Building Damage Detection based on Post-Hurricane Satellite Imagery using Transfer Learning and Convolutional Neural Networks

Khushboo Lavania[1], Aman Khanna[2], Jaswinder Singh[3]

Abstract
The rapid occurrences of the natural disasters across the world over the years has led to destruction of properties worth billions of dollars and also cost a lot of human lives. A Hurricane is one of these dangerous disasters that has caused immense damage to both our ecosystem as well as economic system. The timely gaining of the awareness of the situation after these natural disasters is crucial to disaster response management organisations and first responders. An efficient way to achieve that is the assessment of the damaged buildings using the satellite imagery. In this study, we propose two different custom architectures based on Convolutional Neural Networks (CNN) for the damage annotation in the post hurricane satellite imagery. We also use a pre-trained VGG16 architecture and compare the performances of transfer learning based and the custom convolutional networks. The satellite data for the study is taken from the survey of the 2017 Hurricane that occurred in the Greater Houston area. The pre-trained VGG16 model was able to achieve an overall accuracy of 97.33% on the test set and 96.80% on the validation set. The first custom CNN achieved a validation accuracy of 97.60% while the second more dense custom CNN achieved a validation accuracy of 96.69%. The pre-trained VGG16 model was able to outperform both CNN based custom architectures.

[1] *x19209835@student.ncirl.ie, MSc in Data Analytics, National College of Ireland*
[2] *x19231938@student.ncirl.ie, MSc in Data Analytics, National College of Ireland*
[3] *x19219997@student.ncirl.ie, MSc in Data Analytics, National College of Ireland*

1. INTRODUCTION

There has been a recent surge in the occurrences of various natural disasters like hurricanes, around the world due to the climate change. A hurricane is a type of a storm, also known as a *tropical cyclone*, which usually forms over subtropical and tropical regions in water bodies. It is one of the most disastrous natural events causing immense damage to both our ecosystem as well as our economic system. Every year, hurricanes and the heavy rainfall and high speed winds accompanying them inflict massive damage to property and costs countless human lives. Natural events like earthquake and hurricane cannot be controlled or contained by humanitarian efforts, but we still can mitigate the damage caused by these disasters. One of the key ways to accomplish that is the detection of the damages after the occurrence of a natural disasters like hurricane. This will not only help the different organisations involved in the disaster responses but also the first responders. This will in result lead to reducing the number of casualties. A automated damage detection system will greatly aid in reducing the time taken for producing the damage assessment reports.

Recent advances in the fields of machine learning and artificial intelligence, particularly CNNs (Convolutional Neural Networks) [1] and DNNs (Deep Neural Networks) [2], have achieved state-of-the-art results in a variety of computer vision tasks image recognition [3], image segmentation [4] and object detection [5]. Using satellite imagery, it is possible to build a classification model with a deep learning architecture that can accurately label the damaged/undamaged or flooded/non-flooded structures. The major advantage in using satellite imagery is that they have wider field of view than ground searches. Hence, they can cover more area for inspection in a single image. Manual analysis of satellite images is a time consuming process and could lead to false and missed detections due to the influence of subjective human factors. Therefore, an automated system is necessary in order to correctly analyse and classify these images.

Over the years, researchers have used various machine learning approaches for the the detection of the damaged structures in satellite images. Satellite imagery provides a lot of information to the first responders, emergency crews to help in disaster management. There are different ways to assess the damage based on the types of satellite images used. One of the ways is to use the synthetic aperture radar or SAR images (refer to the Darthmouth Flood Observatory [6]). It is useful in mapping various distinct surface features or textures. The SAR processed images are readable by both humans and algorithms. Another more commonly used technique is to use the optical sensor imagery. The optical cameras depend on the light created by other sources like the Sun, while a radar aperture uses its own radio waves to measure or see the different objects. The synthetic aperture refers to a physically moving radar instrument over an area while it collects the target information. The only problem with using SAR images for detecting damaged structures is that the SAR images have too coarse resolution to allow for the damage assessment on the building level. Additionally, the satellites having SAR sensors are too low in numbers as compared to those having the optical sensors. Therefore, collecting the SAR images is a time consuming and challenging task.

For the purpose of this project, we will be using the optical sensory imagery to analyze the images of a post hurricane scenario. The reason for choosing the optical imagery has been highlighted in the previous paragraph. Using various computer vision techniques, we will design a custom deep learning based architecture that can automatically label a post hurricane satellite imagery of a structure as '*Damaged* or *Undamaged*'. The results can help different disaster response management organisations to identify the affected areas by the Hurricane and take appropriate measures to better plan and distribute the necessary resources to the survivors. The satellite imagery data used in this study was taken before and after the occurrence of Hurricane Harvey in the Greater Houston in the year 2017. The images were labelled damaged/flooded by a team of volunteers through a crowd sourcing project called Tomnod [7]. Figure shows the map of the Greater Houston area which was affected by the Hurricane Harvey in the year 2017. The green circles are the coordinates of the damaged/flooded structures.

