Configuration Manual

A Hybrid Model of Sectorization & Evacuation Path Detection for Disaster Affected Area

MSc in Data Analytics

Vinaysheel Kishor Wagh
Student ID: x18194303

School of Computing
National College of Ireland

Supervisor:
Dr. Paul Stynes
Dr. Pramod Pathak
Dr. Luis Gustavo Nardin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vinaysheel Kishor Wagh
Student ID:	X18194303
Programme:	MSc. Data Analytics **Year:** 2019-2020
Module:	Research Project
Lecturer:	Dr. Paul Stynes, Dr. Pramod Pathak, Dr. Luis Gustavo Nardin
Submission Due Date:	17/08/2020
Project Title:	A Hybrid Model of Sectorization & Evacuation Path Detection for Disaster Affected Area
Word Count:	**988** **Page Count: 15**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vinaysheel Kishor Wagh

Date: 16/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vinaysheel Kishor Wagh
Student ID: x18194303

1 Introduction

This document contains all the required steps to reproduce the implementation of the proposed hybrid model of sectorization and evacuation path detection for disaster affected area. This document also contains hardware specification and system requirements used in the research work which can be considered as minimum recommended system specification.

2 System Requirements

2.1 Hardware Configuration

Figure 1 shows the configuration of DELL laptop which is used for the research. The laptop has intel core i5-8250U processor with 8 GB RAM and 1 TB Hard drive. It is using Windows 10 Home edition operating system.

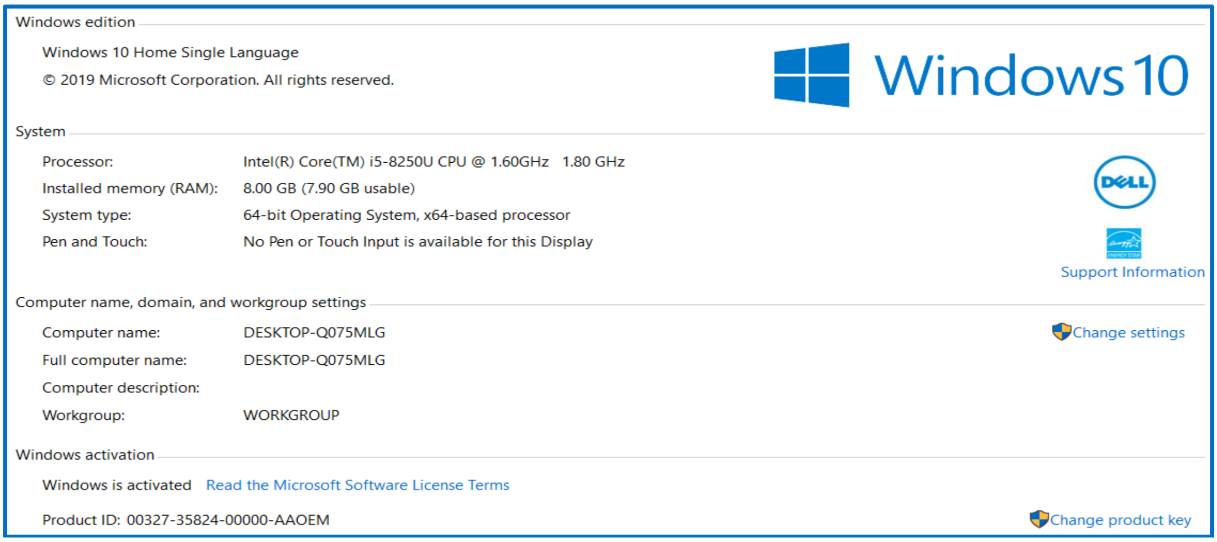


Figure 1: System Specification

2.2 Software Configuration

The project is implemented using Google Collaboratory and R studio software. Next section briefly explains all the steps required to download and install the software packages.

3 Environment Setup

3.1 Google Collaboratory Notebook

The hardware configuration of the laptop was not enough to execute deep learning models. So, Google Collaboratory is used to efficiently run all the deep learning models of the project. Follow following steps to set-up a Google Collaboratory (collab).

