes such as 'flood', 'infrastructure_damage', 'no_damage_buildings_street', 'no_damage_ water_related', and 'no_damage_wildlife_forest', reflecting robust predictive capabilities in specific emergency scenarios. The interpretability of the CNN model is augmented through visualization techniques like LIME, Grad-CAM, and Grad-CAM++. Also, a visualization report featuring the original image alongside interpretability overlays provides information on the characteristics and areas of the original images that influence the model's decisions. In conclusion, the model demonstrates efficacy in rapidly categorizing emergency incidents, providing a valuable tool for the response team. The recommendations for future work underscore the continuous refinement required for optimal performance, including addressing class imbalances, fine-tuning hyperparameters, exploring ensemble models, and expanding the diverse image datasets.*

Keywords *- Emergency response, Machine learning, Resnet50, Tensorflow, Keras.*

1. Introduction

Communities all throughout the world struggle with effective Emergency Response Management (ERM). First responders have to deal with a range of situations, including crimes, traffic accidents, and fires. To reduce the risk to human life, they must respond swiftly to incidents. [1] In order to respond to emergencies and disasters such as fires, typhoons, terrorist attacks, civil unrest, car accidents, and the like, parts of the world, including the Philippines, are using the 911 emergency number as a reporting tool. [2] Moreover, the rise in the frequency and intensity of natural disasters and emergencies has heightened the urgency for advanced technologies to bolster crisis management capabilities. [3,4] In fact, there has been an alarming increase of 73.7% and 11.55% of geological and climate-related hazards from 2020 to 2021, a total of 79 thousand deaths and injuries during fire based on a 2021 survey from 38 participating countries, and 1.35 million worldwide fatalities on road accidents reported in 2022. [5-7] In response to this pressing need, geospatial information systems have emerged as indispensable tools for processing and classifying emergency reports in real-time. [8,9] As a subset of machine learning, deep learning has demonstrated remarkable success in image recognition and classification tasks. [10] By harnessing the power of deep learning, the CNN model excels in discerning intricate patterns and features within emergency images, outperforming conventional approaches.

This strategic use of deep learning aligns with the broader trend in disaster management research, where machine learning techniques are increasingly being embraced for their ability to handle the complexity of dynamic datasets. [11,12] While deep learning algorithms can offer positive outcomes on accuracy, they may be resource-intensive and inefficient

for real-time monitoring applications. [13] The study of Rathod, A. et al. showcases the effectiveness of CNN-based models in achieving high accuracy for disaster image classification; however, there is an inadequate foundation for developing a robust computerized process for disaster response and recovery management systems. [14]

Traditional methods of disaster classification lack precision and speed, which are required for timely decision-making and resource allocation during emergencies is the focus of the study of Shah J. et al., and there are also issues related to data privacy, data transfer latency, and centralized data storage that posed challenges in implementing effective disaster classification systems. [15] Therefore, there is a need to investigate the effectiveness of transfer learning techniques in addressing the lack of training data and improving model performance in deep learning applications, particularly in scenarios with limited datasets. [16]

Additionally, Asif, A. et al. highlight the potential of utilizing neural network-based image processing architectures to improve crisis-related activities. The paper also mentioned the limitations of studies in assessing activity, context, and other related images in emergencies and disasters. At the same time, Tang et al. identify the limitations of existing forest classification algorithms based on graphics analysis, Kallas J., and Napolitano R. on sub-classifying a specific and complex type of structural damage like cracks, and Daly, S. and Thom, J. on the difficulty on recognizing fire and smoke in images. [13,17-19] On the other hand, the study of Mukhopadhyay A. et al. has identified that future research in ERM prediction may also focus on assessing the accuracy of emergency prediction models; additional modelling and empirical studies are needed to comprehend the advantages and drawbacks of these methods fully. [1]

Amidst these challenges, developing the models for emergency and disaster-related applications poses positive results. In the study of Sharma, N., Jain, V., and Mishra, A., highlighted the importance of testing CNNs on multiple datasets to reveal their true capabilities and limitations. Although they have observed that GoogLeNet and ResNet50 outperformed AlexNet in terms of precision in recognizing objects in images, there are still significant variations in the performance of the trained CNNs across different categories of objects.[20]

With the same set of models, the study by Zainorzoli, S.M., et al., the ResNet50 model also achieved the highest accuracy. [21] Likewise, Sushma, L., and Lakshmi, K. P., proved that the ResNet50 model had the highest accuracy, but it was compared to VGG16 and VGG19. [22] The analysis consistently points towards ResNet50 as a superior performer in terms of accuracy. Through comparative evaluations against other popular CNN architectures like AlexNet, GoogLeNet, VGG16, VGG19, and others, ResNet50 consistently demonstrates superior precision and reliability in recognizing objects within images. These corroborated across diverse datasets and applications, which underscores its capability to achieve high levels of accuracy, making it a preferred choice for image classification tasks, including emergency incident image classification.

Generally, the related studies contributed to diverse methodologies and applications to the field of disaster prediction and response, ranging from advanced machine learning models to innovative technological solutions. However, further research and collaboration are needed to address the complex challenges and gaps in diverse datasets of images for different classifications of emergencies and disasters, higher accuracy on prediction rate, and real-time processing of incident reports in disaster response and mitigation. Hence, the papers contributed to this study by introducing a pioneering CNN model tailored for the iRESPOND system.

The iRESPOND system, a key player in emergency data management, receives and processes diverse information streams, including images captured during critical situations. Traditional methods of image analysis often fall short in terms of speed and accuracy, necessitating the integration of advanced machine-learning techniques. This proposed study is strategically designed to address this challenge, leveraging deep learning capabilities to enhance the system's image classification capabilities.

The primary goal of designing this model is to expedite the classification process of emergency incidents based on visual information extracted from images sent to the system. This accelerated categorization process translates into a substantial reduction in response times, aligning with the global goal of achieving more resilient and adaptive disaster management systems. [23] Moreover, the enhanced image classification capability offers the advantage of assessing emergency situations, enabling responders to address impending crises proactively.

The significance of this research lies in its opportunity to contribute to the wider improvement of disaster-mitigation response efforts. By augmenting the iRESPOND system's capabilities through the integration of the CNN model, the authors are not only addressing an immediate need for real-time image classification but also laying the foundation for future advancements in technologically sophisticated disaster management frameworks.