We compare the performances of different custom designed convolutional neural networks (CNNs) and a pre-trained deep learning architecture for the automatic annotation of the damaged/flooded structures in the data. The remaining of the report is organised as follows:

- *Section 2*: Critical review of the existing literature in the domain of damage classification using satellite imagery and use of satellite imagery in different tasks.
- *Section 3*: Describes the proposed methodology for the damage annotation on satellite imagery.
- *Section 4*: Discusses the implementation of the proposed deep learning models and the evaluation of the different results obtained.
- *Section 5*: Discusses the results obtained from different models and summarizes the overall framework and methodology of the study. It also highlights some of the ideas that could not be incorporated into the study as part of future works.

2. RELATED WORKS

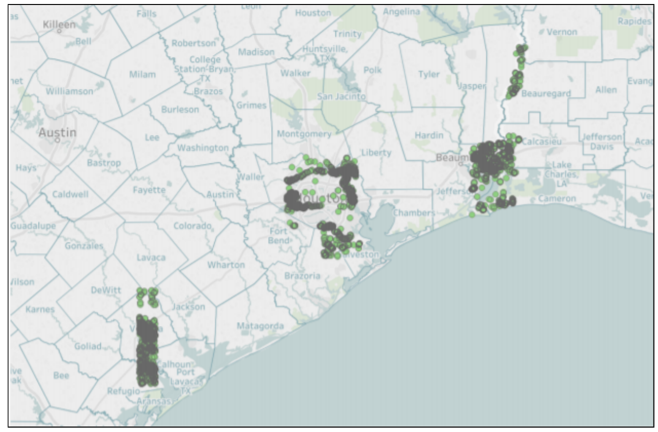

Figure 1. Map of the Greater Houston area which was affected the Hurricane Harvey in 2017

2.1 Introduction

Before conducting any research, it is crucial to analyse and critically review the current work in the chosen domain. This will aid the researchers in framing new ideas and assessing the novelty of the research conducted. In this section, we will be reviewing some of the key literature in the domain of image classification and annotation using satellite imagery.

With the rapid development in the technology used in space, both in terms of the satellite deployment and capabilities of processing of large data, we can now accomplish the things which were once thought as impossible by our ancestors. The development of new technologies like remote sensing, SAR, etc. have allowed us to monitor the planet down to the smallest structure through an enormous network of satellites orbiting the Earth. The satellite imagery has been used for a range of different tasks like classification of land cover, classification of crops, assessment of damaged structures after a natural disaster, aerial segmentation of structures, etc. In this regard, Ulmas and Liiv [8] have used the satellite imagery for creating a convolutional based U-net architecture for the classification and segmentation of land covers. They analyzed their results using two different satellite datasets- BigEarthNet, containing the satellite images from Sentinel-2 satellite of 10 different European countries, an originally composed data containing images from Sentinel-2 and a land cover map of Estonia. They were able to achieve an F-1 score of 0.749 for a multi-class classification of land covers. Their proposed model also highlighted the noisy images in the satellite data containing incorrect labels. In addition, their segmentation model was able to generate automatic land covering mappings based on the Sentinel-2 satellite imagery for different categories like forests, inland waters and arable land. Crop classification is another pioneering application of satellite imagery in the recent years. Viskovic et al. [9] utilized the multi-temporal property of the satellite imagery and build a random forest classifier to classify the different classes of crops on 15 different indices of the same area. They aggregated the different satellite images taken throughout the year of a single area to implement their algorithm. They were able to achieve an accuracy of 84.20% and a Kappa score of 0.8157. In the next subsection, we will highlight some of the key literature in the domain of our research topic i.e damage annotation using satellite images.

2.2 Damage Assessment using Satellite Imagery

The use of satellite imagery to monitor the disaster affected areas is becoming increasingly popular. This is because of the automation of entire process has eliminated the factor of human error. This will aid in properly prioritizing the disaster response, rescue operations and coordinate relief efforts after the occurrence of a disaster.

In 2018, Doshi et al. [10] proposed a novel metric called the Disaster Imapact Index (DII) and used it to analyze the impact of two different disasters- the Harvey hurricane flood and the Santa Rosa fire. They proposed a framework for detection of severely affected areas using a technique called change detection based on Convolutional Neural Networks (CNN). A pre-trained semantic segmentation model has been used to extract man-made features such as roads, buildings, etc by analyzing before and after disaster images. To analyze the impact, the difference between the two segmentation masks is computed. They were able to achieve an F-1 score of 81.2% on the hurricane dataset and an F-1 score of 83.5% on the Santa Rosa fire image dataset.

Remote sensing images have shown promising results in the recent years for effective extraction of the information from buildings about the disaster. In this regard, Ma et al. [11] used the high resolution remote sensing images to build a CNN based classifier to evaluate the degree of damage in the groups of buildings in a post earthquake scenario. They proposed a CNN Inception V3 architecture combining block vector data and the remote sensing imagery to select the best features for detection of degree of damage in a group of structures. They added two layers, Combination and Separate to improve upon the Inception V3 model for easier processing of high resolution remote sensing images. They tested their proposed model on the remote sensing images of Yushu earthquake. they were able to achieve a test accuracy of over 90%, outperforming the traditional machine learning classifiers using artificial feature extraction by over 18% in terms of test accuracy.