1. First Create a Gmail account.
2. Open this link in Google Chrome and following screen will appear.

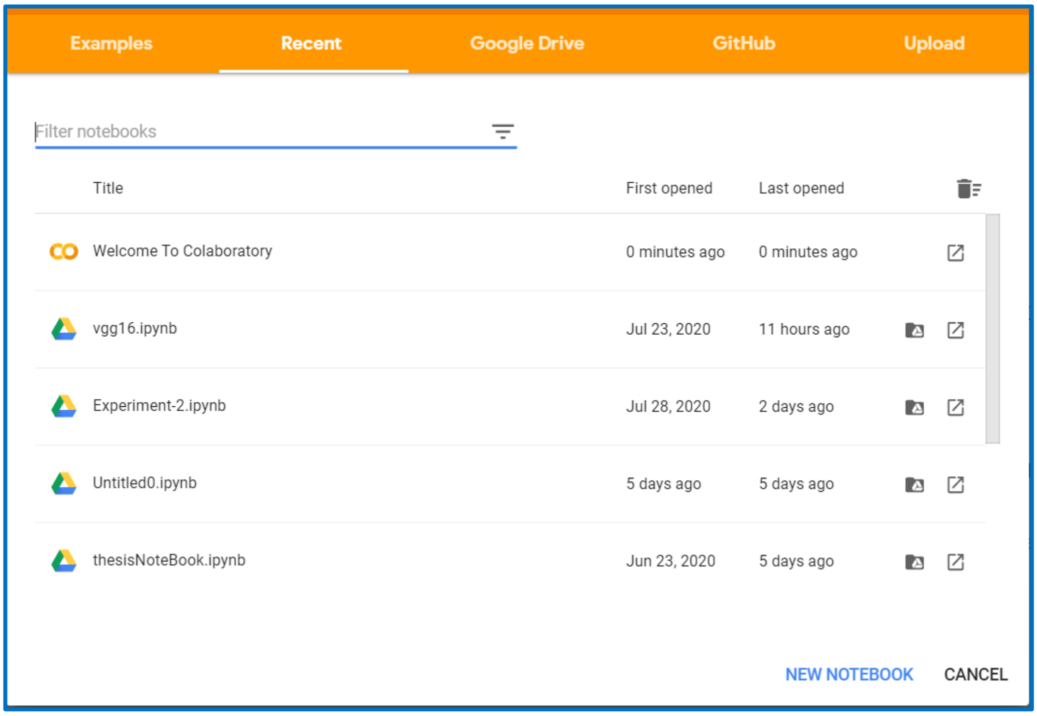


Figure 2: Create Notebook

3. Click on **New Notebook** or click **Upload** to use the existing notebooks.
4. After creating a notebook change the runtime type of the particular notebook to GPU. For this click Runtime and then click on Change runtime type.

