Team Details

- NEEL SHAH 9653380766 neelshah29042002@gmail.com
- DHRUV KUNJADIYA 9321784815 dhruvkunjadiya55@gmail.com
- PRATHAM SHAH 9653179913 ppsprathamshah2313@gmail.com
- ARNAV ZUTSHI 8448612318 arnzut1324@gmail.com

Introduction

Problem Statement

We need to extract data from the post earthquake imagery training set provided to us and classify buildings on the basis of how damaged they are.

The buildings are classified into 5 types .i.e:

1. Destroyed buildings
2. Major Damaged buildings
3. Minor Damaged buildings
4. No damaged buildings
5. Unclassified

SUBTYPE		REPRESENTATION
Unclassified	**0**	**for buildings which could not be classified**
No damaged	**1**	**for building found and classified no-damaged**
Minor Damaged	**2**	**for building found and classified minor-damage**
Major Damaged	**3**	**for building found and classified major-damage**
Destroyed	**4**	**for building found and classified destroyed**

We further need to assign their respective colours to the classified objects in the image. This data is then to be trained in our network and should give us the classification with colour-codes as given. This will require the concepts of deep learning and computer visions to be thoroughly applied.

Need for satellite image classifier:

It is very important to get the information about the region which was affected by the disaster. Without knowing the proper information about damage caused by disaster we cannot think about taking further steps. The damage classifier is essential for taking post disaster action as it helps in specifying the most affected areas and taking necessary action accordingly. The satellite image classifier helps in providing continuous good quality images which can be used to compare the pre and post disaster conditions.

Project Overview

Aim is to propose and implement a multi-class classification approach for disaster assessment from the given data set of post earthquake satellite imagery. We as a team have tried to give a well structured report detailing the approach used for object detection and its implementation. We will be using supervised learning to extract the data as we are provided with a labelled data set.

Classification Approach

i. Motivation

Why use deep learning?

Good performance of deep learning algorithms is limited to the size of data available, and the network structure is considered. One of the most critical challenges for using a deep learning method for monitoring the buildings damaged in the disaster is that the training images of damaged targets are usually not very much.

Why convolutional neural networks?

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

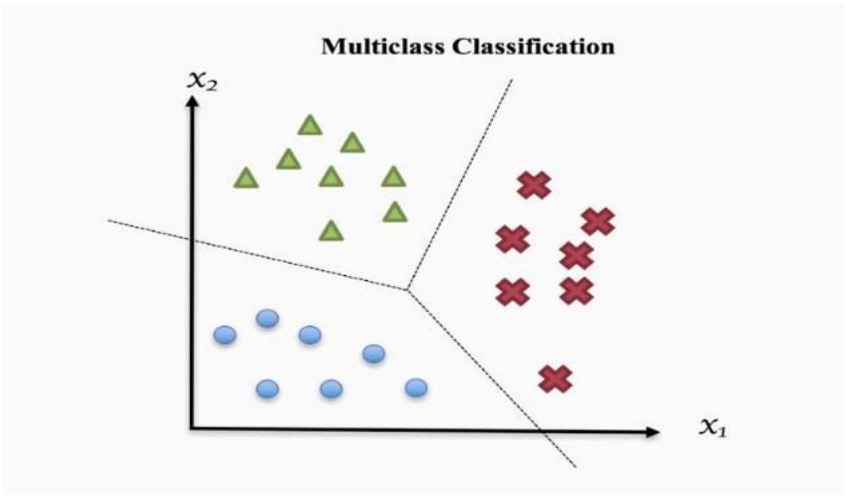
- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colours and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

Why multi-class classification?