One of the most important works on the dataset used in this study was done by Cao and Chloe [12], in which they combined the geolocation features of the affected areas and the satellite imagery for the damage detection in the post Hurricane scenario. Their method showed a significant improvement from performing the same task while using only the satellite images. They found out that using the geolocation features based convolutional network improves the likelihood of detecting the damaged structure. They also analyzed the effect of assigning the same embedding size to both the satellite imagery and the geolocation features. They were able to achieve an accuracy of over 97% with using both imagery and features. A summary of their results is summarized in figure 2.

Method	Metrics			
	ACC	Precision	Recall	F1 score
Img only	$79.5 \pm 8.3\%$	0.88 ± 0.03	0.64 ± 0.30	0.68 ± 0.22
Geo only	$88.6 \pm 1.4\%$	0.86 ± 0.03	0.97 ± 0.003	0.91 ± 0.02
Img + Geo	$\mathbf{97.47 \pm 2.5\%}$	$\mathbf{0.91 \pm 0.14}$	$\mathbf{0.99 \pm 0.003}$	$\mathbf{0.94 \pm 0.08}$

Figure 2. Summary of the results obtained by Cao and Choe [12]

Xu et al. [13] used the satellite imagery from three different disasters to build a convolutional neural network for the detection of damaged buildings. They constructed their own custom dataset by combining satellite images from three different earthquakes in three different areas: Haiti (2010), Mexico (2017) and Indonesia (2018). They used a 5 level scale to assess the damage to a building: *'No Damage', 'Possible Damage', 'Moderate Damage', 'Severe Damage', 'Destroyed'*. They later combined the *'Severe Damage'* and *'Destroyed'* class into a single *'Damaged'* class because the assessments were noisy and the labels were inconsistent across some datasets. They constructed 4 different AlexNet based CNN architectures each tailored to handle different challenges in the multi-class classification task. They evaluated the performance of each of the architectures on the Haiti dataset using a 5-fold cross validation method.

Another important earthquake damage detection study was done by Naito et al. [14] in which they they used the aerial imagery of the 2016 Kumamoto earthquake to classify the damaged structures into one of the four levels of damages: Level-1:*no damage*; Level-2 *minor damage*; Level-3 *moderate damage*; Level-4 *major damage*
They designed different architectures based on SVM and CNN and used different clustering techniques to extract the features from the images and compared the results on the basis of different metrics like recall, precision and accuracy. They also compared the results of visual interpretation with their proposed CNN architecture.

Li et al. [15] conducted a study on a small Hurricane Sandy dataset using a pre-trained VGG16 model. It is evident to see that the classification accuracy of the model increases by using a pre-trained model. To improve the flexibility of the model owing to complex scenario the Gaussian Blur and Gaussian noise has been added. To further improve the performance the data has been enhanced by using an additional Hurricane Irma dataset. As the CNN requires a large amount of data for training, therefore using more number of images to train the neural model has a direct impact on the classification accuracy. As per the author, the generalization ability of the model could additionally be improved by using a larger dataset for future work. Deep Learning technique i.e. pre-trained EfficientNet as a baseline encoder has been used for the study by Shao et al. [16] on the areas affected by five different types of natural disasters. A symmetrical U-shaped structure of the neural model has been designed which utilizes before and after satellite pictures of the areas impacted. It is beneficial to efficiently grab the information, especially where the buildings have been demolished to the ground. The developed model is successful in achieving an F1 score of 82.9%. But the author emphasizes that a more robust model can be designed which identifies damaged buildings based upon remote sensing pixel classification to effectively apply the model for real-world scenarios.

The multi-resolution fact of CNN has been considered for the research conducted by Duarte et al. [17] to identify buildings crashed by disaster by using satellite images of different resolution. The model is successful in extracting more significant information by the addition of feature maps. By using feature maps and images of various resolution the classification accuracy of the model improved by 4%. But as per the author the model is not able to recognize minor signs of damages which is essential to make the classification decision.

In a study conducted by Albert et al. [18], authors are trying to identify different patterns in urban neighborhoods using large satellite imagery data. The models are trained on this data and further comparing different deep convolutional networks architecture which includes VGG16 and ResNet. They developed two step task which includes: first foreseeing urban land use classes from satellite image, after that turning this classification into a ceaseless range by inserting the features extracted from convolutional classifier into lower dimensional manifold. The performed classification task gave encouraging results and as for future works, authors tend to perform more different ways to compare the urban neighbors using other several techniques.

Cade et al. [19] have proposed a method to analyze and detect building that damaged after an occurrence of natural disasters. They have worked in a two-step fashion where first step was to take satellite images as input data and detect whether a building is damaged that need critical attention. The ResNet50 architecture was utilized and obtained high accuracy on natural disaster damage classification and further to detect the building damage detection, Mask R-CNN was used which gave overall suboptimal performance due to computational challenges and time restrictions which authors faced. Finally, in future they proposed to use a larger training dataset and to train the models for a larger period.

2.3 Conclusion

After critically reviewing the key literature in the domain, we can conclude that different pre trained deep learning architectures like VGG, Inception V3, Resnet have performed exceptionally well in the satellite imagery classification tasks. They were even able to outperform the state-of-the-art models in some cases. Therefore, for the purpose of our study, we will also be using a pre-trained VGG16 architecture for the damage annotation in the optical satellite imagery. We will also design two different custom architectures using data augmentation and Leaky ReLU activation function and compare them based on different metrics like test and validation accuracy, ROC curves, precision, etc. The next section will discuss the methodology implemented for the study.