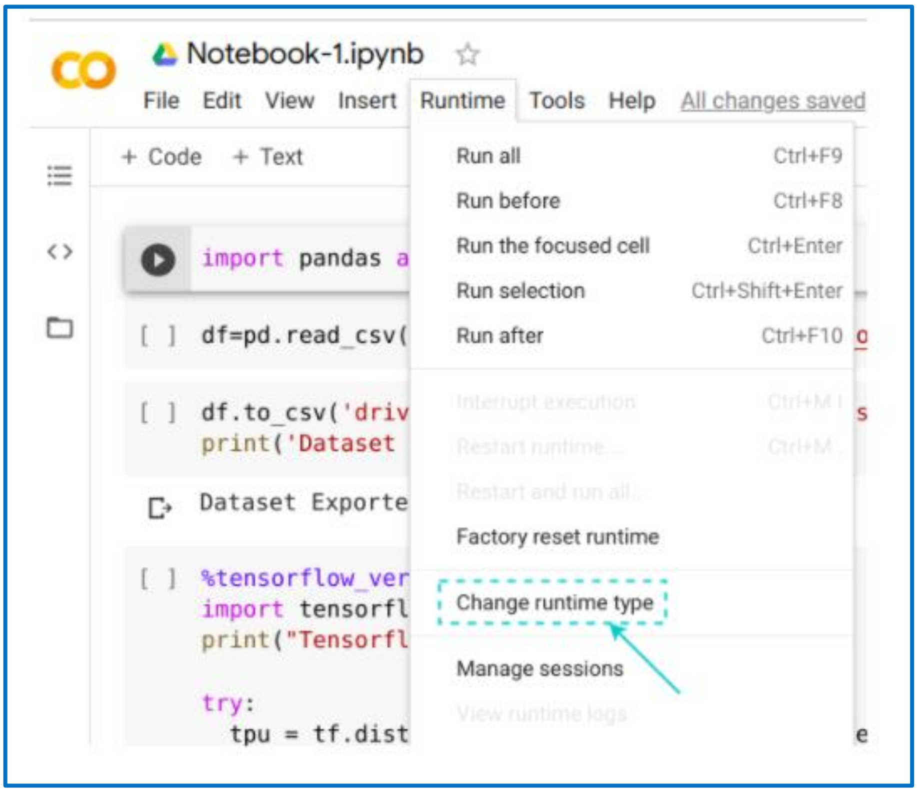


Figure 3: Change Runtime

5. Then select **GPU** and click Save.

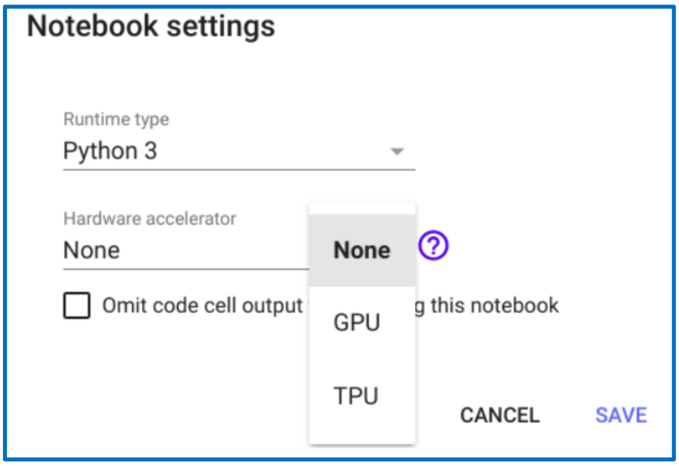


Figure 4: Runtime Settings

6. Next click on the dropdown of Connect button and select Connect to hosted runtime.

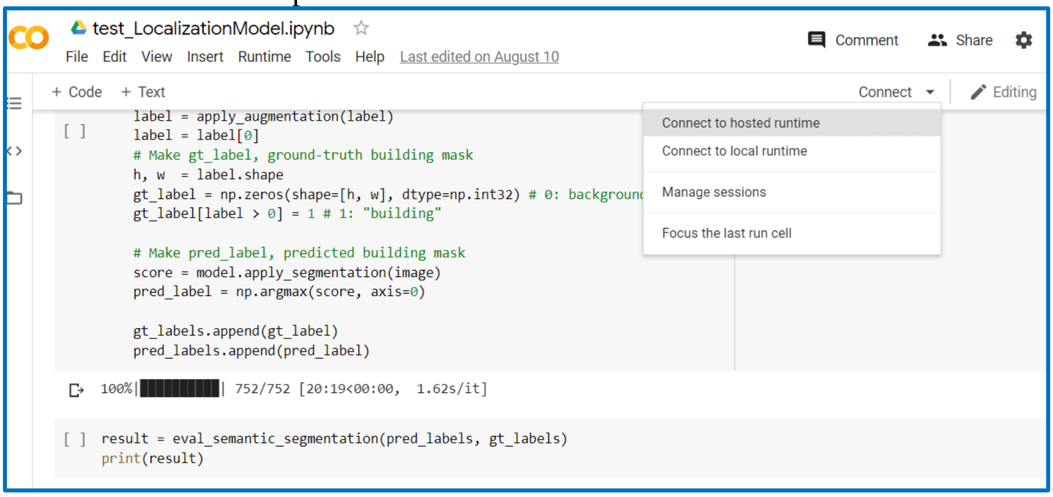


Figure 5: Connect to Google's Infrastructure

7. Then install all the libraries given in requirements.txt file using following command.

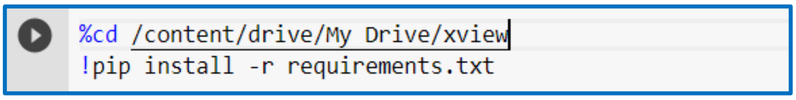


Figure 6: Command to install Libraries

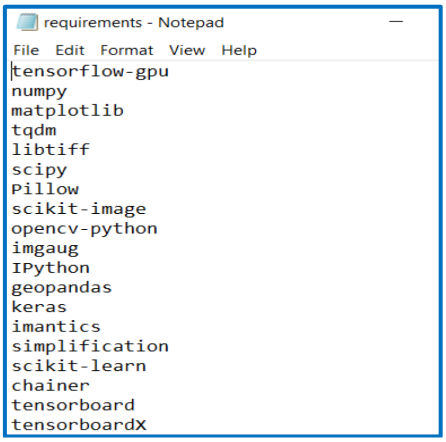
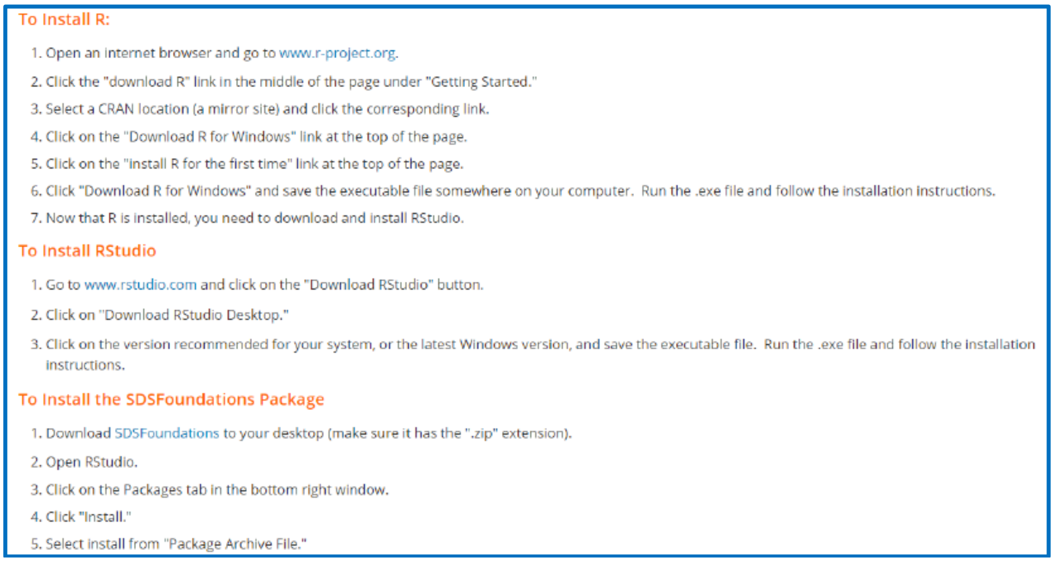


Figure 7: requirements.txt

3.2 R-Studio

For installation of R-Studio go to this link and press 'download R' button to start downloading. Figure 8 contains all the steps to download and install R-studio.


To Install R:

1. Open an internet browser and go to www.r-project.org.
2. Click the "download R" link in the middle of the page under "Getting Started."
3. Select a CRAN location (a mirror site) and click the corresponding link.
4. Click on the "Download R for Windows" link at the top of the page.
5. Click on the "Install R for the first time" link at the top of the page.
6. Click "Download R for Windows" and save the executable file somewhere on your computer. Run the .exe file and follow the installation instructions.
7. Now that R is installed, you need to download and install RStudio.

To Install RStudio

1. Go to www.rstudio.com and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.

To Install the SDSFoundations Package

1. Download SDSFoundations to your desktop (make sure it has the ".zip" extension).
2. Open RStudio.
3. Click on the Packages tab in the bottom right window.
4. Click "Install."
5. Select install from "Package Archive File."

Figure 8: R-Studio Installation Guide

4 Implementation

4.1 Data Source and Data storage

The satellite image data for the research project can be downloaded from the https://xview2.org/dataset repository. The size of the dataset is 14.42 GB and the collab notebook is connected to the hosted environment. So, upload the data on google drive and then mount the drive to the collab notebook. For mounting the google drive to your notebook follow the below given steps.