Therefore, this paper adds a valuable dimension to the evolving field of geospatial information systems in emergency management. The introduction of a novel CNN model, informed by deep learning principles, reflects a strategic response to the escalating challenges posed by natural disasters and emergency situations. By enhancing the

iRESPOND system's image classification capabilities, the research strives to play a pivotal role in the ongoing pursuit of effective, early and informed disaster response and mitigation strategies.

2. Tools, Frameworks, and Techniques

This section provides the training tools, frameworks, and techniques for the efficient development of the CNN model, which will contribute to the system's overall effectiveness in providing early warnings and improving disaster response and mitigation efforts.

2.1. TensorFlow

This open-source machine learning framework offers a thorough platform for developing and deploying neural network models. Because of its flexibility, scalability, and support for a broad spectrum of applied machine learning, it is extensively utilized in both academics and industry. [24]

2.2. Keras

An open-source Python API for high-level neural networks that functions as a user-friendly interface for neural network development and training. Keras is a well-liked option for rapid deep-learning model construction and prototyping, as it can operate on top of TensorFlow. [25]

2.3. ResNet50 - Residual Network with 50 layers

A convolutional neural network architecture, which is known for its deep structure and the use of residual learning, helps mitigate the vanishing gradient problem in very deep networks. ResNet50 has been widely adopted for image classification tasks due to its excellent performance and ability to train deep networks effectively. [26]

2.4. Adam - Adaptive Moment Estimation

Another popular optimization algorithm combines ideas from Root Mean Squared Propagation (RMSProp) and momentum optimization. For every parameter, it keeps track of two moving averages: the mean (first instant) and the uncentered variance (second moment).

The learning rates for each parameter are then adaptively adjusted using these moving averages. Adam is known for its robustness across different types of neural network architectures and has become a default choice for many deep learning applications. [27]

2.5. Local Interpretable Model - Agnostic Explanations (LIME)

A method for providing context for machine learning models' predictions. It functions by varying the data and tracking how the predictions change. This method is especially useful for understanding the decision boundaries of complex models like neural networks. [28]

2.6. Gradient - Weighted Class Activation Mapping (Grad-Cam)

Grad-CAM is an interpretive and visual method for deep neural network recommendations, especially in convolutional neural networks. It generates heatmaps that show the areas of an input image that are significant in determining the final classification. Grad-CAM offers information about the areas in which the network is most active. [29]

2.7. Gradient - Weighted Class Activation Mapping Plus Plus (Grad-Cam++)

On the other hand, Grad-CAM++ enhances the localization of important areas in an image. It incorporates a weighted combination of positive and negative gradients to improve localization accuracy. This refinement makes Grad-CAM++ particularly effective in providing detailed and accurate visual explanations for model predictions. [30]

The iRESPOND system employs the ResNet50 architecture as the backbone of its Convolutional Neural Network (CNN). ResNet50's deep structure and residual learning capabilities enhance the model's capacity to identify complex patterns and features of emergency images sent to the iRESPOND system. TensorFlow serves as the underlying framework for implementing the CNN model, while Keras provides a user-friendly interface for designing and training the neural network. This combination allows for efficient development, training, and deployment of the deep learning model within the iRESPOND architecture.

Moreover, Adam is used for optimization in order to handle non-stationary objectives and noisy gradients effectively. This is particularly advantageous in scenarios where emergency images might exhibit diverse and dynamic features.

In addition, LIME is employed for model interpretation and explanation. By varying the emergency images supplied and tracking predicted alterations, LIME helps provide local, interpretable insights into how the CNN model makes decisions. Finally, Grad-CAM and Grad-CAM++ are utilized as visualization techniques to generate heatmaps highlighting areas of interest in emergency images. These visualizations aid in understanding which image elements have the greatest influence on the CNN model's ability to make decisions crucial for validating the model's focus on relevant features during the classification of emergency incidents.

3. Methodology

In developing and fine-tuning the model for the iRESPOND system, as depicted in Figure 1, uses the training phases are used as follows to provide a detailed account of the steps employed in order to ensure the model's efficacy for rapidly categorizing emergency incidents based on visual information.

Fig. 1 Training phases (Methodology)

3.1. Image Dataset

The choice of an image dataset is a critical decision in CNN development. With its extensive and diverse collection of labelled images, a dataset provides a rich environment for training a CNN to recognize intricate patterns and features across various classes. [31] This diversity ensures that the model generalizes well to new, unseen data.

3.2. Global Variable Modification

Global variable modifications, such as setting a generic seed, specifying epochs, and adjusting the learning rate, play a significant part in the reproducibility and optimization of the model training process. The concept of hyperparameter tuning emphasizes the importance of systematically adjusting these parameters to achieve optimal model performance. [32]

3.3. Reading and Decoding

Efficient data input is crucial for optimizing CNN training. TensorFlow's data input pipelines guide the decoding and pre-processing of images. Creating a streamlined and effective data pipeline ensures the model is fed with properly formatted and processed input during training. [24]

3.4. Partitioning Dataset

The dataset partitioning into train split, valid split, and test split is a crucial step in model evaluation. It emphasizes the significance of proper dataset splitting to ensure unbiased assessments of a model's generalization performance. This strategic partitioning is fundamental for assessing how well the model performs on new, unseen data. [33]

3.5. Model Building

Building the CNN model involves careful consideration of architecture, hyperparameters, and layers. Leveraging pre-trained models like ResNet50 is a common practice in transfer learning. [26] This approach allows the model to benefit from previously learned features, which is particularly helpful when handling limited labelled data. Keras, as a high-level neural networks API, simplifies the process of specifying and building complex CNN architectures. [25]

3.6. Performance Plotting

Understanding the model's behavior during training requires being able to visualize its performance. Utilizing libraries like Matplotlib, the plotting of training and validation on accuracy and loss over epochs presents insight into the convergence and possible overfitting of the model. These visualizations are essential for making informed decisions about model adjustments. [34]

3.7. Model Result and Visualization

The observed results, encompassing model performance metrics and any noteworthy findings serve as the culmination of the CNN development process. Powers' work on evaluation metrics in reference offers an extensive range of metrics that collectively provide a thorough assessment of the model's classification performance. [35] The discussion of results should focus on accuracy and the model's ability to generalize to diverse scenarios. Alongside visualization techniques, this can help interpret the model's decisions by highlighting important areas in the input images. [26]

4. Results and Discussion

This provides a comprehensive evaluation of the CNN-based image classification model developed within the iRESPOND System, focusing on its modelling approach, performance analysis, and implementation.