3. METHODOLOGY

A well planned methodology is a crucial part of implementing the proposed technique for the study. In this research, we will be employing a Knowledge Discovery in Database (KDD) methodology for achieving the desired objective i.e classification of the damaged structures in a post Hurricane satellite imagery. The steps followed are described below and are summarized in the figure 3.

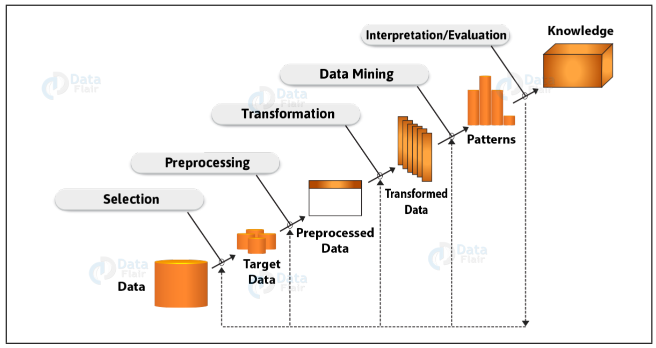

Figure 3. Steps in a KDD based methodlogy

3.1 Data Selection

The objective of this study is to identify buildings that have been impacted by a natural disaster. To achieve this, satellite images of the area affected by the 2017 Hurricane Harvey calamity have been chosen.

3.1.1 Dataset Source

The data for this study has been made publicly available for research purposes taken from the IEEE data portal [20].

3.1.2 Ethical Concern

There is no ethical concern involved in using the dataset. A public usage license is available for this dataset.

3.1.3 Dataset Description

The dataset has four folders which contains the staellite images of damaged and undamaged buildings. The description of the data directories is as follows: (Also refer to figure 4)

- **Validation Another**: It includes 1000 images of each class.
- **Test Another**: It is an imbalanced folders which includes 9000 images.
- **Test**: Another balanced folder which includes 1000 images.
- **Train Another**: It includes 5000 images of each class.

```
train_another:
damage                    (5000 files)
no_damage                 (5000 files)
validation_another:
damage                    (1000 files)
no_damage                 (1000 files)
test_another:
damage                    (8000 files)
no_damage                 (1000 files)
test:
damage                    (1000 files)
no_damage                 (1000 files)
```

Figure 4. Description of the directories of the data

3.2 Data Pre-Processing and Transformation

It is another essential step to transform data into appropriate forms before applying any machine learning model. The real-world data accommodates noise and dealing with such kind of data increases processing time and also reduces classification accuracy. Therefore, data transformation is a vital step to deal with such a situation. The steps which are followed for data transformation are explained below and are represented by a flow chart in figure 8.

3.2.1 Importing Data into Colab

The dataset was downloaded on local machine in .zip format and further it was uploaded on google drive as a zipped file. The .zip file was then unzipped on the google drive to avoid re-uploading dataset every time we run the compiled code and also to cut short the time to read the data through Colab worksheet.

3.2.2 Extracting Labels from Images

The images of the datasets are not mapped with their respective label. Therefore, before proceeding further all the images are linked with their respective labels using TensorFlow py function. The extracted labels, lattitude and longitudes of the images were then stored in a pandas dataframe (refer to figure 5).

	path	Damaged	Sub_directory_name	location	Lattitude	Longitude
0	C:\Users\Jaswinder Singh\Downloads\SEM-2\DMML-...	damage	test	-93.548123_30.900623	-93.548123	30.900623
1	C:\Users\Jaswinder Singh\Downloads\SEM-2\DMML-...	damage	test	-93.560128_30.894917	-93.560128	30.894917
2	C:\Users\Jaswinder Singh\Downloads\SEM-2\DMML-...	damage	test	-93.578271_30.779923999999998	-93.578271	30.779924
3	C:\Users\Jaswinder Singh\Downloads\SEM-2\DMML-...	damage	test	-93.590598_30.694956	-93.590598	30.694956
4	C:\Users\Jaswinder Singh\Downloads\SEM-2\DMML-...	damage	test	-93.604017_30.793719	-93.604017	30.793719

Figure 5. Glimpse of the pandas dataframe containing image labels and coordinates (lattitudes and longitudes)

3.2.3 Extracting Features from Images

The size of the window is controlled on the basis of the physical distance between two points. Therefore, there could be round-off errors in the conversion of the distance to the number of pixels. To avoid this error, we project the number of pixels into the same feature dimension. These images are then fed to a convolutional neural network in order to extract further useful features like edges.

3.2.4 Handling Unlabelled images

Few of the images do not have their labels. Training the model with unlabelled images is not the correct procedure. Hence, such images are dropped. The dataset is quite large, so removing few unlabelled images will not hinder the performance of the model.