1. Run the following line of code in collab and click on the given link from output section.

```
from google.colab import drive
drive.mount('/content/drive')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4

Enter your authorization code:
```

Figure 9: Code to Mount Drive

2. In google consent page allow google file stream to access your drive files.

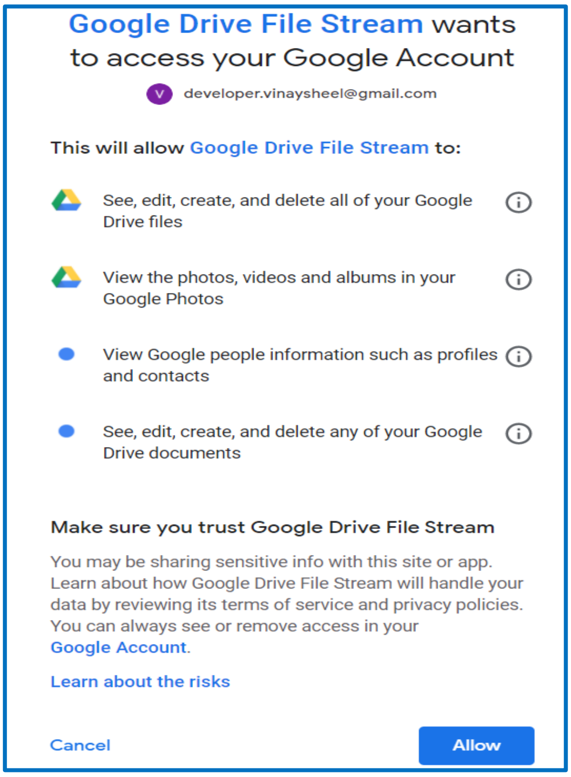


Figure 10: Consent Page

3. In next screen copy the given code as shown in Figure 11 and paste it in the textbox of Figure 7 and press Enter.

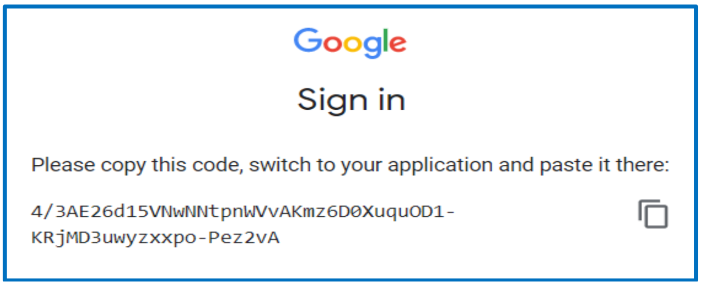


Figure 11: Authorization Code

4. Google Drive mounted successfully at /content/Drive

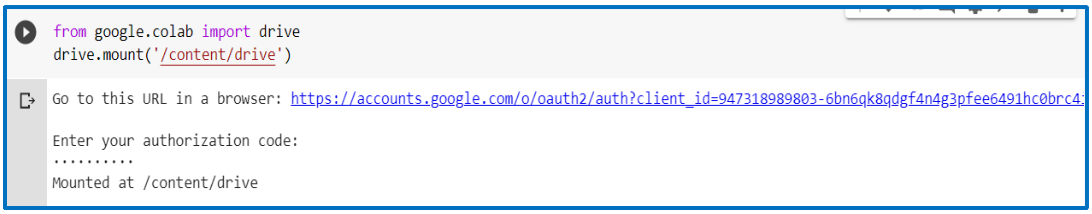


Figure 12: Drive Mounted Successfully

4.2 Import Library

First Import all the required libraries given in the Figure 13.

```python
import argparse
import numpy as np

import chainer
import chainer.functions as F
import chainer.links as L
from chainer import training
from chainer.training import extensions
from unet import UNet
from dataset import LabeledImageDataset
from tensorboardX import SummaryWriter
import os

from PIL import Image
import time
import numpy as np
import pandas as pd
from tqdm import tqdm
import os
import math
import random
import argparse
import logging
import json
import cv2
import datetime

from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix, accuracy_score
from sklearn.utils.class_weight import compute_class_weight
import shapely.wkt
import shapely
from shapely.geometry import Polygon
from collections import defaultdict

import tensorflow as tf
import keras
import ast
from keras import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Add, Input, Concatenate
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.resnet50 import ResNet50
from keras import backend as K
```

Figure 13: Libraries Used

4.3 Data Pre-Processing & Transformation

```python
for filename in files:
    disaster = filename.split("_")[0]
    # If the output directory and disater name do not exist make the directory
    if not path.isdir(path.join(output_dir, disaster)):
        makedirs(path.join(output_dir, disaster))
    # Check if the images directory exists
    if not path.isdir(path.join(output_dir, disaster, "images")):
        # If not create it
        makedirs(path.join(output_dir, disaster, "images"))
    # Move the pre and post image to the images directory under the disaster name
    cp(
        path.join(base_dir, "images", filename),
        path.join(output_dir, disaster, "images", filename),)
    post_file = filename.replace("_pre_", "_post_")
    cp(
        path.join(base_dir, "images", post_file),
        path.join(output_dir, disaster, "images", post_file),)
    # Check if the label directory exists
    if not path.isdir(path.join(output_dir, disaster, "labels")):
        # If not create it
        makedirs(path.join(output_dir, disaster, "labels"))
    pre_label_file = filename.replace("png", "json")
    # Move the pre and post label files to the labels directory under the disaster name
```

Figure 14: Split Dataset

The code in Figure 14 Split the dataset into multiple folders based on name of the disaster. Figure 15 shows the final folder structure of the dataset.

