Course: MACHINE LEARNING

Course Id: ESC 211

Submitted By :

18104113 Riya Handa

18104114 Ishan Kansal

18104118 Chirag Mittal

18104120 Ritvik Bhat

C





Application of ML/DL based techniques on satellite images for post-disaster damage assessment.

Contents

[Introduction 1](#_Toc40461226)

[Problem statement 2](#_Toc40461227)

[Dataset Used 3](#_Toc40461228)

[Model Used: 4](#_Toc40461229)

[Methodology 5](#_Toc40461230)

[Convolutional Neural Network- 8](#_Toc40461231)

[Prediction 10](#_Toc40461232)

[Conclusion: 10](#_Toc40461233)

# Introduction

Weather related natural disasters cost the world economy around 100 billion dollars every year. According to the Centre for Research on the Epidemiology of Disaster, 69.800 deaths per year are inﬂicted as the result of these disasters and earthquakes. The effects are felt around the world; however, most deaths occur in low- or middle-income areas. Advances in technology and preparedness have decreased the amount of deaths caused by natural disasters since the second part of the previous century. However, due to an increase in the frequency of disasters, more people are affected and more damages occur; with the most economic damage recorded in 2011.

When a disaster strikes, quick and accurate situational information is critical to an effective response. Before responders can act in the affected area, they need to know the location, cause, and severity of the damage. However, disasters can strike anywhere, disrupting local communication and transportation infrastructure, making the process of assessing specific local damage difficult, dangerous, and slow. This project is related to programming for automation in damage assessment during disaster using ML/DL techniques on remotely sensed data.

The rapid and accurate acquisition of disaster losses can provide great help for disaster emergency response and decision-making. Remote sensing (RS) and Geographic Information System (GIS) can help assess earthquake damage within a short period of time after the event.

But if a manually evaluated image is used to evaluate a damaged area, large false and missed detections could occur owing to the inﬂuence of subjective human factors. Therefore, the processing of aerial images to identify and assess the extent of damage in an area is a challenging task.

With the advancement in machine learning and deep learning, it became easy to evaluate an image for evaluating a damaged area.

In this project, we first detect buildings in our dataset of images using popular YOLO framework and then we propose an approach based on CNNs and ordinal regression (OR) aiming at assessing the degree of building damage caused by disasters with aerial imagery. CNNs hierarchically extract useful high-level features from input building images, and then OR is used to classify the features into four different damage grades. Then, we can get the degree of damaged buildings.

OR (also called “ordinal classification”) is used to predict an ordinal variable. In this project, the building damage degree, on a scale from “no observable damage” to “collapse”, is just an ordinal variable. However, typical multiclass classification ignores the ordered information between the damage degrees, while damage degrees have a strong ordinal correlation. Thus, we cast the assessment problem of the degree building damage as an OR problem and develop an ordinal classifier and corresponding loss function to learn our network parameters. Information utilization was improved by OR, so we can achieve a better accuracy with the same or a lesser amount of data.

So basically, we are creating a model that can extract building polygons and assess the building damage level of polygons on an ordinal scale.

# Problem statement

Given a satellite image, detect the buildings and extract the information (polygons) of the buildings in that image implementing the convolutional neural network (CNN) using the popular YOLO framework and then do assessment of the Degree of Building Damage Caused by Disaster Using Convolutional Neural Networks in Combination with Ordinal Regression.

# Dataset Used

The data that we are using is a type of unstructured data which unlike structured data is not in the form of a Data-frame containing rows and columns (Excel sheets, csv files). The Dataset we are using contains data divided into 4 folders consisting of

1. test

2. test-another (for test set)

3. train-another (for the training set)

4. validation-another (for the validation set)

Each folder consists of two folders (damage and no-damage) consisting of images of buildings.Damage Folder has images of damaged buildings as well as images of buildings that are not damaged.

The data are satellite images from Texas after Hurricane Harvey divided into two groups (damage and no-damage). The goal is to make a model which can automatically identify if a given region is likely to contain flooding damage.

-> test (balanced test data) - 1000 images for each class.

-> test-another (unbalanced test data) - 8000(damaged-class) and 1000(undamaged-class) images.

-> train-another (the training data) - 5000 images for each class.

-> validation-another (for the validation set) - 1000 images for each class.