3.2.5 Image Resizing

To reduce the computational complexity and increase the classification accuracy all the images are resized to the same size of 128*128*3 using the TensorFlow resize images method. A plot of the resized images read using openCV library is shown in the figure 6

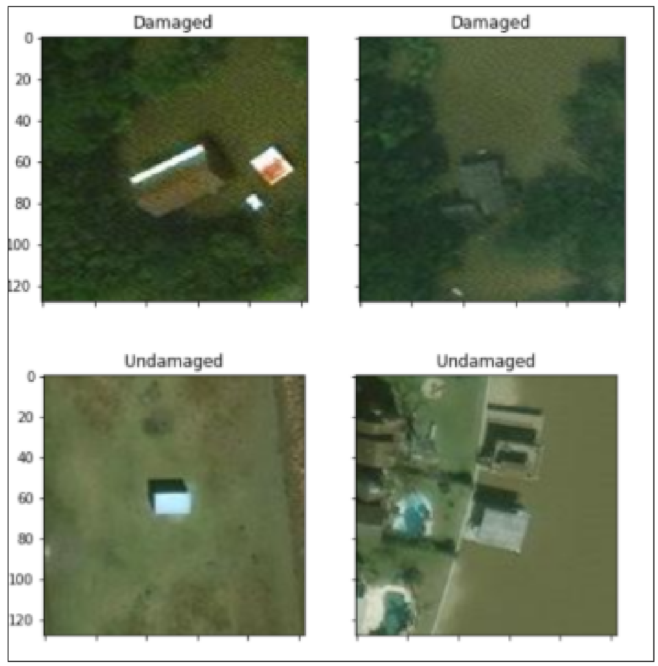

Figure 6. Plot of resized images (128*128*3)

3.2.6 Data Augmentation

CNN is considered to be a data greedy model. The more the data better is the performance of the model [2]. Sometimes, getting such large data for a real-world problem is not possible. An effective solution to deal with such an issue is data augmentation. It provides different ways to enhance the diversity of data. The below steps are followed to increase the data for training the model. The function used for performing data augmentation is shown in the figure 7

- **Image Flipping:** The images are randomly flipped with a 50% probability of each orientation either left-right or up-down. It is done by using TensorFlow random flip left right and random flip up down function. The flipped images are then added to the dataset for data extention.

- **Image Rotation:** The images are rotated with either 0, 90, 180, or 270 degrees.The probability of each rotation was 25%. It is also performed by using the TensorFlow rot90 function.
- **Data Translation:** The images are randomly translated either into x, y or in both the direction by using TensorFlow image translate function.

```
# Converting images to tensor --- Image Transformation
def image_transformation(img, label):
    # converting to Tensor
    img = tf.convert_to_tensor(img)
  # JPEG image is read as uint16. therefore, it has to be converted to uint8
    img = tf.dtypes.cast(img, tf.uint8)
 # Shape of the tensor
    img.set_shape((128, 128, 3))
 # Convert to float32
    img = tf.image.convert_image_dtype(img, tf.float32)
    img = tf.image.resize(img, [128, 128])
# convert the labels into a Tensor and set the shape
    label = tf.convert_to_tensor(label)
    label.set_shape(())
    return img, label

AUTOTUNE = tf.data.experimental.AUTOTUNE
```

Figure 7. Function used for performing data augmentation on the data

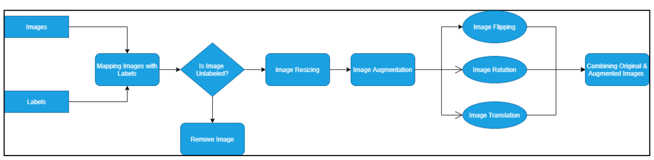

Figure 8. Steps followed for image transformation

4. Implementation and Evaluation

Three different deep learning architectures were implemented for accurate classification of the damaged structures in the post Hurricane satellite images. The first architecture i.e VGG16 employed a methodology called as transfer learning to classify the images. The main idea behind a transfer learning methodology is that a pre-trained model containing a fixed number of layers with specified weights is chosen. These layers combined all together forms the feature layer of a transfer learning architecture. The next layer is called the *classification layer*. The classification layer is added to convert the output to the desired number of classes, which in our case is two.

The second and third architectures are two different custom designed CNNs for performing the desired classification task. Each of the two architectures have different features like no. of layers, type of optimizers and activation functions, etc. The three architectures are explained below in detail:

4.1 VGG16 (Transfer Learning)

We implement a well known VGG16 architetcure [21] which is known to perform exceptionally well on the ImageNet dataset. It contains thirteen convolutional layers and has nearly fifteen million parameters. The different layers in the VGG16 architecture are shown in the figure 9. We used the ImageNet weights for our pre-trained VGG16 model. and added two layers to the pre-trained VGG16 for performing the desired image classification task. The full VGG16 architecture now contains 3 layers. (refer to figure 10)

- *Pre-trained Layer*: It converts each of the images into a $4 \times 4 \times 512$ block of features.
- *Averaging Layer*: It averages over the 4×4 spatial locations and convert the block of features into a single 512 element vector(per image).
- *Dense Layer*: It converts the block of features into a single prediction per image. We do not need an activation function here because this prediction is treated as a raw prediction value. The positive numbers predict class 1 and the negative numbers predict class 0. A suitable command was added to convert these into a binary prediction for each image.

