

# Team ID - VB-215274

---

## Team Members

- a. Name: Tanishq Selot  
Contact: [me200003076@iiti.ac.in](mailto:me200003076@iiti.ac.in)
- b. Name: Mihir Karandikar  
Contact: [cse200001044@iiti.ac.in](mailto:cse200001044@iiti.ac.in)
- c. Name: Atharva Mohite  
Contact: [me200003016@iiti.ac.in](mailto:me200003016@iiti.ac.in)
- d. Name: Yeeshukant Singh  
Contact: [ee200002082@iiti.ac.in](mailto:ee200002082@iiti.ac.in)

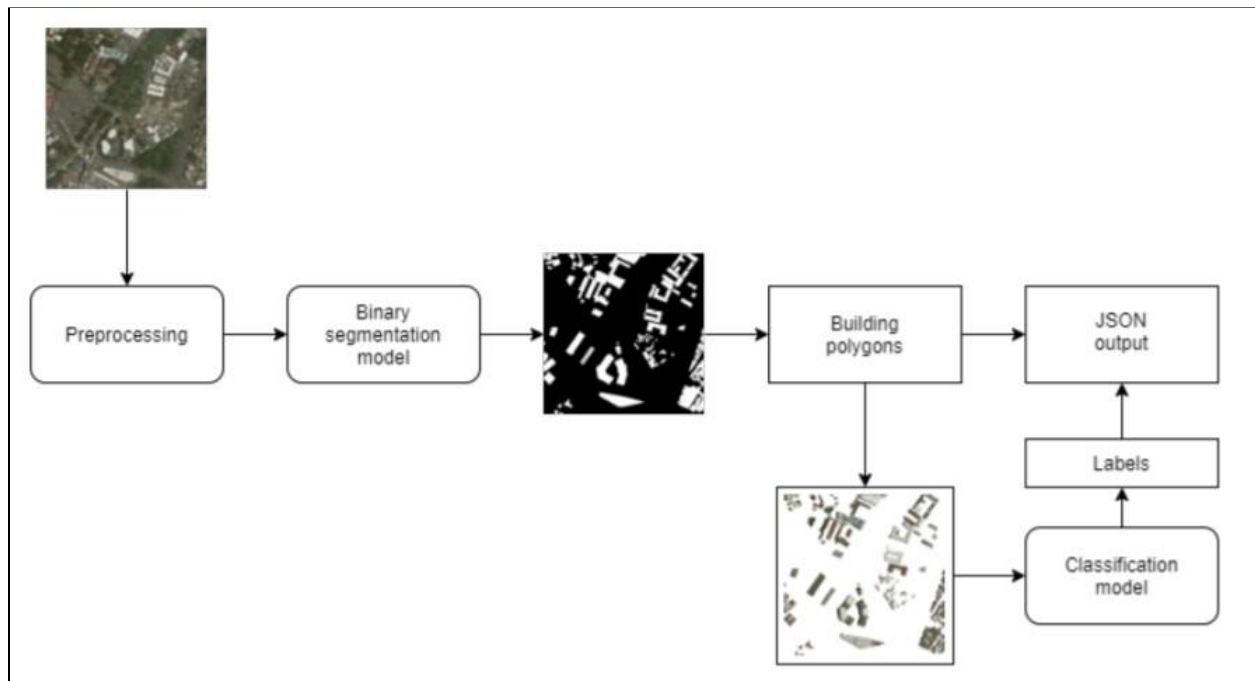
## 1. Introduction

Image classification has gained much attention due to its application in different computer vision tasks. One of them is satellite image classification. Nowadays, satellite images are widely used for Area classification and change detection. Classification is used for various applications like Crop monitoring, Mapping Soil characteristics and Forest cover, Natural disaster assessment and response, Environmental monitoring, Urban and Regional planning, and similarly other areas of interest. Therefore, it becomes vital to accurately extract information from satellite images and use it appropriately for classification and predictions due to its various significant benefits.

Many ML algorithms have been developed to extract useful information from the image data efficiently in recent years. The power of such algorithms depends on how we extract the information from the considerable amount of data found in satellite images. We have developed an algorithm **to estimate the damage to infrastructure** and classify damage severity in urban areas due to Earthquakes by performing instance segmentation. In instance segmentation or Simultaneous detection and segmentation (SDS) we label each foreground pixel with an object and instance.