**Data-Set Source:** <https://www.kaggle.com>

**Data-Set Link:** [**https://drive.google.com/open?id=1grtr6Fo87mChDRm677Ws\_xjj-835CbxH**](https://drive.google.com/open?id=1grtr6Fo87mChDRm677Ws_xjj-835CbxH)

In this project we will instead give damaged-buildings as well as undamaged-buildings images collectively and randomly with the objective of measuring the intensity of the damage to the building. As a result of this we can classify whether building is damaged or undamaged because with no intensity of damage automatically means that the building is undamaged.

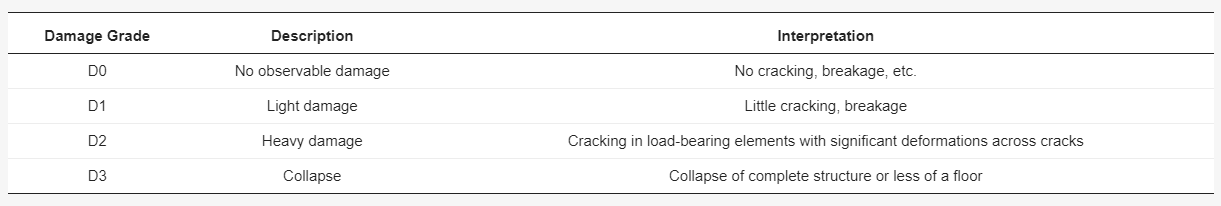
Training a YOLO model takes a very long time and requires a fairly large dataset of labelled bounding boxes. Therefore, we are going to load an existing pre-trained Keras YOLOv2 model stored in "yolo.h5".

The images cannot be directly input into the deep learning model. They need to be cut to obtain image blocks of the size specified in the YOLO (You Look Only Once) Model/Algorithm. For labelling of the collapsed buildings LabelImg software is being used.

Further we can do data enhancement of our dataset just like other images by flipping, rotating, increasing noise, colour transformation, etc... Generally, we do rotate or flipping by 90’, 180’, 270’ to improve detection performance of the model and colour transformation eliminates the influence of colour deviation on model’s performance.

For assessing the degree of building damage, a well-labelled dataset is very important, as it is used for training. Images of buildings at all levels of damage from the Texas after Hurricane Harvey were used to construct the dataset.

Before training the model, we needed to build a building dataset of different damage degrees. Building damage was classified into four classes D0, D1, D2, D3.



Our labelled Y- vector for each damage grade-

D0- (1, 0, 0, 0)

D1- (1, 1, 0, 0)

D2- (1, 1, 1, 0)

D4- (1, 1, 1, 0)

# Model Used:

1. Convolutional Neural Networks (CNN)

2. Keras YOLOv2 (Pre-Trained Model)

3. Neural Networks Approach towards Ordinal Regression

# Methodology

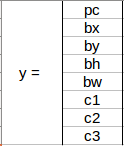
First, we detect buildings (damaged or undamaged) in our dataset of images using popular YOLO framework.

The YOLO framework (You Only Look Once) takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it’s incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.

Here are the steps followed by YOLO for detecting objects in a given image-

* YOLO first takes an input image.
* The framework then divides the input image into grids (say a 3 X 3 grid).
* Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found, of course).

We need to pass the labelled data to the model in order to train it. Suppose we have divided the image into a grid of size 3 X 3 and there is only one class which we want the object to be classified into and that is “building”.



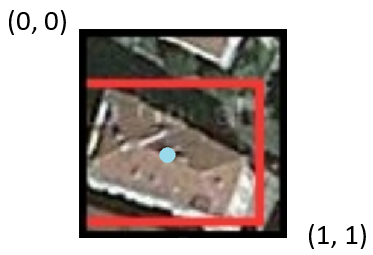
Here,

* pc defines whether an object is present in the grid or not (it is the probability).
* bx, by, bh, bw specify the bounding box if there is an object.
* c1 represent the class “building”.

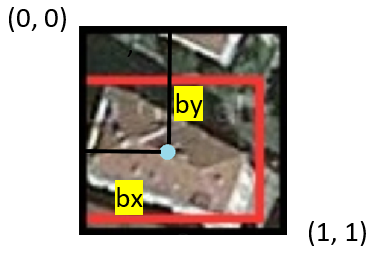
Before we write the y label for any grid, it’s important to first understand how YOLO decides whether there actually is an object in the grid. In the image, let say there are two objects (two buildings), so YOLO will take the mid-point of these two objects and these objects will be assigned to the grid which contains the mid-point of these objects.