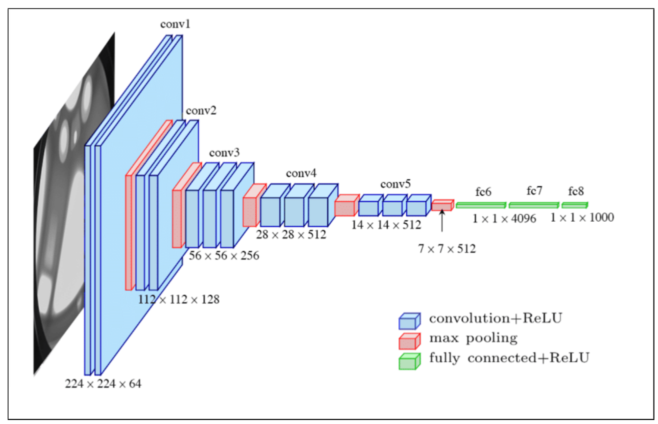

Figure 9. VGG 16 architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 513
=================================================================
Total params: 14,715,201
Trainable params: 7,079,937
Non-trainable params: 7,635,264
```

Figure 10. Summary of the VGG16 pre-trained model

We implemented a deep learning technique called fine-tuning. In this technique the fully connected layers of the existing pre-trained network are removed and are replaced with a new set of connected layers. The weights are then fine tuned to recognize the new classes of objects. The reason for implementation of this technique is because the training dataset size is large and is quite similar to the dataset that pre-trained model was trained on.

4.1.1 Evaluation

The fine tuned VGG16 model with 50 epochs was able to achieve an overall test accuracy of 97.33% and a validation accuracy of 96.80%. The learning curves and the confusion matrix are shown in figures 11 and 12 respectively.

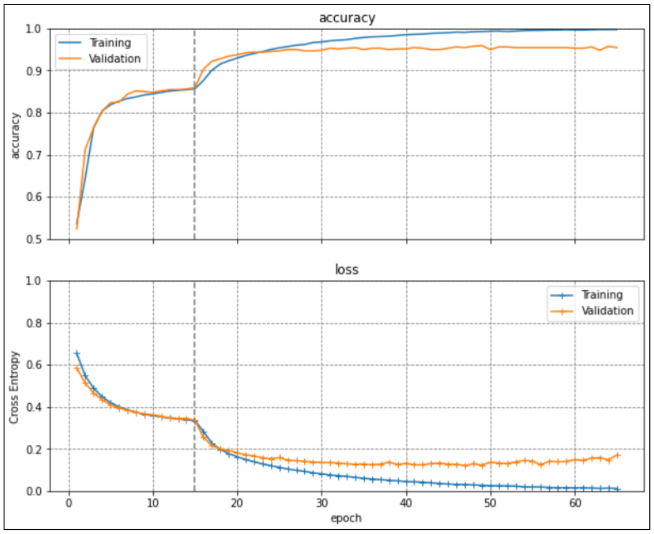

Figure 11. Learning curves for the pre-trained VGG16 model

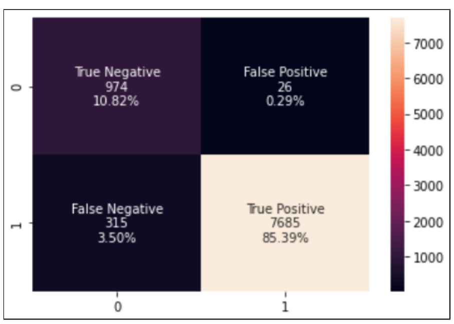

Figure 12. Confusion Matrix for the VGG16 pre-trained model

4.2 Custom CNN architecture - I

The first custom convolutional neural network uses the leaky ReLU (rectified linear activation unit) as the activation function. Leaky ReLU is an improved version of the ReLU function designed to address the problem of vanishing gradients for the inputs that are less than zero causing the deactivation of neurons in that region. This problem is sometimes called as the *dying ReLU* problem. Leaky ReLU rectifies this problem since it is defined as the extremely small linear component of the values of the inputs.

The custom architecture designed contains 16 layers and contains over 3.4 million parameters. The description of the layers is shown in figure 13

4.2.1 Evaluation

The binary crossentropy was chosen as the loss function for the custom model and optimizer chosen was RMSprop with

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 72, 72, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 34, 34, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 15, 15, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3211776
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
```

Figure 13. Summary for the custom CNN designed model architecture

a learning rate of 10^{-4}. The model was able to achieve a validation accuracy of 97.60% and an AUC value of 0.9948. the learning curves, classification report and the ROC curve are shown in the figures 14 , 15 and 16 respectively.

4.3 Custom CNN architecture - II

The second custom CNN architecture basically uses various techniques in order to prevent overfitting in the later epochs. It contains the following components:

- *Data Augmentation:* As described in the pre-processing part, the data augmentation is done to avoid the overfitting problem in the deep learning architectures. Augmentation essentially performs a series of transformations like flipping(horizontally and vertically), rotations, etc. to the training set. This will effectively increase the number of training images but will reduce the chances of overfitting and hence increase the validation and test accuracies.

- *Fully Dropout layer:* It is another technique used to prevent the overfitting in deep learning models. It works by randomly setting the outgoing edges of the neurons in the hidden layer to 0 at each update in the training process.