Figure 15: Dataset according to disaster

Next, Perform Data augmentation and normalization for segmentation model and classification model using the block of code from Figure 16.

```
def get_example(self, i):
    image_filename, label_filename = self._pairs[i]

    image_path = os.path.join(self._root, image_filename)
    image = _read_image_as_array(image_path, self._dtype)
    if self._distort:
        image = random_color_distort(image)
        image = np.asarray(image, dtype=self._dtype)

    image = (image - self._mean) / 255.0

    label_path = os.path.join(self._label_root, label_filename)
    label_image = _read_label_image_as_array(label_path, self._label_dtype)

    h, w, _ = image.shape

    label = np.zeros(shape=[h, w], dtype=np.int32) # 0: background
    label[label_image > 0] = 1 # 1: "building"

    # Padding
    if (h < self._crop_size) or (w < self._crop_size):
        H, W = max(h, self._crop_size), max(w, self._crop_size)

        pad_y1, pad_x1 = (H - h) // 2, (W - w) // 2
        pad_y2, pad_x2 = (H - h - pad_y1), (W - w - pad_x1)
        image = np.pad(image, ((pad_y1, pad_y2), (pad_x1, pad_x2), (0, 0)), 'symmetric')

        if self._test:
            # Pad with ignore_value for test set
            label = np.pad(label, ((pad_y1, pad_y2), (pad_x1, pad_x2)), 'constant', constant_values=255)
        else:
            # Pad with original label for train set
            label = np.pad(label, ((pad_y1, pad_y2), (pad_x1, pad_x2)), 'symmetric')
```

Figure 16: Image Centering & Padding

```python
# Randomly flip and crop the image/label for train-set
if not self._test:

    # Horizontal flip
    if random.randint(0, 1):
        image = image[:, ::-1, :]
        label = label[:, ::-1]

    # Vertical flip
    if random.randint(0, 1):
        image = image[::-1, :, :]
        label = label[::-1, :]

    # Random crop
    top  = random.randint(0, h - self._crop_size)
    left = random.randint(0, w - self._crop_size)

# Crop the center for test-set
else:
    top = (h - self._crop_size) // 2
    left = (w - self._crop_size) // 2

bottom = top + self._crop_size
right = left + self._crop_size

image = image[top:bottom, left:right]
label = label[top:bottom, left:right]

return image.transpose(2, 0, 1), label
```

Figure 17: Data Augmentation

Following figures of this section illustrates the code to create data-frame for evacuation path detection model. Figure 18 fetches post disaster json files and group them by disaster. In Figure 19 centre of the polygon is computed and extracted the damage dictionary.

```python
# Fetch all post disaster json files
labels_generator = Path('/content/drive/My Drive/xview/xBD/santa-rosa-wildfire/labels').rglob(pattern=f'*post_*.json')
#print(labels_generator)
# Group json files by disaster
def get_disaster_dict(labels_generator):
    disaster_dict = defaultdict(list)
    for label in labels_generator:
        disaster_type = label.name.split('_')[0]
        disaster_dict[disaster_type].append(str(label))
    return disaster_dict
disaster_dict = get_disaster_dict(labels_generator)
# Select a particular disaster
disaster_labels = disaster_dict['santa-rosa-wildfire']
```

Figure 18: Fetch Post Disaster Labels

```python
[ ]  # Get polygons center coords (lat, long)
     def get_centroid(coords):
         polygons = [ wkt.loads(polygon['wkt']) for polygon in coords ]
         #print(polygons)
         centroid =  MultiPolygon(polygons).centroid
         #print(centroid)
         try:
             return {'centroid_x': centroid.x, 'centroid_y': centroid.y, 'latlong': centroid }
         except IndexError as e:
             return {'centroid_x': None, 'centroid_y': None, 'latlong': None }

[ ]  def get_damage_dict(coords):
         damage_list = [ get_damage_type(coord['properties']) for coord in coords]
         #print(damage_list)
         return Counter(damage_list)
```

Figure 19: Get Centre of Polygons

In Figure 20, centroid of each image and damage dictionary is added to the dataframe. Then generate a pandas dataframe and sort it according to destroyed type.

```
[ ] # Add centroid and damage dict to metadata
    def metadata_with_damage(label_path):
        data = read_label(label_path)
        coords = data['features']['lng_lat']

        damage_dict = get_damage_dict(coords)
        #print(damage_dict)
        centroid = get_centroid(coords)

        data['metadata'].update(centroid)
        data['metadata']['path'] = label_path
        data['metadata'].update(damage_dict)
        #print(data)
        return data['metadata']

[ ] def generate_metadata_df(disaster_labels):
        metadata_list = [metadata_with_damage(label_path) for label_path in disaster_labels]
        df = pd.DataFrame(metadata_list)
        return df.fillna(df.mean())

[ ] # Sort df by destroyed count
    df = generate_metadata_df(disaster_labels)
    sorted_df = df.sort_values(by=['destroyed'], ascending=False)
```

Figure 20: Add Metadata and Create Data-Frame

Figure 21 shows the final sorted dataframe.