When we solve a classification problem having only two class labels, then it becomes easy for us to filter the data, apply any classification algorithm, train the model with filtered data, and predict the outcomes. But when we have more than two class instances in input train data, then it might get complex to analyze the data, train the model, and predict relatively accurate results. To handle these multiple class instances, we use multi-class classification. Multi-class classification is the classification technique that allows us to categorize the test data into multiple class labels present in trained data as a model prediction.There are mainly two types of multi-class classification techniques:-

- One vs. All (one-vs-rest)
- One vs. One



ii. Methodology

Approach for Multi-Class Classification

One of the numerous fundamental tasks to perform rescue operations after an earthquake is to check the status ofWe buildings that have been destroyed.
First thing that we need to do is building detection. For building detection we can use a convolutional neural network model which will detect the buildings from satellite imagery. After detecting the buildings we can use a CNN model like U-Net which is used mainly for image segmentation to classify the images as per their respective color-codes.

Activation Function used:

Softmax activation function is used as an activation function in our case as we are doing multi class classification.What it does is that in N values network configurations classification tasks having N classes as the output, the softmax function normalizes outputs by assigning probabilities to the sum of weighted values such that the sum of probabilities is equal to1. Thus each class has its own probability distribution in the class membership. The Softmax function can then be described as the mathematical function converting the output into probability vector numbers from the input vector numbers with each vector value’s probabilities proportional to its relative value vector scale.

Optimiser used: Adam Optimiser

Adam optimizer involves a combination of two gradient descent methodologies:

- Momentum:

  This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the ‘exponentially weighted average’ of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

- Root Mean Square Propagation (RMSP):

  Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the ‘exponential moving average’.

Adam Optimizer inherits the strengths or the positive attributes of the above two methods and builds upon them to give a more optimised gradient descent.

Loss functions tried:

- Focal Loss:

By using Focal Loss we can reduce the imbalance in the dataset. We have tried focal loss as a loss function in our problem as the classes were highly imbalanced, focal loss can be useful in such cases.

- Categorical_crossentropy:

Categorical cross entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.Our problem was on a similar basis so we tried it.

iii. Implementation

Tech-stack:

- Tensorflow library-Python:
- Keras-Python:
- Matplotlib:
- Numpy
- SkLearn

Workflow:

- We first formed **masks** using skimage and matplotlib libraries using the provided JSON data to us. This process of getting masks was needed for **training our model**.

- Next , we have **defined our model**. After defining the model we are **reading images** and masks and **resizing** the images and masks.
- Now we will **encode** our labels and further **split** our data set. We have defined **sample weights** as our data was **highly imbalanced**. After defining class weights we have defined the loss function.
- We have used **focal loss** as our loss function and optimizer used was **Adam** and the metric function used was **accuracy.** After defining the loss function , optimizer , and accuracy metrics we are ready to **train** our model.
- We have tried different things while training the model like **changing the number of epochs** , changing the **weights** that we have defined , also we have changed the **image size** to see if there is any change in the accuracy or not. Different loss functions were also tried and some of them which were better are mentioned above in the methodology section.
- After training the model **we are saving** the model to use it while testing. Finally we are ready to test our model and after testing we have shown the results that we got.

Different things tried:

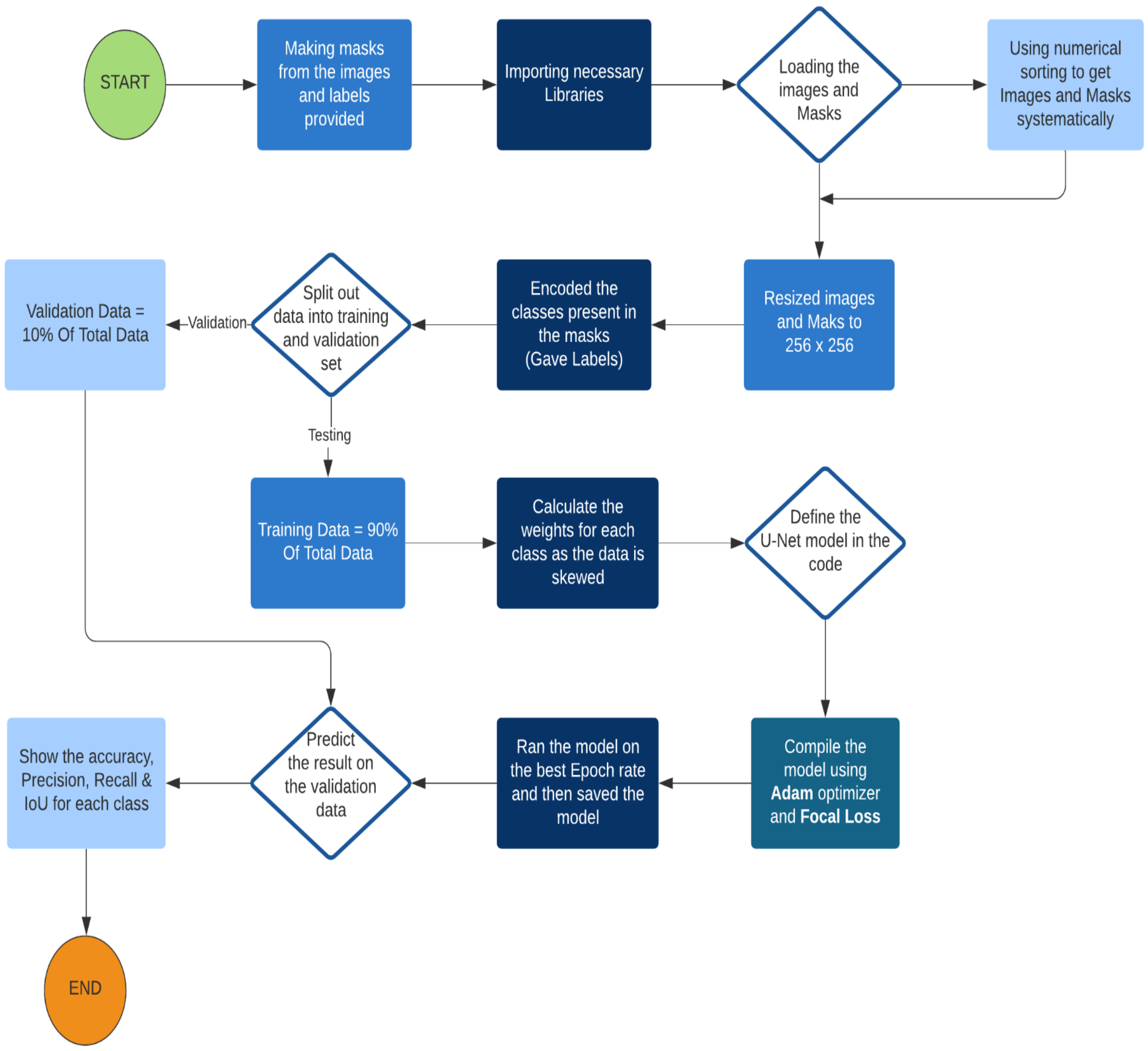
- One of the major issues faced during the implementation was due to less dataset. For this issue we have tried different things like data augmentation to increase the data set but it was not giving the expected results as in our case we realized that if we augment the data then minority classes are increasing but with that the major classes are increasing in huge amounts compared to the minority classes.

SUB-TYPE	OCCURRENCE (NUMBER)
NO-DAMAGE	31075
UN-CLASSIFIED	75
MINOR-DAMAGE	18
MAJOR-DAMAGE	104
DESTROYED	2

As you can see in the above table the occurrence of DESTROYED sub-type is very less so it is very difficult for the model to classify destroyed types.

- We have also used SMOTE to reduce the imbalance dataset. The SMOTE class acts like a data transform object from scikit-learn in that it must be defined and configured, fit on a dataset, then applied to create a new transformed version of the dataset.
- For example, we can define a SMOTE instance with default parameters that will balance the minority class and then fit and apply it in one step to create a transformed version of our dataset.