- *L2 regularization (Adam optimizer):* Regularization is a process in which an additional information is introduced in order to prevent the overfitting. The regression implementing L2 is also called the ridge regression. For a model implementing L2 regularization, regression function basically stays the same but the loss function now contains the regularization term. The L2 norm and

Figure 14. Learning Curves for the custom CNN architecture

	precision	recall	f1-score	support
damage	0.89	1.00	0.94	8000
no_damage	0.00	0.00	0.00	1000
accuracy			0.89	9000
macro avg	0.44	0.50	0.47	9000
weighted avg	0.79	0.89	0.84	9000

Figure 15. Classification report for the custom CNN architecture

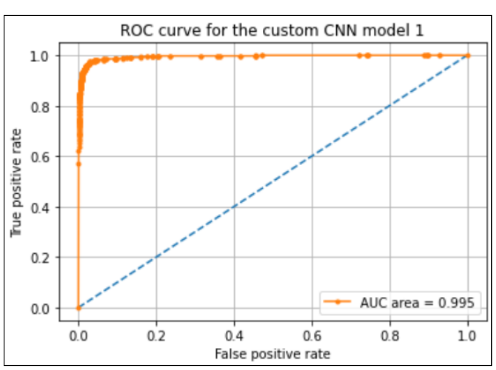

Figure 16. ROC curve for the custom architecture

loss function is given as:

$$\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}} \tag{1}$$

$$\text{Loss} = \text{Error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2 \tag{2}$$

- *Leaky ReLU function:* AS explained in the previous custom architecture, leaky ReLU is used to overcome the dying ReLU problem in the ReLU function.

The custom architecture contains 20 layers in total and about 3.4 million parameters. The model architecture summary is shown in figure 17