img_name	centroid_x	centroid_y	latlong	path	no-damage	major-damage	destroyed	minor-damage	un-classified
santa-rosa-wildfire_00000084_post_disaster.png	-122.749407	38.481419	POINT (-122.749406877816 38.48141941138705)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	11.00000	1.8	207.0	2.0	1.606061
santa-rosa-wildfire_00000063_post_disaster.png	-122.749256	38.473472	POINT (-122.7492557850193 38.47347175253174)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	29.00000	1.0	206.0	2.0	1.606061
santa-rosa-wildfire_00000155_post_disaster.png	-122.754197	38.477776	POINT (-122.7541966347944 38.47777603117169)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	84.00000	1.8	203.0	1.0	1.606061
santa-rosa-wildfire_00000161_post_disaster.png	-122.743818	38.477562	POINT (-122.7438182442444 38.47756246323021)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	74.00000	1.0	168.0	1.0	1.606061
santa-rosa-wildfire_00000079_post_disaster.png	-122.727716	38.473312	POINT (-122.7277160594859 38.47331205314139)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	63.00000	2.0	115.0	2.0	1.606061
...	...	...	...	...	...	...	...	...	...
santa-rosa-wildfire_00000283_post_disaster.png	-122.668091	38.515743	POINT (-122.6680905178813 38.51574269260035)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	3.00000	1.8	1.0	1.0	1.606061
santa-rosa-wildfire_00000300_post_disaster.png	-122.676082	38.533292	POINT (-122.6760821504748 38.53329239138814)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	53.67052	1.8	1.0	2.0	1.606061
santa-rosa-wildfire_00000298_post_disaster.png	-122.655532	38.498265	POINT (-122.6555315031532 38.49826462933488)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	6.00000	1.8	1.0	2.0	1.606061
santa-rosa-wildfire_00000274_post_disaster.png	-122.680180	38.505762	POINT (-122.6801798341798 38.50576238443799)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	53.67052	1.0	1.0	2.0	1.606061
santa-rosa-wildfire_00000245_post_disaster.png	-122.676057	38.545541	POINT (-122.6760573643569 38.5455411545861)	/content/drive/My Drive/xview/xBD/santa-rosa-w...	53.67052	1.8	1.0	2.0	1.606061

Figure 21: Snippet of dataframe

4.4 Modelling

This section contains code snippet of the proposed hybrid model which consists of segmentation model, classification model and evacuation path detection model.

4.4.1 Segmentation Model (U-Net)

Figure 22 contains code for implementation of the U-Net model. The code is divided into blocks due to space issues and all of the functions from the snippet belong to the UNET() class.

```python
def __init__(self, class_num=2, ignore_label=255):

    self.__class_num = class_num
    self.__ignore_label = ignore_label

    super(UNet, self).__init__(
        c0=L.Convolution2D(3, 32, 3, 1, 1),
        c1=L.Convolution2D(32, 64, 4, 2, 1),
        c2=L.Convolution2D(64, 64, 3, 1, 1),
        c3=L.Convolution2D(64, 128, 4, 2, 1),
        c4=L.Convolution2D(128, 128, 3, 1, 1),
        c5=L.Convolution2D(128, 256, 4, 2, 1),
        c6=L.Convolution2D(256, 256, 3, 1, 1),
        c7=L.Convolution2D(256, 512, 4, 2, 1),
        c8=L.Convolution2D(512, 512, 3, 1, 1),
        dc8=L.Deconvolution2D(1024, 512, 4, 2, 1),
        dc7=L.Convolution2D(512, 256, 3, 1, 1),
        dc6=L.Deconvolution2D(512, 256, 4, 2, 1),
        dc5=L.Convolution2D(256, 128, 3, 1, 1),
        dc4=L.Deconvolution2D(256, 128, 4, 2, 1),
        dc3=L.Convolution2D(128, 64, 3, 1, 1),
        dc2=L.Deconvolution2D(128, 64, 4, 2, 1),
        dc1=L.Convolution2D(64, 32, 3, 1, 1),
        dc0=L.Convolution2D(64, class_num, 3, 1, 1),

        bnc0=L.BatchNormalization(32),
        bnc1=L.BatchNormalization(64),
        bnc2=L.BatchNormalization(64),
        bnc3=L.BatchNormalization(128),
        bnc4=L.BatchNormalization(128),
        bnc5=L.BatchNormalization(256),
        bnc6=L.BatchNormalization(256),
        bnc7=L.BatchNormalization(512),
        bnc8=L.BatchNormalization(512),
        bnd8=L.BatchNormalization(512),
        bnd7=L.BatchNormalization(256),
        bnd6=L.BatchNormalization(256),
        bnd5=L.BatchNormalization(128),
        bnd4=L.BatchNormalization(128),
        bnd3=L.BatchNormalization(64),
        bnd2=L.BatchNormalization(64),
        bnd1=L.BatchNormalization(32) )
```

```python
def forward(self, x):

    e0 = F.relu(self.bnc0(self.c0(x)))
    e1 = F.relu(self.bnc1(self.c1(e0)))
    e2 = F.relu(self.bnc2(self.c2(e1)))
    del e1
    e3 = F.relu(self.bnc3(self.c3(e2)))
    e4 = F.relu(self.bnc4(self.c4(e3)))
    del e3
    e5 = F.relu(self.bnc5(self.c5(e4)))
    e6 = F.relu(self.bnc6(self.c6(e5)))
    del e5
    e7 = F.relu(self.bnc7(self.c7(e6)))
    e8 = F.relu(self.bnc8(self.c8(e7)))

    d8 = F.relu(self.bnd8(self.dc8(F.concat([e7, e8]))))
    del e7, e8
    d7 = F.relu(self.bnd7(self.dc7(d8)))
    del d8
    d6 = F.relu(self.bnd6(self.dc6(F.concat([e6, d7]))))
    del d7, e6
    d5 = F.relu(self.bnd5(self.dc5(d6)))
    del d6
    d4 = F.relu(self.bnd4(self.dc4(F.concat([e4, d5]))))
    del d5, e4
    d3 = F.relu(self.bnd3(self.dc3(d4)))
    del d4
    d2 = F.relu(self.bnd2(self.dc2(F.concat([e2, d3]))))
    del d3, e2
    d1 = F.relu(self.bnd1(self.dc1(d2)))
    del d2
    d0 = self.dc0(F.concat([e0, d1]))

    return d0
def __call__(self, x, t):

    h = self.forward(x)
    loss = F.softmax_cross_entropy(h, t, ignore_label=self.__ignore_label)
    accuracy = F.accuracy(h, t, ignore_label=self.__ignore_label)
    chainer.report({'loss': loss, 'accuracy': accuracy}, self)
    return loss
```