4.1. Modelling Approach

4.1.1. Image Datasets

The dataset encompasses a diverse range of disaster-related scenarios, with each image meticulously labelled to facilitate the model's learning process. The dataset comprises image labels and their corresponding counts. As shown in Table 1, each label represents a distinct category of emergency or disaster-related scenarios, ranging from natural disasters like earthquakes and floods to human-inflicted accidents and various types of infrastructure and environmental damage. The dataset's strength lies in its diversity, encompassing a substantial number of images for each category, which is essential for training the CNN model.

Table 1. Disaster image dataset

Label/Category	Image Count
1. accident_human_inflicted	241
2. earthquake	37
3. el_niño	202
4. flood	1,036
5. infrastructure_damage	1,419
6. landslide	457
7. no_damage_buildings_street	4,573
8. no_damage_human	121
9. no_damage_water_related	2,275
10. no_damage_wildlife_forest	2,272
11. urban_fire	420
12. wild_fire	515

4.1.2. Global Variable Modification

The global variables play a vital role in configuring and controlling various aspects of the model in the iRESPOND system. As shown in Figure 2, each variable is carefully defined to influence different components of the model's architecture, training, and evaluation processes of each global variable:

CLASSES

The CLASSES variable defines the different categories or classes present in the dataset. Each class corresponds to a specific type of emergency incident or disaster scenario. This list is crucial for model output interpretation, evaluation, and ensuring that the CNN can predict within the defined classes.

SEED

This variable sets the random seed for various stochastic processes in the model. A fixed seed, which is 68765 in this case, ensures reproducibility during training, essential for consistent results when experimenting with the model or comparing different iterations.

TRAIN_SPLIT, VALID_SPLIT, TEST_SPLIT

This defines the proportion of the images allocated to train, validate, and test, respectively. Proper splitting guarantees that a diverse dataset is utilized to train the model, validated on distinct examples, and tested on unseen instances, promoting generalization.

IMAGE_SHAPE_2D, IMAGE_SHAPE_3D

The CNN model expects the input images to have these dimensions. Here, images are planned to have either a 2D shape (grayscale) or a 3D shape (RGB). Consistent image dimensions are critical for compatibility with the model architecture.

DIRECTORIES

These variables specify the paths to different directories for storing and organizing the dataset. The source directory contains the original dataset, while the refactored directory holds the pre-processed and augmented data.

The training, validation, and test directories store subsets of the data for their respective purposes.

EPOCHS

This establishes the iteration count or passes through the entire training dataset during training. A carefully chosen number of epochs ensures the model converges to an optimal state without overfitting or underfitting.

LEARNING_RATE

This variable determines each iteration's step size upon updating the model parameters. It is a critical hyperparameter influencing the convergence and stability of the training process.

BASE_MODEL

The BASE_MODEL defines the ResNet50 serving as a pre-trained backbone model that leverages transfer learning, allowing the model to benefit from features learned on large-scale image datasets.

PREPROCESSING_METHOD

This method variable specifies the pre-processing function carried out to input images before providing them to CNN. In this case, the ResNet50-specific pre-processing method is employed to ensure compatibility with the pre-trained ResNet50 model.

OPTIMIZER

This defines the optimization algorithm used during training. In this instance, the Adam optimizer with a specified learning rate is chosen for its effectiveness in optimizing deep neural networks. Each of these global variables contributes to the overall flexibility, efficiency, and adaptability of the model, which reflects careful consideration of diverse factors crucial for successful model training and deployment.

4.1.3. Reading/Decoding and Partitioning Dataset

Figure 3 shows these are the scripts used to prepare and organize the image dataset for training the model. It is worth noting that this script performs dataset partitioning for training, validation, and testing and organizes the data into directories suitable for training a CNN model. The `prime_dataset` function could be used as part of a broader data preparation pipeline before training the iRESPOND system's CNN model. Additionally, the script uses shell commands (`ls`) and system utilities (`shutil`) for file operations, which can vary in compatibility across different operating systems.

Read Each Image With its Class Label

- This part of the code iterates through each class in the `CLASSES` list.
- It uses the `ls` command to list the files in the specified directory (`SOURCE_DIRECTORY + folder`).
- The file names are split using a regular expression to extract individual image names.
- The extracted class-label pairs are appended to the `images` list.

Train, Validate, and Test Partitioned Images

- Train, validate, and test datasets directories are generated for each class.
- Scripts then iterates through each class:
  - Randomly selects a fraction of the images for training and moves them to the respective training directory.
  - Randomly selects a fraction of the remaining images for validation and moves them to the respective validation directory.
  - The remaining images are moved to the testing directory.

```
CLASSES = ('accident_human_inflicted',
           'earthquake',
           'el_niño',
           'flood',
           'infrastructure_damage',
           'landslide',
           'no_damage_buildings_street',
           'no_damage_human',
           'no_damage_water_related',
           'no_damage_wildlife_forest',
           'urban_fire',
           'wild_fire')

SEED = 68765

TRAIN_SPLIT = 0.7
VALID_SPLIT = 0.2
TEST_SPLIT = 0.1

IMAGE_SHAPE_2D = (224, 224)
IMAGE_SHAPE_3D = (224, 224, 3)

SOURCE_DIRECTORY = './assets/disaster_data/'
REFACTORED_DIRECTORY = './assets/refactored_data/'
TRAIN_DIRECTORY = './assets/refactored_data/train/'
VALID_DIRECTORY = './assets/refactored_data/valid/'
TEST_DIRECTORY = './assets/refactored_data/tests/'

EPOCHS = 50

LEARNING_RATE = 0.001

BASE_MODEL = ResNet50(weights='imagenet', include_top=False, input_shape=IMAGE_SHAPE_3D)
PREPROCESSING_METHOD = preprocessing_function=tf.keras.applications.resnet50.preprocess_input

OPTIMIZER = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 3s 0us/step
```