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 72, 72, 64)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 34, 34, 128)       0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
dropout_12 (Dropout)         (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)   (None, 15, 15, 128)       0
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_13 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_14 (Dropout)         (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               3211776
_________________________________________________________________
leaky_re_lu_14 (LeakyReLU)   (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
```

Figure 17. Summary for the second custom CNN architecture

4.3.1 Evaluation

The second custom designed architecture was able to achieve a test accuracy of 97.91% and a validation accuracy of 96.69%. The AUC value for the model came out to be 0.993 which can be considered ideal for all practical purposes. The ROC curve and the learning curves for the architecture are shown in the figures 18 and 19 respectively.

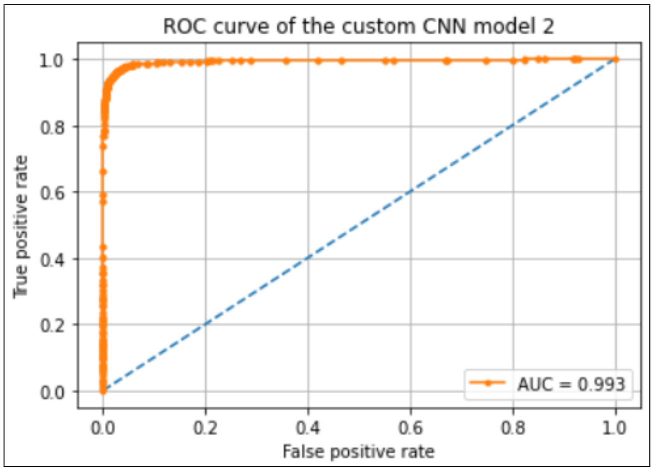

Figure 18. ROC Curve for the second custom CNN architecture

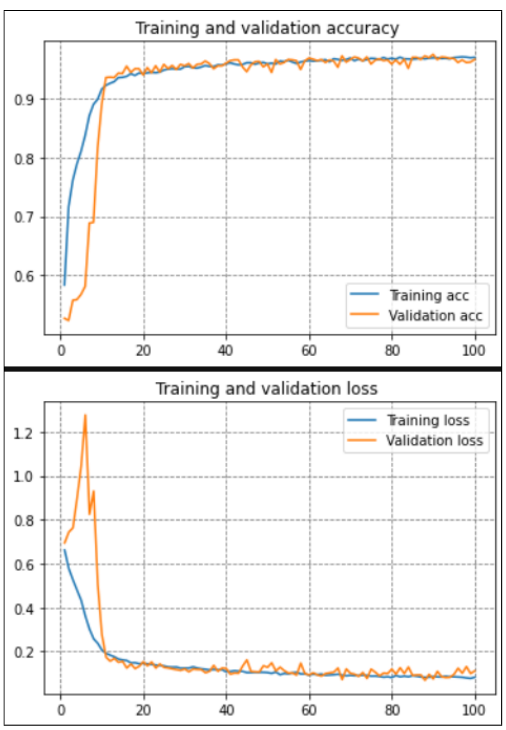

Figure 19. Learning curves for the second custom CNN architecture

5. Conclusion

An aggregate of three models were designed and compared in this project - the initial two were simply the basic, planned custom CNN models, and the remainder last model was developed using pre-trained transfer learning algorithms, namely VGG16. Comparing these models created some fascinating outcomes that are summed up below. All these models were trained and validated on the selected dataset and further the different classes were predicted on the test data. Each of the CNN model was studied and their performance were measured on different metrics namely Accuracy, Recall, Precision and F1 score. Accuracy and loss graphs were also contemplated. The confusion matrix was also drawn for each model to measure the effectiveness. The highest accuracy rate achieved was 97.91% by the second custom CNN architecture, followed by pre-trained VGG16 model and the first custom CNN. Both the custom models also performed quite well but were little prone to the issue of overfitting. As examined on test data, the second custom CNN model which included data augmentation, fully dropout layer and L2 regularization outperformed the other models to achieve an accuracy of 97.91% and F1 score of 0.97. In terms of training time, VGG16 took the least amount of time to train while the second custom CNN having 20 layers took the longest to train.

6. Future Works

Transfer learning and deep learning are evolving subjects, combined with the new technological updates in satellite imagery, much advancement is possible in future for this project:

- The post hurricane satellite imagery dataset contained very less images and further combined with data augmentation to increase image quantity. But more images could be added in input data to enhance training data for the models. More techniques can be explored like blurring techniques or adding more dataset to increase variation.
- Although the comparison of these tests was done in controlled conditions, other variables must be considered for future works. For instance, compilation for this project was done on Google Colab notebook, due to the scarcity of GPU enabled machines, which resulted in allotment of many varieties of GPUs for processing of code. Although, it wouldn't affect the results so much but this can be overlooked if the compilation of development of code can be done on physical hardware.
- As overfitting can be a problem for the model to perform at their highest efficiency, we can consider different approach or techniques to enhance the performance and overcome this issue.

References

[1] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[2] Abul Bashar. Survey on evolving deep learning neural network architectures. *Journal of Artificial Intelligence*, 1(02):73–82, 2019.

[3] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[4] Sanxing Zhang, Zhenhuan Ma, Gang Zhang, Tao Lei, Rui Zhang, and Yi Cui. Semantic image segmentation with deep convolutional neural networks and quick shift. *Symmetry*, 12(3):427, 2020.

[5] Yalong Pi, Nipun D Nath, and Amir H Behzadan. Convolutional neural networks for object detection in aerial imagery for disaster response and recovery. *Advanced Engineering Informatics*, 43:101009, 2020.

[6] Darthmouth flood observatory. [Accessed 5th May 2021].

[7] Maxar Technologies. In the blink of an eye – looking back on nine years with tomnod, 2019. [Accessed 5th May 2021].

[8] Priit Ulmas and Innar Liiv. Segmentation of satellite imagery using u-net models for land cover classification. *arXiv preprint arXiv:2003.02899*, 2020.

[9] Lucija Viskovic, Ivana Nizetic Kosovic, and Toni Mastelic. Crop classification using multi-spectral and multitemporal satellite imagery with machine learning. In *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5. IEEE, 2019.

[10] Jigar Doshi, Saikat Basu, and Guan Pang. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*, 2018.

[11] Haojie Ma, Yalan Liu, Yuhuan Ren, Dacheng Wang, Linjun Yu, and Jingxian Yu. Improved cnn classification method for groups of buildings damaged by earthquake, based on high resolution remote sensing images. *Remote Sensing*, 12(2):260, 2020.

[12] Quoc Dung Cao and Youngjun Choe. Post-hurricane damage assessment using satellite imagery and geolocation features. *arXiv preprint arXiv:2012.08624*, 2020.

[13] Joseph Z Xu, Wenhan Lu, Zebo Li, Pranav Khaitan, and Valeriya Zaytseva. Building damage detection in satellite imagery using convolutional neural networks. *arXiv preprint arXiv:1910.06444*, 2019.

[14] Shohei Naito, Hiromitsu Tomozawa, Yuji Mori, Takeshi Nagata, Naokazu Monma, Hiromitsu Nakamura, Hiroyuki Fujiwara, and Gaku Shoji. Building-damage detection method based on machine learning utilizing aerial photographs of the kumamoto earthquake. *Earthquake Spectra*, 36(3):1166–1187, 2020.

[15] Yundong Li, Wei Hu, Han Dong, and Xueyan Zhang. Building damage detection from post-event aerial imagery using single shot multibox detector. *Applied Sciences*, 9(6):1128, 2019.

[16] Jinyuan Shao, Lina Tang, Ming Liu, Guofan Shao, Lang Sun, and Quanyi Qiu. Bdd-net: A general protocol for mapping buildings damaged by a wide range of disasters based on satellite imagery. *Remote Sensing*, 12(10):1670, 2020.

[17] Diogo Duarte, Francesco Nex, Norman Kerle, and George Vosselman. Satellite image classification of building damages using airborne and satellite image samples in a deep learning approach. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4(2), 2018.

[18] Adrian Albert, Jasleen Kaur, and Marta C Gonzalez. Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1357–1366, 2017.

[19] Cade Crow, Melvin Juwono, Abuzar Royesh, and Tsion T Tesfaye. Satellite image analysis for building detection and natural disaster damage classification.

[20] Quoc Dung Cao; Youngjun Choe. Detecting damaged buildings on post-hurricane satellite imagery based on customized convolutional neural networks, 2018.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Appendix

Here are the details of the 'Video Action Classification' project we wanted to pursue but couldn't because of its computational complexity and the availability of limited computing resources. The link to the dataset is given for your reference. The code was compiled and run on google Collab.

Topic: Video action classification using deep learning

Data: 20 Bn-something-something

Total no. of videos: 220,847