Figure 22: Implementation of U-Net Architecture

```python
[ ]  # Setup an optimizer
     optimizer = chainer.optimizers.Adam()
     optimizer.setup(model)
     # Load mean image
     mean = np.load(os.path.join(args.dataset, "mean.npy"))

     # Load the MNIST dataset
     train = LabeledImageDataset(os.path.join(args.dataset, "train.txt"), args.images, args.labels,
                                 mean=mean, crop_size=args.tcrop, test=False, distort=False)

     test = LabeledImageDataset (os.path.join(args.dataset, "val.txt"), args.images, args.labels,
                                 mean=mean, crop_size=args.vcrop, test=True, distort=False)

     train_iter = chainer.iterators.SerialIterator(train, args.batchsize)
     test_iter = chainer.iterators.SerialIterator(test, args.test_batchsize, repeat=False, shuffle=False)

     # Set up a trainer
     updater = training.StandardUpdater(
         train_iter, optimizer, device=args.gpu)
     trainer = training.Trainer(updater, (args.epoch, 'epoch'), out=log_dir)
```

Figure 23: Trainer Setup

4.4.2 Classification Model (ResNet50)

The classification model is implemented using transfer learning technique. First pretrained model is downloaded from the Keras library and integrated with the convolution layers as shown in Figure 24.

```
weights = 'imagenet'
inputs = Input(shape=(128, 128, 3))

base_model = ResNet50(include_top=False, weights=weights, input_shape=(128, 128, 3))

for layer in base_model.layers:
    layer.trainable = False

x = Conv2D(32, (5, 5), strides=(1, 1), padding='same', activation='relu', input_shape=(128, 128, 3))(inputs)
x = MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None)(x)

x = Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None)(x)

x = Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None)(x)

x = Flatten()(x)

base_resnet = base_model(inputs)
base_resnet = Flatten()(base_resnet)

concated_layers = Concatenate()([x, base_resnet])

concated_layers = Dense(2024, activation='relu')(concated_layers)
concated_layers = Dense(524, activation='relu')(concated_layers)
concated_layers = Dense(124, activation='relu')(concated_layers)
output = Dense(4, activation='relu')(concated_layers)
#output = concated_layers
model = Model(inputs=inputs, outputs=output)

#model.load_weights("/content/drive/My Drive/xview/classification.hdf5")
model.load_weights("/content/drive/My Drive/xview/classification_model/-saved-model-29-0.71.hdf5")
validation_gen = validation_generator("/content/sample_data/output/csv/train.csv", "/content/sample_data/output/process_images")
predictions = model.predict(validation_gen)

test_trues = validation_gen.classes
test_pred = np.argmax(predictions, axis=-1)

Found 54862 validated image filenames belonging to 4 classes.
```

Figure 24: Implementation of Classification Model

4.4.3 Evacuation Path Detection Model

Figure 25 and 26 contains code snippet of the evacuation path detection model.

```
def isInside(circle_x, circle_y, rad1, rad2, route):
    x = route[0]
    y = route[1]
    if (((x - circle_x)*111139) * ((x - circle_x)*111139) +
        ((y - circle_y)*111139) * ((y - circle_y)*111139) <= rad1 * rad1):
        return True;
    elif (((x - circle_x)*111139) * ((x - circle_x)*111139) +
          ((y - circle_y)*111139) * ((y - circle_y)*111139) <= rad2 * rad2):
          return True;
    else:
        return False;

def getSafeCoord(circle_x, circle_y, rad1, rad2, route):
    if rad1 >= rad2:
        print("radius1")
        radius = rad1
    else:
        print("radius2")
        radius = rad2
    x_coord = circle_x + (radius/111139)
    y_coord = circle_y + (radius/111139)
    return (x_coord,y_coord)
```

Figure 25: Code to determine safe co-ordinate

```python
#dangerRoute = []
def djkstra(route, df):
    safeRoute = []
    dangerRoute = []
    #lengthofroute = len(route)
    i = 0

    while i != (len(route)-1):
        for idx, row in df.iterrows():

            circle_x = row.loc['centroid_y']
            circle_y = row.loc['centroid_x']
            destroyedArea = row.loc['destroyed']
            damagedArea = row.loc['major-damage']

            # print(damagedArea)
            if isInside(circle_x, circle_y, destroyedArea, damagedArea, route[i]):
                if i == 0:
                    print("isInside() if-2")
                    safeRoute.append(route[i])
                    print("Inside disaster prone area--->finding route in safe area")
                    borderCoord = getSafeCoord(circle_x, circle_y,destroyedArea, damagedArea,route[i])
                    print(borderCoord)
                    dangerRoute.append(route[i])
                    alternateRoute = checkAlternateRoute(borderCoord)
                    djkstra(alternateRoute[0], df)
                    route =alternateRoute[0]
                    Limitation of the research----> mark the road in red color
                else:
                    print("isInside() else-2")
                    i -= 1
                    dangerRoute.append(route[i])
                    alternateRoute = checkAlternateRoute(route[i])
                    djkstra(alternateRoute[0], df)
                    route =alternateRoute[0]
            else:
                safeRoute.append(route[i])
                #i += 1
        i += 1
    return safeRoute, dangerRoute
```