We will run both forward and backward propagation to train our model. During the testing phase, we pass an image to the model and run forward propagation until we get an output y.

Now, let’s see how to decide bx, by, bh, and bw. In YOLO, the coordinates assigned to all the grids are:



bx, by are the x and y coordinates of the midpoint of the object with respect to this grid. In this case, it will be (around) bx = 0.4 and by = 0.7:



bh is the ratio of the height of the bounding box (red box in the above example) to the height of the corresponding grid cell, which in our case is around 0.9. So, bh = 0.9. bw is the ratio of the width of the bounding box to the width of the grid cell. So, bw = 0.5 (approximately).

bx and by will always range between 0 and 1 as the midpoint will always lie within the grid. Whereas bh and bw can be more than 1 in case the dimensions of the bounding box are more than the dimension of the grid.

Now we have to decide whether the predicted bounding box is giving us a good outcome (or a bad one). For this, we calculate intersection over union. IoU, or Intersection over Union, will calculate the area of the intersection over union of the actual bounding box and the predicted one.

If IoU is greater than 0.5, we can say that the prediction is good enough. 0.5 is an arbitrary threshold we have taken here, but it can be changed according to our needs. Intuitively, the more you increase the threshold, the better the predictions become.

One of the most common problems our building detection algorithm is that rather than detecting the building just once, they might detect it multiple times.

The Non-Max Suppression technique is useful here so that we get only a single detection per object.

* Discard all the boxes having probabilities less than or equal to a pre-defined threshold (say, 0.5).
* For the remaining boxes:
  1. Pick the box with the highest probability and take that as the output prediction.
  2. Discard any other box which has IoU greater than the threshold with the output box from the above step.
* Repeat step 2 until all the boxes are either taken as the output prediction or discarded.



**Bounding boxes simplification**

Now, we are assessing the degree of building damage caused by disasters using convolutional neural network and ordinal regression.

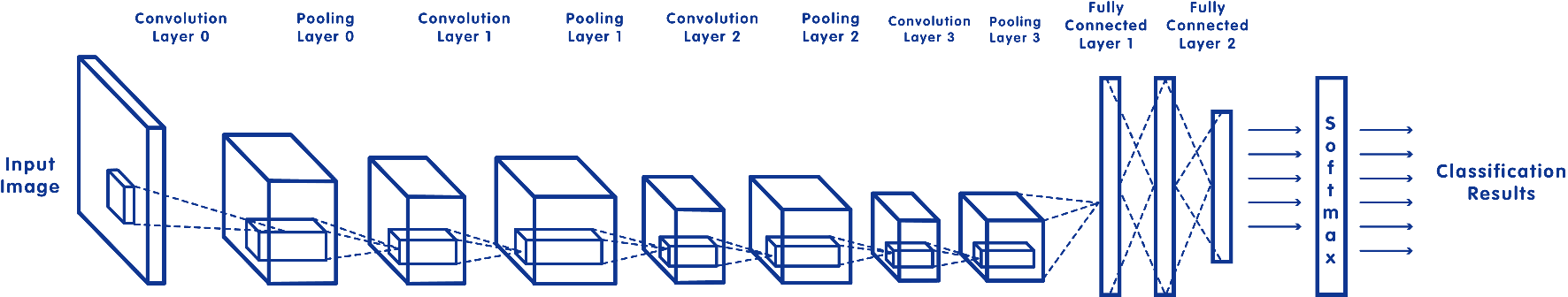
# Convolutional Neural Network-

The convolution layer convolves the input image with a set of learnable filters, each producing one feature map in the output image. After crossing a nonlinear activation layer, it can get the picture feature of the next layer. The input feature map is compressed in the pooling layer. On the one hand, the pooling layer shrinks the feature map and simplifies the network-computing complexity. On the other hand, it compresses and extracts the main features. Generally, there are two kinds of operations in the pooling layer: max pooling and average pooling. In this project, max pooling is adopted. The fully connected layer can connect all the features and convey results to the classifier.