Fig. 2 Global variables (Code Snippet)

```
def prime_dataset():
  # Read Each Image With its Class Label
  images = []
  folders=CLASSES

  for folder in folders:
    t = folder
    x = !ls $SOURCE_DIRECTORY$t
    for i in x:
      for j in re.split(r'[-;,\t\s]\s*', i):
        if j == '':
          continue
        images.append({'Class':t,'Image':j})


  # Partition Images into Traning, Validation, and Testing
  for c in folders:
      os.makedirs(f'{TRAIN_DIRECTORY}{c}', exist_ok=True)
      os.makedirs(f'{VALID_DIRECTORY}{c}', exist_ok=True)
      os.makedirs(f'{TEST_DIRECTORY}{c}', exist_ok=True)

  counter=0
  for c in folders:
      numOfFiles = len(next(os.walk(f'{SOURCE_DIRECTORY}{c}/'))[2])
      for files in random.sample(glob(f'{SOURCE_DIRECTORY}{c}/*'), int(numOfFiles*TRAIN_SPLIT)):
          shutil.move(files, f'{TRAIN_DIRECTORY}{c}')

      for files in random.sample(glob(f'{SOURCE_DIRECTORY}{c}/*'), int(numOfFiles*VALID_SPLIT)):
          shutil.move(files, f'{VALID_DIRECTORY}{c}')

      for files in glob(f'{SOURCE_DIRECTORY}{c}/*'):
          shutil.move(files, f'{TEST_DIRECTORY}{c}')
      counter+=1

  shutil.rmtree(SOURCE_DIRECTORY)
```

Fig. 3 Reading/Decoding and partitioning dataset (Code Snippet)

```python
def build_model(measure_performance:bool = True):
  ImageFile.LOAD_TRUNCATED_IMAGES = True

  train_batches = ImageDataGenerator(preprocessing_function=PREPROCESSING_METHOD).flow_from_directory(directory=TRAIN_DIRECTORY, target_size=IMAGE_SHAPE_2D, classes=CLASSES, batch_size=128)
  valid_batches = ImageDataGenerator(preprocessing_function=PREPROCESSING_METHOD).flow_from_directory(directory=VALID_DIRECTORY, target_size=IMAGE_SHAPE_2D, classes=CLASSES, batch_size=128, shuffle=False)
  test_batches =  ImageDataGenerator(preprocessing_function=PREPROCESSING_METHOD).flow_from_directory(directory=TEST_DIRECTORY, target_size=IMAGE_SHAPE_2D, classes=CLASSES, batch_size=128, shuffle=False)

  input_shape = IMAGE_SHAPE_3D
  nclass = len(CLASSES)
  epoch = EPOCHS
  base_model = BASE_MODEL
  base_model.trainable = False

  add_model = Sequential()
  add_model.add(base_model)
  add_model.add(Layer())
  add_model.add(GlobalAveragePooling2D())
  add_model.add(Dropout(0.5))
  add_model.add(Dense(nclass, activation='softmax'))

  model = add_model
  model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=LEARNING_RATE), loss='categorical_crossentropy', metrics=['accuracy'])
  es = EarlyStopping(monitor='val_loss', mode='auto', verbose=1 ,  patience = 10)


  fitted_model= model.fit(x=train_batches, validation_data=valid_batches, epochs=epoch, callbacks=[es])
  score, accuracy = model.evaluate(x=test_batches, batch_size=128)

  print(Fore.GREEN + u'\n\u2713 ' + f'Accuracy ==> {accuracy}')
  print(Fore.GREEN + u'\n\u2713 ' + f'Loss ==> {score}')

  plt.rcParams["figure.figsize"] = (15,8)

  if measure_performance:
    plt.plot(fitted_model.history['accuracy'])
    plt.plot(fitted_model.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    plt.plot(fitted_model.history['loss'])
    plt.plot(fitted_model.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    y_pred = model.predict(test_batches)


    ax = sns.heatmap(confusion_matrix(test_batches.classes, y_pred.argmax(axis=1)), annot=True, cmap='Blues', fmt='g')
    ax.set_title('Confusion Matrix')
    ax.set_xlabel('Predicted Values')
    ax.set_ylabel('Actual Values')
    ax.xaxis.set_ticklabels(CLASSES)
    ax.yaxis.set_ticklabels(CLASSES)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    plt.show()


    labels = {value: key for key, value in train_batches.class_indices.items()}
    print("Label Mappings for classes present in the training and validation datasets\n")
    for key, value in labels.items():
      print(f"{key} : {value}")

    print(classification_report(test_batches.classes, y_pred.argmax(axis=1), target_names=labels.values()))


  return model
```

Fig. 4 Model building and performance plotting (Code Snippet)

Remove Source Directory

- Once the images are moved to their respective training, validation, and testing directories, the original source directory is removed using `shutil.rmtree (SOURCE_DIRECTORY)`.

4.1.4. Model Building and Performance Plotting

The following function, as shown in Figure 4, serves as a comprehensive tool for building, training, and evaluating the iRESPOND system's CNN model, providing key points into its performance through various visualizations and metrics.

Data Preparation

- The script uses `ImageDataGenerator` to create data generators for training, validation, and testing sets. These generators pre-process images on the fly.

Model Configuration

- The script configures the input shape, number of classes, and number of epochs and sets the base model (ResNet50) as non-trainable.

Model Architecture

- The CNN model is constructed using the ResNet50 base model, global average pooling, dropout, and a dense softmax layer.

Model Compilation and Training

- Compiling the model through Softmax loss and Adam optimizer. The fit method is used in training.

Performance Metrics and Evaluation

- Accuracy and loss are printed on the test set after model evaluation.

- If `measure_performance` is set to True, various performance metrics are plotted, including accuracy and loss curves, a confusion matrix, and a classification report.

Return Statement

- The function returns the trained model.

4.1.5. Model Outputs and Visualization

Upon training the model using a diverse dataset comprising images captured during emergency situations, the model output provided an overview of the model's performance metrics (accuracy, precision, recall, and F1-score) evaluated on both training and validation datasets. These provide information about the model's overall effectiveness in classifying emergency incident images. Visualizations such as confusion matrices which illustrate the distribution of predicted classes compared to ground truth labels on the image datasets provided. These matrices identified systematic errors or biases in the model's predictions for further refinement and optimization. Additionally, techniques such as mapping are used to see the areas of interest within input images that have contributed to the model's classification decisions. The heatmaps present significant insights into the characteristics and patterns the model identifies as indicative of different emergency incident categories.