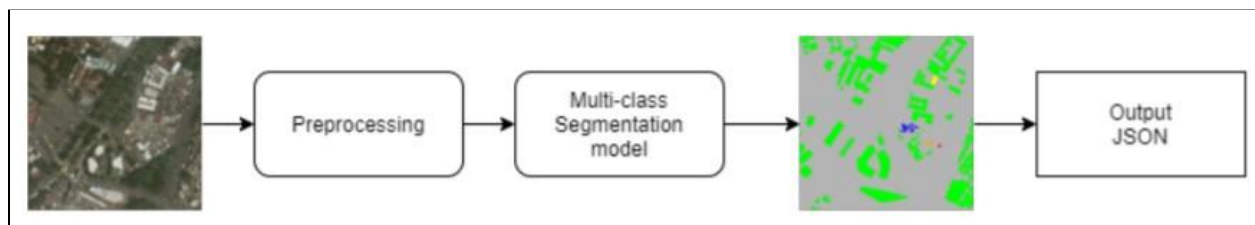
Following is what we understood from the problem statement that needed to be done. The given problem statement has been segregated into two sub-problems, namely :

- **Image Segmentation** - Labeling each pixel of the image as building/non-building.



- **Classification** - Classifying each building into 5 classes :

1. No damage
2. Minor damage
3. Major damage
4. Destroyed
5. Unclassified



Thus it has four-level damage annotation scale:-

Score	Label	Visual Description of the Structure
0	No damage	Undisturbed. No sign of water, structural damage, shingle damage, or burn marks.
1	Minor damage	Building partially burnt, water surrounding the structure, volcanic flow nearby, roof elements missing, or visible cracks.
2	Major damage	Partial wall or roof collapse, encroaching volcanic flow, or the structure is surrounded by water or mud.
3	Destroyed	Structure is scorched, completely collapsed, partially or completely covered with water or mud, or no longer present.

## 2. Classification Approach

### i. Motivation

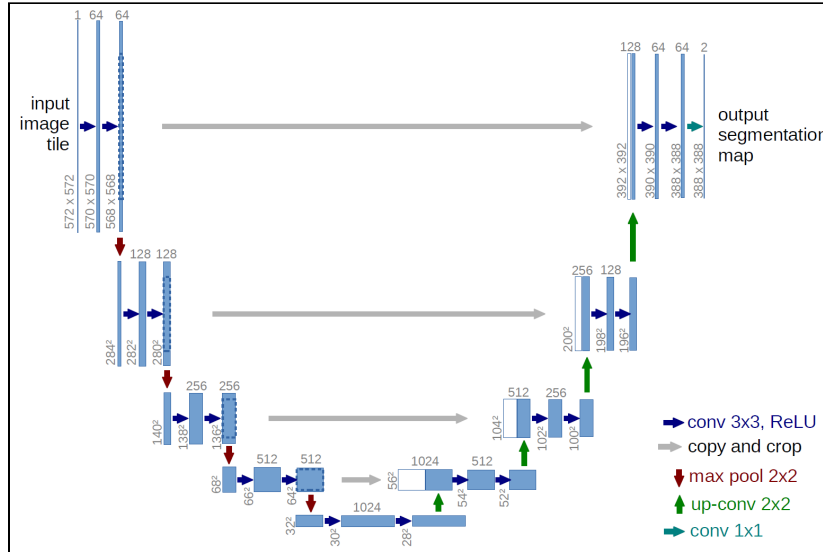
Since we perceived the problem as a combination of image segmentation and classification tasks, we searched for their solutions. We wanted a model efficient in segregation between buildings and other features while performing classification. Therefore, after a little search, we planned to implement a modified 'U-Net' architecture based on the Convolutional Neural Network. Initially introduced for Biomedical Image Segmentation, U-Net has proved to be very useful in segmentation tasks with modification for satellite image segmentation. **One important modification in U-Net is that there exist a large number of feature channels in the up-sampling part that allow the network to propagate context information to higher resolution layers.**

In order to handle the class imbalance, we tried to implement class weights during the training of the network. However what we found was that doing so did not have any major effect on the results. We also tried to implement sample weights for each sample individually, however that proved to have a negative impact on the final outcome.