Parameters of CNN can be obtained by training. The training includes two processes: forward and back propagation. Forward propagation calculates the classification results of samples by current network weights. Back propagation compares the calculated classification results with true values, and then updates the network weights backward, layer by layer.

**Ordinal regression-**

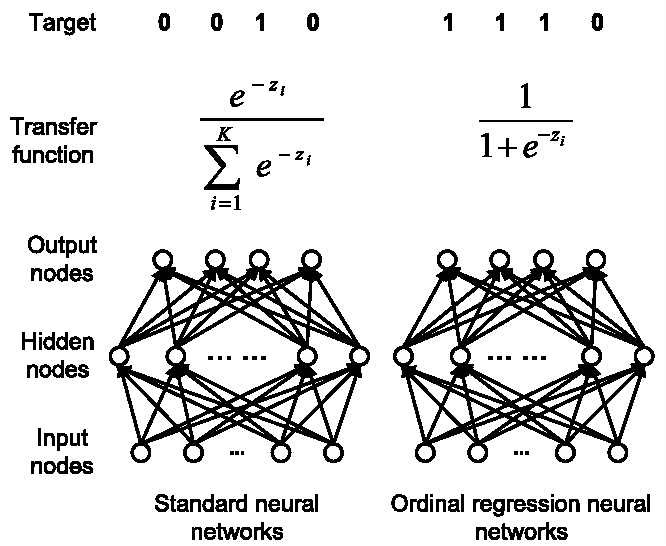
ordinal regression (also called “ordinal classification”) is a type of regression analysis used for predicting an ordinal variable, i.e. a variable whose value exists on an arbitrary scale where only the relative ordering between different values is significant.



**Neural network approach to ordinal regression-**

Our neural network approach considers the order of the categories. If a data point x belongs to category k, it is classified automatically into lower-order categories (1, 2, ..., k − 1) as well. So, the target vector of x is t = (1, 1, ..., 1, 0, 0, 0), where ti (1 ≤ i ≤ k) is set to 1 and other elements zeros. Thus, the goal is to learn a function to map the input vector x to a probability vector o = (o1, o2, ..., ok, ...oK), where oi (i ≤ k) is close to 1 and oi (i ≥ k) is close to 0.

Each output node Oi of our neural network uses a standard sigmoid function.



Output node Oi is used to estimate the probability oi that a data point belongs to category i independently, without subjecting to normalization as traditional neural networks do. Thus, for a data point x of category k, the target vector is (1, 1, .., 1, 0, 0, 0), in which the first k elements is 1 and others 0. This sets the target value of output nodes Oi (i ≤ k) to 1 and Oi (i > k) to 0. The targets instruct the neural network to adjust weights to produce probability outputs as close as possible to the target vector.

Training of the neural network for ordinal regression proceeds very similarly as standard neural networks. The cost function for a data point x can be relative entropy or square error between the target vector and the output vector. For relative entropy, the cost function for output nodes is fc = Σ (ti log oi + (1 − ti) log (1 − oi)), i=1 to k.

In our project value of K=4.

# Prediction

In the test phase, to make a prediction, our method scans output nodes in the order O1, O2, ..., OK. It stops when the output of a node is smaller than the predefined threshold T (e.g., 0.5) or no nodes left. The index k of the last node Ok whose output is bigger than T is the predicted category of the data point.

# Conclusion:

In this project we aimed at achieving the degree of damage of buildings as output by giving input as images so as to give information about the level of damage in an area. Firstly, we used the popular YOLO Algorithm which takes the entire image in a single instance and predicts the bounding box coordinates thus giving us the coordinates of the objects. We used a pre-trained model for this as training required a very long time as well as dataset. Then we applied Convolutional Neural Networks where the convolution layer convolves the input image with a set of learnable filters, each producing one feature map in the output image. We extracted the features and then used ordinal regression by neural network to get the respective probabilities. After scanning of the output nodes we’ll get the actual index hence the degree of damage of the building. This project involved applying & using multiple domains & models including YOLO, CNN, Ordinal Regression and can be a big asset to the organizations responsible for monitoring damage assessment of cities by providing degree of damage of buildings thus giving quick, accurate information regarding status of area in terms of damage instead of manually assessing specific local damage which can be slow as well as difficult.