4.2. Performance Analysis

A total of 9,484 images were used in the training set, 2,707 images in the validation set, and 1,366 images in the test set to train the model under 50 epochs on a dataset with 12 classes. Throughout the training, the model demonstrated a consistent increase in both training and validation accuracy, reaching 95.72% and 93.87%, respectively, by the 19th epoch. The corresponding training and validation losses exhibited a steady decline, indicating improved predictive capabilities of the model. Early stopping was implemented, concluding training after 19 epochs due to a lack of significant improvement in the validation loss over the preceding ten (10) epochs. Subsequent evaluation of the test set revealed 95.02% accuracy and 0.2242 loss, as graphically depicted in Figures 5 and 6, affirming the model's robust performance on previously unseen data. This outcome indicates that the training data's underpinning patterns have been effectively learned by the model and can generalize well to new instances.

Moreover, as shown in Table 2, the model's performance on individual classes is assessed through precision, recall, and F1-score metrics, delivering a detailed understanding of its classification capabilities. Notably, the precision metric measures the accuracy of positive predictions, the recall metric gauges the model's ability to capture all relevant instances, and the F1 score provides a balance between precision and recall.

Examining the results for each class reveals variations in the model's performance across different categories. For instance, the model exhibits high precision, recall, and F1-score for classes like 'no_damage_buildings_street,' 'no_damage_water_related,' and 'no_damage_wildlife_ forest,' indicating robust performance in accurately identifying these instances. However, some classes, such as 'earthquake,' show lower scores, suggesting challenges in correctly predicting instances of this class.

Overall, the model achieves an impressive weighted average F1 score of 0.95 on the test set, emphasizing its proficiency in classifying various emergency scenarios. This aligns with the high accuracy of 95.02% reported earlier, affirming the model's efficacy in handling diverse and complex situations.

In addition, the provided confusion matrix, as shown in Figure 7, illustrates the model's performance in classifying different emergency scenarios across the specified categories. Each cell in the matrix represents the count of instances, with rows indicating the actual classes and columns indicating the predicted classes.

Analyzing the confusion matrix offers insightful information on the model's strengths and potential areas for improvement. For instance, the diagonal elements show correctly classified examples, going from top left to bottom right.

Table 2. Model performance

Label/Category	Precision	Recall	F1-Score	Support
accident_human_inflicted	0.85	0.68	0.76	25
earthquake	0.00	0.00	0.00	4
el_niño	0.95	0.86	0.90	21
flood	0.85	0.89	0.87	104
infrastructure_damage	0.91	0.94	0.92	143
landslide	0.83	0.65	0.73	46
no_damage_buildings_street	1.00	1.00	1.00	458
no_damage_human	0.85	0.92	0.88	12
no_damage_water_related	0.96	0.99	0.97	229
no_damage_wildlife_forest	1.00	0.99	0.99	228
urban_fire	0.82	0.86	0.84	43
wild_fire	0.96	0.91	0.93	53
accuracy	-	-	0.95	1366
macro avg	0.83	0.81	0.82	1366
weighted avg	0.95	0.95	0.95	1366

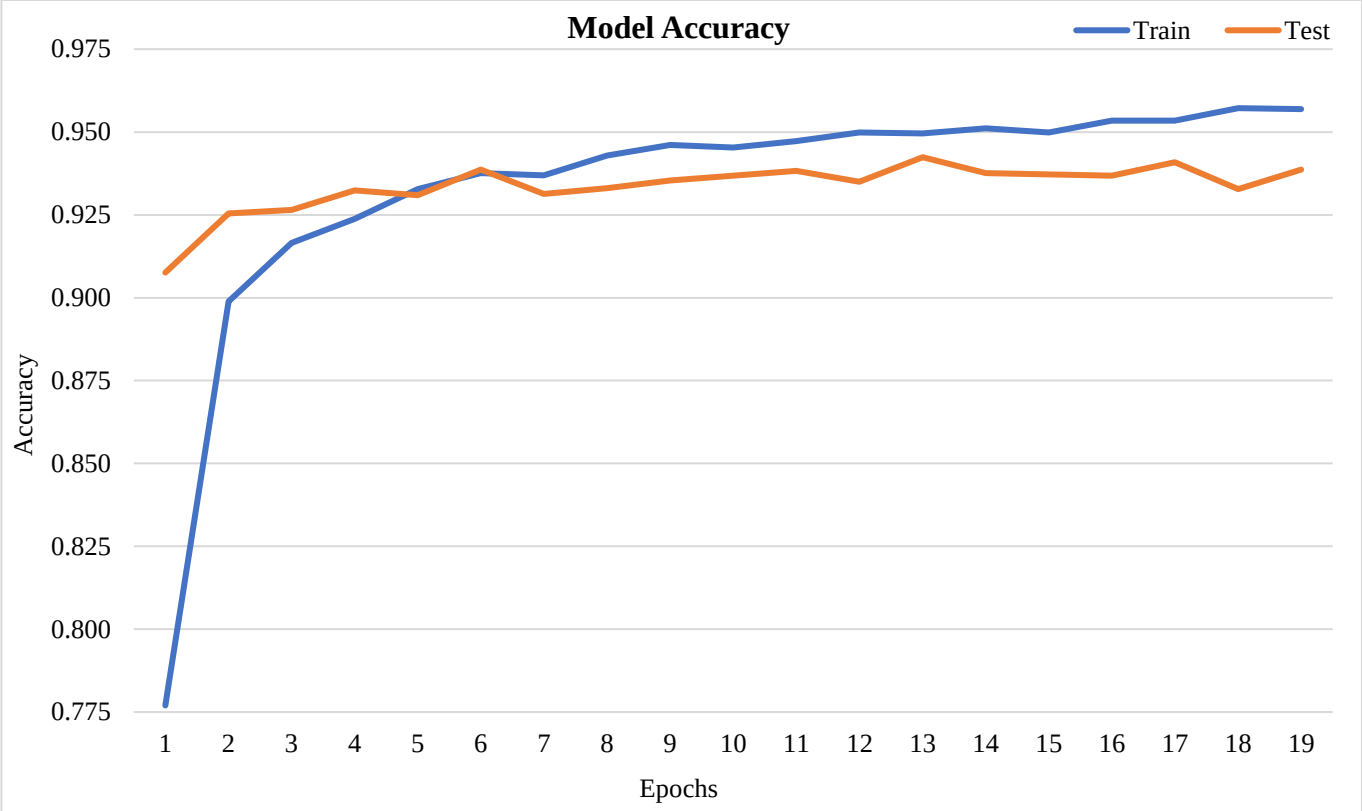

Fig. 5 Model graph (Accuracy)