Previously experimented model:

Another model we experimented with was **ResNet152 concatenated with our original U-Net model**. So this model basically had two U-Net architectures one after the other, both consisting of their encoding and decoding layers. However, due to the large size of

the model in comparison to the dataset, this model was not able to give satisfactory results.



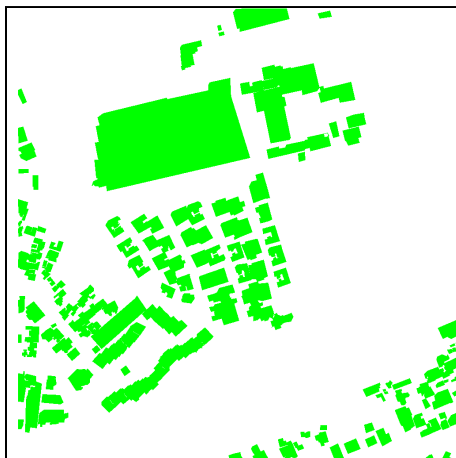
**Unet architecture**

Reasons for choosing Unet:

- Yielding precise results with fewer training images.
- Easy implementation.
- Easy aggregation with other classification architectures.
- The whole image is processed in forward pass leading to preservation of its full context. This is a big advantage over patch-based segmentation which is very localized.

## ii. Methodology

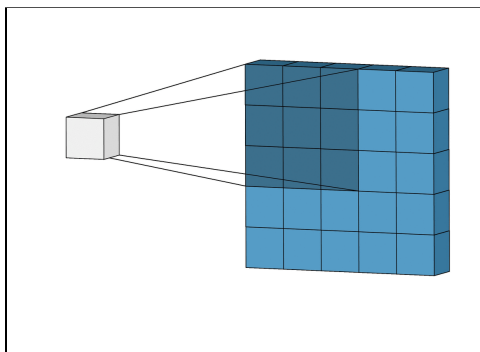
First, we made image masks for the training dataset images using the **JSON annotations** in the Labels folder.



*segmented\_img\_mexico-earthquake\_00000001\_post\_disaster*

### Pre-processing:

- Training images and masks were converted to numpy arrays.
- They were **resized to 512x512 px from 1024x1024 px** using OpenCV.
- Image numpy array was normalized by dividing image array by 255 as the pixel values range from 0 to 256, apart from 0 the range is 255. So dividing all the values by 255 will convert it to range from 0 to 1. This is done to avoid complex computations while passing through a deep neural network.
- We did a **90%-10% train-test split**.
- Different colours in the mask represent different categories (6 classes) represented by integers in the numpy array. These integers are converted into binary values (**one-hot encoding**) and the columns in the numpy array equal the number of categories.
- *One hot encoding is better than label encoding because label encoding leads to each class having a particular rank/hierarchy.*
- Reshape to (512,512,3) for inputting to the Unet model.
- *Data augmentation cannot be done in tasks involving annotations as cropping/rotations will lead to change in location of particular categories.*



**Convolutions can happen faster when fewer pixels are involved.**

### Model:

- Input shape - (512,512,3)
- The model used has a classical U-Net architecture consisting of the encoding part, the decoding part, and concatenations between the two parts.
- Input shape = (None,512,512,3)
- Output shape = (None,512,512,6)
- Number of Convolutional Layers in the encoding part = 12

- Number of Transpose Convolutional Layers in the decoding part = 5
- Number of Convolutional Layers in the decoding part = 11
- Number of concatenations = 5

Dropout Layers:

- 2 layers with 0.1 Dropout in Encoding part
- 2 layers with 0.2 Dropout in Encoding part
- 2 layers with 0.3 Dropout in Encoding part
- 3 layers with 0.1 Dropout in Decoding part
- 2 layers with 0.2 Dropout in Decoding part

Compiling the model:

- As each building pixel is to be classified into 6 classes (categories), we used **categorical\_crossentropy** loss function.
- We used **Adam** optimizer because:

*1. It maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).*

*2. The learning rates are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). Hence, it is useful when the data is noisy.*

Fitting the model:

- The model was trained for 10 epochs.
- Batch size = 16
- Verbose=1. Gives us the details about how the model improved after each epoch.
- 10% of the test dataset was also used as the validation dataset.