Code Workflow:


START
Making masks from the images and labels provided
Importing necessary Libraries
Loading the images and Masks
Using numerical sorting to get Images and Masks systematically
Resized images and Maks to 256 x 256
Encoded the classes present in the masks (Gave Labels)
Split out data into training and validation set
Validation
Validation Data = 10% Of Total Data
Testing
Training Data = 90% Of Total Data
Calculate the weights for each class as the data is skewed
Define the U-Net model in the code
Compile the model using **Adam** optimizer and **Focal Loss**
Ran the model on the best Epoch rate and then saved the model
Predict the result on the validation data
Show the accuracy, Precision, Recall & IoU for each class
END

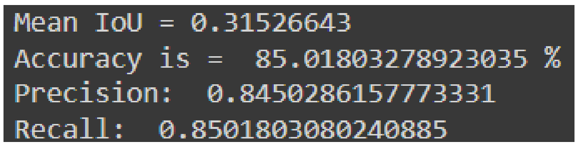
Results and Evaluation Metrics

i. Accuracy: `85.01803278923035 %`

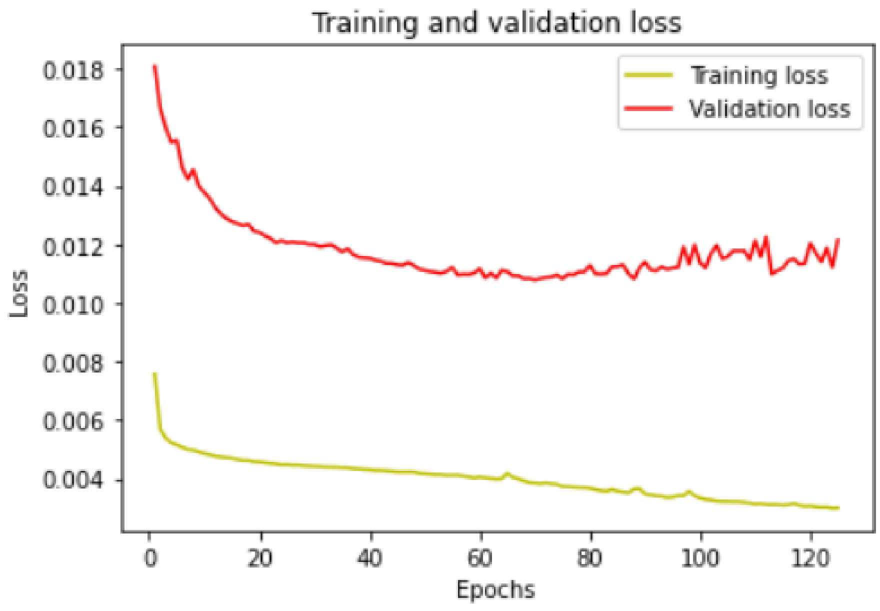
ii. IoU: `0.31526643`

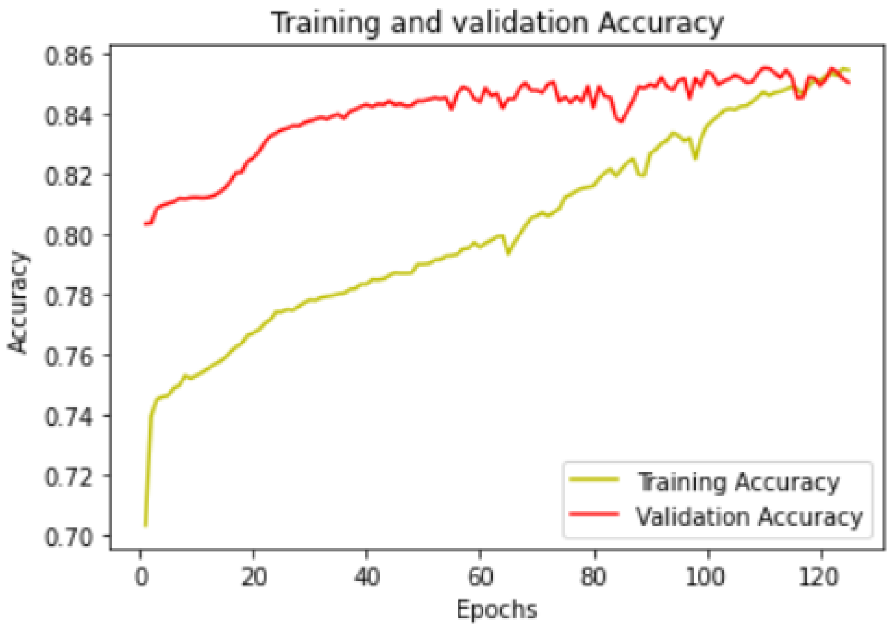
iii. Precision: `0.8450286157773331`

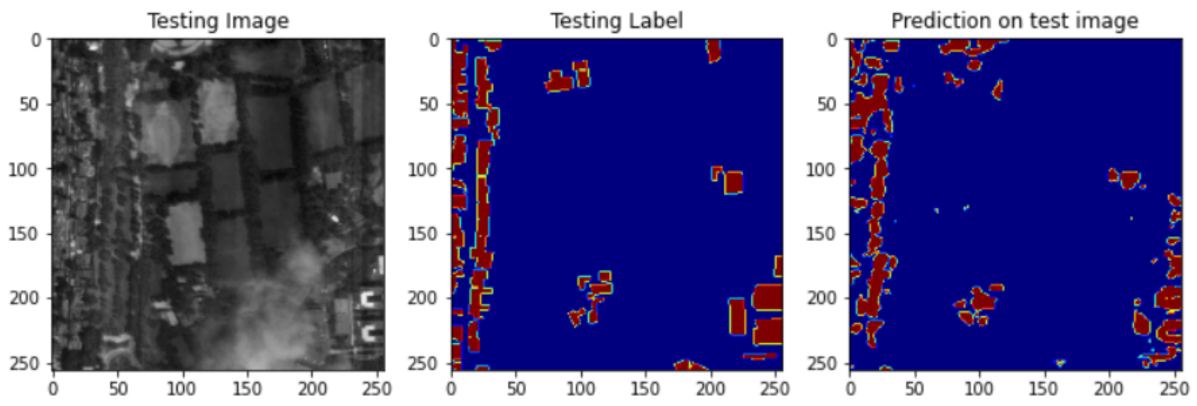
iv. Recall: `0.8501803080240885`

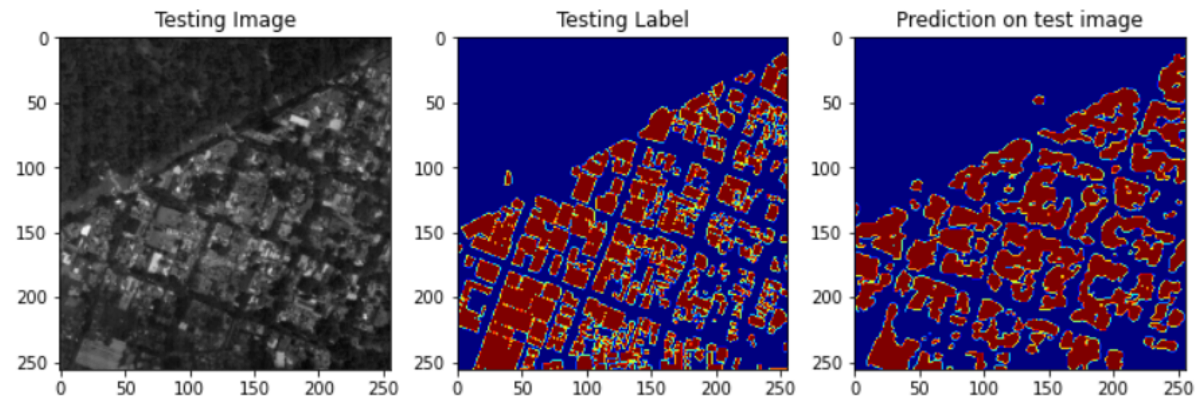


```
Mean IoU = 0.31526643
Accuracy is =  85.01803278923035 %
Precision:  0.8450286157773331
Recall:  0.8501803080240885
```









Conclusion

It was a very good experience for us to try this problem and we have done our best to find a solution for this problem statement. We have learned many things like how to search for errors and find solutions for it without wasting more time on it.We also learned how to coordinate in a team and work together and how to read a statement carefully and proceed towards it.