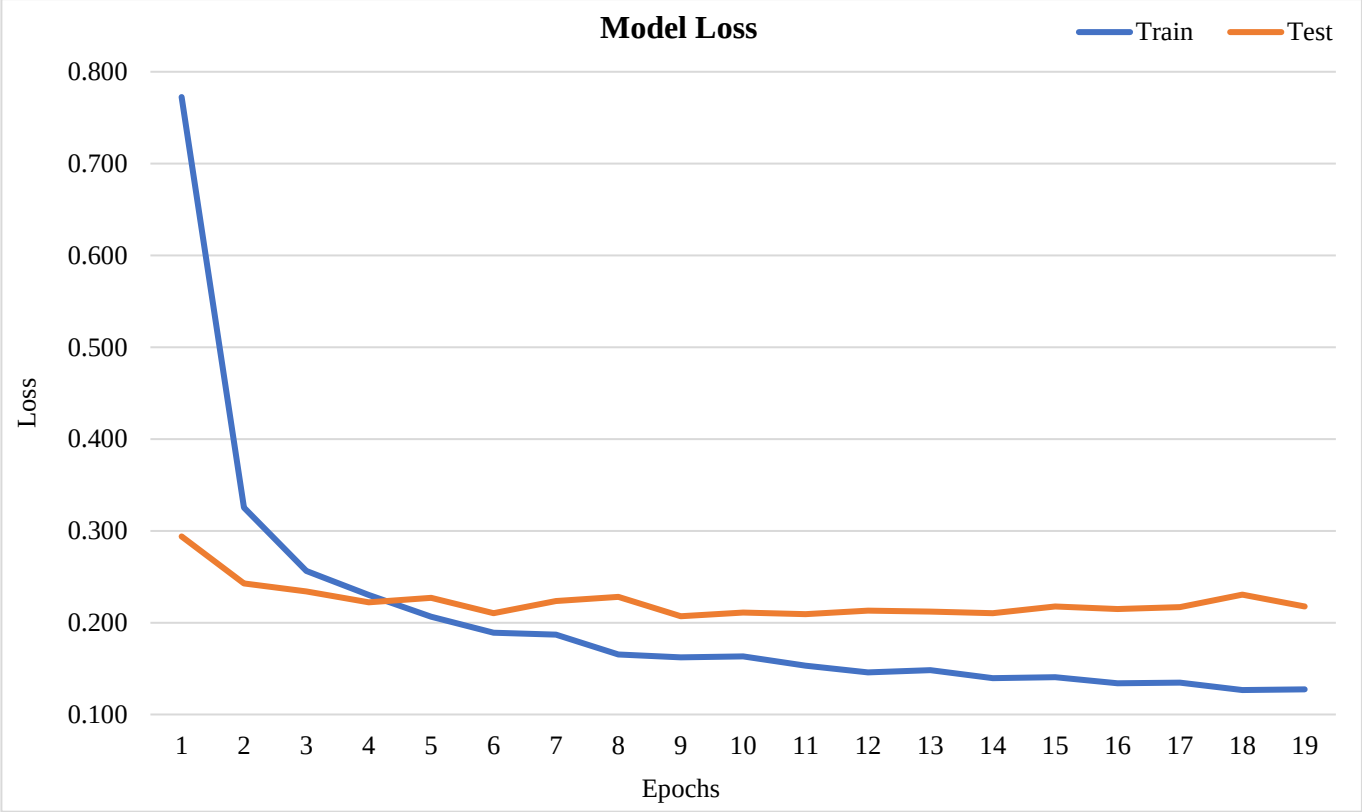

Fig. 6 Model graph (Loss)