### iii. Implementation

**W**e have constructed a multi U-Net model.

Layers of U-Net:

- The **2D Convolutional layer** performs a convolutional operation on an image of size ( $n_{in} \times n_{in} \times \text{channels}$ ) to compute an output ( $n_{out} \times n_{out} \times k$ ) where  $k$  is the number of kernels for the layer.

**Function:** `keras.layers.Conv2D(filters, kernel_size, padding=same, activation=relu, kernel_initializer=he_normal)`

- **Dropout layer** randomly sets the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.

**Function:** `tf.keras.layers.Dropout(rate)`

- **Transposed convolutional layer** carries out upsampling i.e. to generate an output feature map that has a spatial dimension greater than that of the input feature map.

**Function:** `tf.keras.layers.Conv2DTranspose(filters, kernel_size, padding=same, strides=(2,2))`

- **Batch normalization layer** standardizes the inputs to a layer for each mini-batch, thus reducing the number of training epochs required to train deep networks.

**Function:** `tf.keras.layers.BatchNormalization()`

- **Max-pooling layer** down samples the feature map by calculating the maximum value for each patch of the feature map.

**Function:** `tf.keras.layers.MaxPooling2D(pool_size=(2, 2))`

### Activation function:

The activation function is a non-linear transformation that we do over the input before sending it to the next layer of input or finalizing it as output. We have employed a softmax activation function for the output layer. **Softmax** activation function predicts the multinomial probability distribution.

### Optimizer:

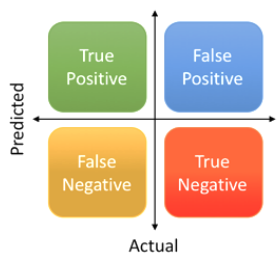
Optimizer is an algorithm used to reduce the loss function and update the weights in backpropagation. It is one of the arguments of the `model.compile()` function of keras. After comparing the performance of the model for various optimizers and analyzing their advantages, we concluded that **adam** is the best optimizer for gradient descent.

### Characteristics of the U-Net model:

Input image width	512
Input image height	512
Number of channels	3
Number of classes	6

Total number of parameters	7,775,686
Number of trainable parameters	7,775,686
Number of non-trainable parameters	0
Batch size	2
Number of epochs	20
Kernel size	3x3 and 2x2
Total execution time	574 seconds
Activation function at the output layer	Softmax
Optimizer	Adam

### 3. Results and Evaluation Metrics

<p><b>Precision</b> = <math>\frac{\text{True Positive}}{\text{Actual Results}}</math> or <math>\frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}</math></p> <p><b>Recall</b> = <math>\frac{\text{True Positive}}{\text{Predicted Results}}</math> or <math>\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}</math></p> <p><b>Accuracy</b> = <math>\frac{\text{True Positive} + \text{True Negative}}{\text{Total}}</math></p>	 <p>A 2x2 confusion matrix diagram. The vertical axis is labeled 'Predicted' and the horizontal axis is labeled 'Actual'. The four quadrants are: Top-Left (Green) 'True Positive', Top-Right (Blue) 'False Positive', Bottom-Left (Yellow) 'False Negative', and Bottom-Right (Red) 'True Negative'.</p>
--	---

**Intersection over Union (IoU)** is used when calculating mAP. It is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box.

Metric:	Score:
1. Accuracy	82.74 %
2. IoU	0.4100
3. Precision	0.8182
4. Recall	0.8274



## 4. Conclusion

---

In this study, a Unet approach is proposed for instance segmentation of the given dataset of post earthquake satellite imagery. The given dataset is exiguous and highly imbalanced towards the *No damage* class. We tried to address this skewness by tweaking the class weights.

We also constructed two U-net architectures using Resnet152.

We fine tuned our final Unet model by varying the various hyperparameters of the model. Using the proposed Unet model, we can directly implement a multi-class classification, without segregating buildings and other features on the feature map. We have succeeded in achieving an overall accuracy of 82.74% and an IoU of 0.41. Employing state of the art algorithms to extract features from the limited satellite image data is a part of our future work. A further understanding of the U-Net accuracy and sensitivity to class characteristics would surely improve the accuracy.

**End**

---