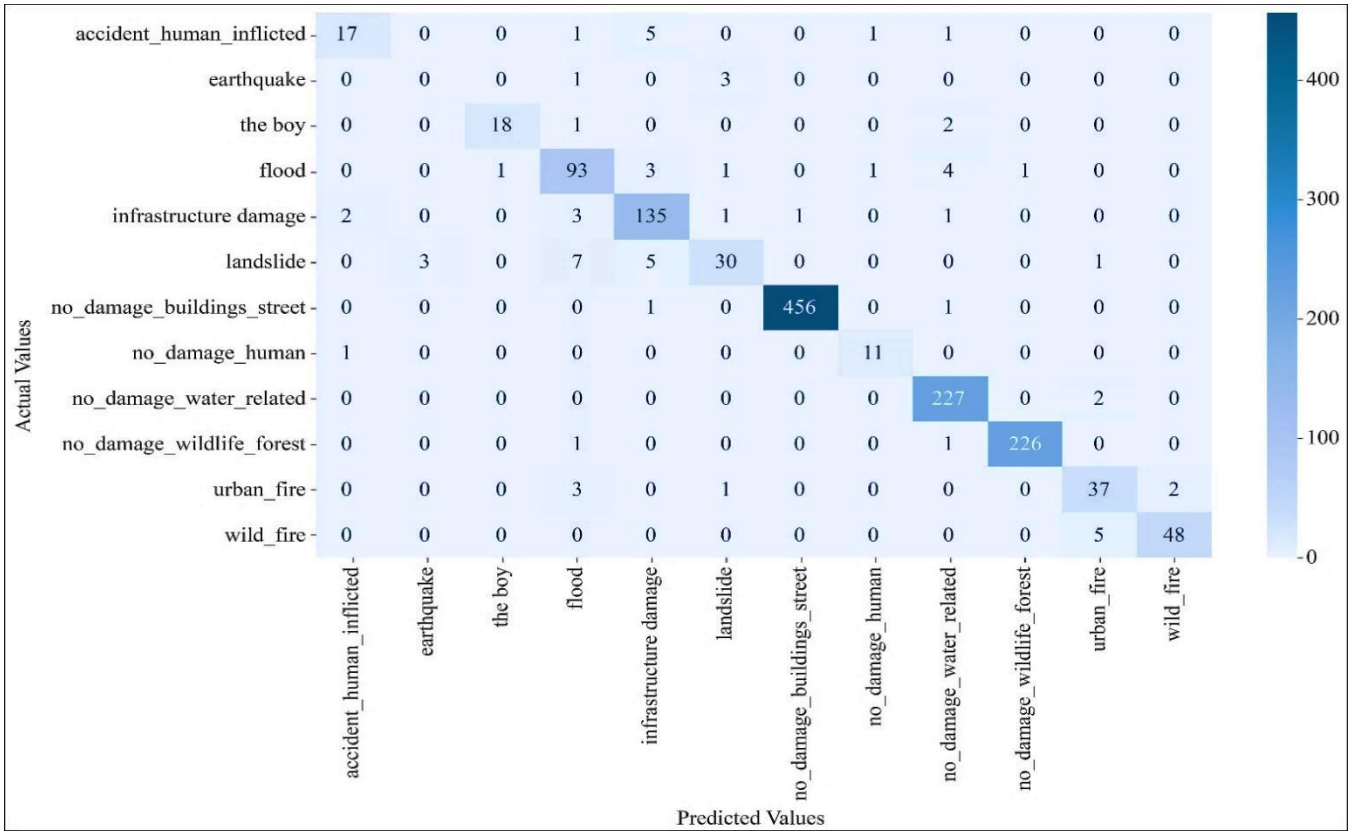

Fig. 7 Confusion matrix

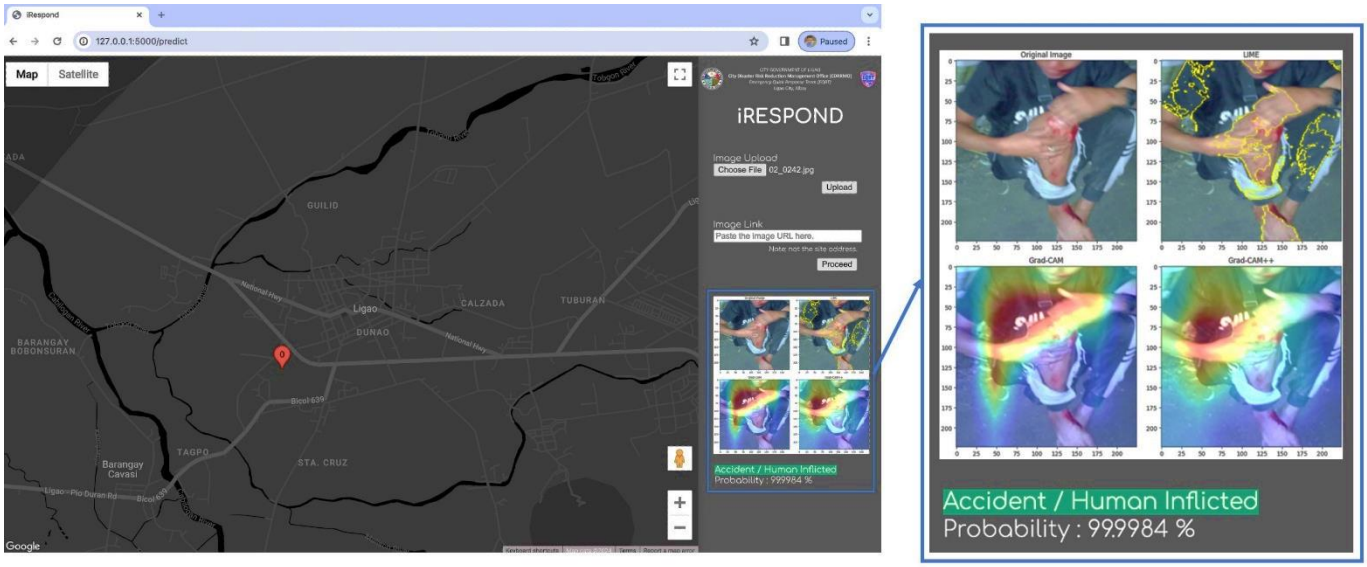

Fig. 8 Visualization report (Prototype)

Notably, classes like 'no_damage_buildings_street' and 'no_damage_water_related' exhibit high counts along the diagonal, indicating strong predictive accuracy. However, off-diagonal elements reveal instances of misclassification. The model appears to face challenges in accurately predicting instances of 'landslide' and 'urban_fire,' as evident from non-negligible counts in other columns for these classes.

4.3. Prototype Implementation

Upon the developed CNN model being integrated into the prototype of the iRESPOND System, the visualization report includes multiple techniques to interpret and explain the predictions. As shown in Figure 8, four visualization methods are employed: Original Image, which displays the original image, providing a visual representation of the emergency

scenario input into the model; LIME, which highlights specific areas in the image that influence most to the decision; Grad-CAM produces a heatmap, emphasizing the areas contributing significantly to the classification of the sample uploaded image; and Grad-CAM++ an enhanced version of Grad-CAM, providing more accurate localization of important areas in the image that refines the heatmap generation process, offering a clearer indication of the areas influencing the model's decision for the specified class. Also, a sample model's output for the 'Accident/Human Inflicted' category is presented alongside these visualizations, providing a comprehensive overview of the features and areas in the original image contributing to the model's classification.

5. Conclusion and Future Work

The iRESPOND System's trained CNN model demonstrates commendable performance in rapidly categorizing emergency incidents from captured images. The model exhibits a high accuracy of 95.02% on the test set, showcasing its effectiveness in providing early warnings and aiding disaster response efforts. The detailed evaluation, including precision, recall, and F1-score metrics for individual classes, offers a nuanced understanding of the model's strengths and areas for improvement. Notably, the model excels in classes like 'no_damage_buildings_street' and 'no_damage_water_related,' indicating robust predictive capabilities in certain emergency scenarios. Incorporating visualization techniques, such as LIME, Grad-CAM, and Grad-CAM++, enhances the model's interpretability. These methods contribute to the transparency of the CNN model, allowing us to gain insights into the features and areas in the input images that influence the predictions. The visualization report, including the original image and interpretability overlays, strengthens and facilitates a better understanding of the model's decision-making process.In general, the CNN model trained for the iRESPOND system produced results consistent with the findings of related research about CNN-based models' efficacy in disaster image classification.

The high accuracy of 95.02% on the test set of the iRESPOND model indicates progress towards accomplishing the goal of precise categorization and robust automated systems for disaster response, as emphasized by Rathod et al. and Shah et al. Moreover, the challenges found in previous research on identifying particular kinds of structural damage, including cracks and the obstacles associated with distinguishing fire and smoke in photos, correspond with the subtle differences in the iRESPOND model's performance in other classes. Results from the studies of Sharma et al., Zainorzoli et al., and Sushma and Lakshmi further confirm the iRESPOND model's architecture decision, as does ResNet50's persistent outperformance. Furthermore, as stressed in related literature, the iRESPOND model's integration of visualization tools parallels the demand for interpretability and transparency in AI-driven disaster response systems. Overall, the trained model of the iRESPOND System is a step forward in addressing the issues and using the potential noted in the larger body of work, advancing the development of reliable and effective disaster response systems.

While the CNN model of iRESPOND demonstrates promising results, continuous refinement and exploration of future enhancements can be done for future work. Addressing potential class imbalances, especially for classes with lower counts, could further improve the model's performance. Techniques such as oversampling, undersampling, or synthetic data generation should be explored. Fine-tuning hyperparameters, such as learning rate or dropout rates, could optimize the model's training process. Systematic experimentation with hyperparameter tuning may lead to improved accuracy and generalization. Implementing dynamic learning rate adjustment strategies, such as learning rate schedulers, could enhance training efficiency and convergence, potentially reducing training time. Expanding the dataset with diverse images and scenarios, including data from external sources, can contribute to the model's adaptability to a broader range of emergency incidents. External validation on different datasets could confirm the model's generalizability.

Funding Statement

The authors declared that no grants or funding were involved in supporting this work.

References

[1] Mary Ann E. Ignaco, "Mobile Application for Incident Reporting," *International Journal on Informatics Visualization*, vol. 5, no. 4, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Barry Sheehan et al., "On the Benefits of Insurance and Disaster Risk Management Integration for Improved Climate-Related Natural Catastrophe Resilience," *Environment Systems and Decisions*, vol. 43, pp. 639-648, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Global Assessment Report on Disaster Risk Reduction, GAR Special Report on Drought 2021, United Nations Office for Disaster Risk Reduction. [Online]. Available: https://www.undrr.org/publication/gar-special-report-drought-2021

[4] Trends in Disasters - Disasters in 2020 and 2021, World Disaster Report, pp. 210-243, 2022. [Online]. Available: https://www.ifrc.org/sites/default/files/2023-01/20230130_2022_WDR_DataAnnex.pdf

[5] Center for Fire Statistics of CTIF, World Fire Statistics, pp. 1-144, 2023. [Online]. Available: https://www.ctif.org/sites/default/files/2023-06/CTIF_Report28-ESG.pdf

[6] Yifan Xu et al., "Global, Regional, and National Burden of Road Injuries from 1990 to 2019," *International Journal of Environment Research and Public Health*, vol. 19, no. 24, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Ralf Bill et al., "Geospatial Information Research: State of the Art, Case Studies and Future Perspectives," *PFG-Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 90, pp. 349-389, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Robertas Damaševičius, Nebojsa Bacanin, and Sanjay Misra, "From Sensors to Safety: Internet of Emergency Services (IoES) for Emergency Response and Disaster Management," *Journal of Sensor and Actuator Networks*, vol. 12, no. 3, pp. 1-45, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[10] Saman Ghaffarian, Firouzeh Rosa Taghikhah, and Holger R. Maier, "Explainable Artificial Intelligence in Disaster Risk Management: Achievements and Prospective Futures," *International Journal of Disaster Risk Reduction*, vol. 98, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Saman Ghaffarian et al., "Post-Disaster Building Database Updating Using Automated Deep Learning: An Integration of Pre-Disaster OpenStreetMap and Multi-Temporal Satellite Data," *Remote Sensing*, vol. 11, no. 20, pp. 1-20, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yongqing Tang et al., "ForestResNet: A Deep Learning Algorithm for Forest Image Classification," *Journal of Physics*, vol. 2024, no. 1, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Archit Rathod et al., "Leveraging CNNs and Ensemble Learning for Automated Disaster Image Classification," *arXiv*, vol. 1, pp. 1-13, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jash Shah et al., "Proposed Methodology for Disaster Classification Using Computer Vision and Federated Learning," *Research Square*, pp. 1-9, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Laith Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of Big Data*, vol. 8, no. 53, pp. 1-74, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Amna Asif et al., "Automatic Analysis of Social Media Images to Identify Disaster Type and Infer Appropriate Emergency Response," *Journal of Big Data*, vol. 8, no. 83, pp. 1-28, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shannon Daly, and James A. Thom, "Mining and Classifying Image Posts on Social Media to Analyse Fires," *ISCRAM 2016 Conference Proceedings - 13th International Conference on Information Systems for Crisis Response and Management*, pp. 1-14, 2016. [Google Scholar] [Publisher Link]

[18] Ayan Mukhopadhyay et al., "A Review of Incident Prediction, Resource Allocation, and Dispatch Models for Emergency Management," *Accident Analysis & Prevention*, vol. 165, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] J. Kallas, and R. Napolitano, "Automated Large-Scale Damage Detection on Historic Buildings in Post-Disaster Areas Using Image Segmentation," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, pp. 797-804, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Neha Sharma, Vibhor Jain, and Anju Mishra, "An Analysis of Convolutional Neural Networks for Image Classification," *Procedia Computer Science*, vol. 132, pp. 377-384, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[21] Siti Maisarah Zainorzuli et al., "Comparison Study on Convolution Neural Network (CNN) Techniques for Image Classification," *Journal of Electrical and Electronic Systems Research*, vol. 20, pp. 11-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] L. Sushma, and K.P. Lakshmi, "An Analysis of Convolution Neural Network for Image Classification using Different Models," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 10, pp. 629-637, 2020. [Google Scholar] [Publisher Link]

[23] Sendai Framework for Disaster Risk Reduction 2015-2030, United Nation Office for Disaster Risk Reduction, 2019. [Online]. Available: https://www.undrr.org/publication/sendai-framework-disaster-risk-reduction-2015-2030

[24] Martín Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," *OSDI'16: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pp. 265-283, 2016. [Google Scholar] [Publisher Link]

[25] Simple, Flexible, Powerful, Keras, 2023. [Online]. Available: https://keras.io.

[26] Kaiming He et al., "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[27] Diederik P. Kingma, and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representation*, San Diego, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 1135-1144, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[29] Ramprasaath R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 618-626, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[30] Aditya Chattopadhay et al., "Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks," *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, USA, pp. 839-847, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[31] Jia Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database," 2009 *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, pp. 248-255, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[32] James Bergstra, and Yoshua Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281-305, 2012. [Google Scholar] [Publisher Link]

[33] Holger R. Maier et al., "On How Data are Partitioned in Model Development and Evaluation: Confronting the Elephant in the Room to Enhance Model Generalization," *Environmental Modelling & Software*, vol. 167, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[34] John D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[35] David M.W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *International Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011. [CrossRef] [Google Scholar] [Publisher Link]