Figure 26: Implementation of Dijkstra's

5 Evaluation of Proposed Model

Figure 27 and 28 and contains output of segmentation model. In which Figure 27 contains IOU and accuracy of building class and Figure 28 shows precision, recall and F1-score of the model

```python
result = eval_semantic_segmentation(pred_labels, gt_labels)
building_class = 1
print("IoU for class Building = ", result['iou'][building_class])
print("Accuracy for class Building = ", result['class_accuracy'][building_class])
```

```
IoU for class Building =  0.736839929000374
Accuracy for class Building =  0.8131766129245268
```

Figure 27: IOU & Accuracy of U-Net

```
[ ] confusion = calc_semantic_segmentation_confusion(pred_labels, gt_labels)
    print("confusion:")
    print(confusion)

    tn = confusion[0][0]
    fp = confusion[0][1]
    tp = confusion[1][1]
    fn = confusion[1][0]

    precision = float(tp) / float(tp + fp)
    recall       = float(tp) / float(tp + fn)
    test_accuracy = float(tp + tn) / float(tp + tn + fp + fn)
    f1_score = 2*((precision * recall) / (precision + recall))
    print()
    print("accuracy: ", test_accuracy)
    print("precision: ", precision)
    print("recall: ", recall)
    print("f1-score: ",f1_score)

    confusion:
    [[737137332   4824390]
     [  8699885  37867545]]

    accuracy:  0.9828487317612831
    precision:  0.8869952837696394
    recall:  0.8131766129245268
    f1-score:  0.8484834056348037
```

Figure 28: Performance Evaluation of U-Net

Figure 29 shows code and output of Wilcoxon rank sum test which is performed in RStudio.

```
> df <- read.table("C:/Users/Vinaysheel Wagh/Downloads/outputIouFile.csv",
+                  header = TRUE,
+                  sep = ",")
> wilcox.test(df[['Proposed_Model']], df[['Previous_Research']], paired = FALSE,alternative = "greater", mu =0.022, correct = FALSE)

	Wilcoxon rank sum test

data:  df[["Proposed_Model"]] and df[["Previous_Research"]]
W = 305559, p-value = 0.003383
alternative hypothesis: true location shift is greater than 0.022
```

Figure 29: Hypothesis Test Result

Figure 30 contains precision, recall, F1-score and confusion matrix of classification model.

```
print("Evaluation of ResNet50 architecture")
f1_weighted = f1_score(testTrues, testPred, average='weighted', labels=np.unique(testPred))
print("f1 score is: "+str(f1_weighted))

precision = precision_score(testTrues, testPred, average='weighted', labels=np.unique(testPred))
print("precision : "+str(precision))
recall =recall_score(testTrues, testPred, average='weighted')
print("recall : "+str(recall))
accuracy = accuracy_score(testTrues, testPred)
print("accuracy : "+str(accuracy))

print()
confusionMatrix = confusion_matrix(testTrues, testPred)
print(confusionMatrix)

Evaluation of ResNet50 architecture
f1 score is: 0.813174387268404
precision : 0.8332377586548305
recall : 0.7484050891327331
accuracy : 0.7484050891327331
```

Figure 30: Performance evaluation of Classification Model

6 Visualization

This section contains snippet of the outputs generated by the proposed hybrid model. The outcome of the model contains sectorize disaster affected area and evacuation path. Figure 31 shows code snippet for visualization of Sectorize disaster affected area generation and the sectorize disaster affected area as output of the visualization code.

```
# Go through each list and write it to the post image we just loaded
for polygon in no_damage_polygons:
    x,y = polygon.exterior.coords.xy
    coords = list(zip(x,y))
    draw.polygon(coords, damage_dict["no-damage"])

for polygon in minor_damage_polygons:
    x,y = polygon.exterior.coords.xy
    coords = list(zip(x,y))
    draw.polygon(coords, damage_dict["minor-damage"])

for polygon in major_damage_polygons:
    x,y = polygon.exterior.coords.xy
    coords = list(zip(x,y))
    draw.polygon(coords, damage_dict["major-damage"])

for polygon in destroyed_polygons:
    x,y = polygon.exterior.coords.xy
    coords = list(zip(x,y))
    draw.polygon(coords, damage_dict["destroyed"])

img.save(path_to_output)
display_img(path_to_output)
```

Figure 31: Sectorize Disaster Affected Area

Figure 32 illustrates the visualization of evacuation path detection model using folium library.

```
# Create a map with evacuation route
evacuationMap_case1 = folium.Map(location=location, tiles='openstreetmap', zoom_start=13)
decoded_route_coord = decodedRoute[1]

# Add points to the map
for idx, row in df.iterrows():
    generator = generate_circle(row)
    for circle in generator:
        circle.add_to(evacuationMap_case1)

folium.PolyLine(decodedRoute[1], color='darkgreen', weight=4.5, opacity=1).add_to(evacuationMap_case1)
folium.Marker(
        location=decoded_route_coord[63],
        tooltip='Rescue Shelter',
        icon=folium.Icon(color='green')
    ).add_to(evacuationMap_case1)

folium.features.RegularPolygonMarker(location = decoded_route_coord[0],
                                     tooltip = "Disatser affected area",
                                     number_of_sides = 8,
                                     radius = 10,
                                     weight = 1,
                                     color = "green",
                                     fill_opacity = 0.8,
                                     rotation = 30).add_to(evacuationMap_case1)
#folium.PolyLine(decodedRoute[1], color='red', weight=4.5, opacity=.5).add_to(evacuationMap_case1)
# Display the map
embed_map(evacuationMap_case1, 'evacuationMap_case1.html')
```

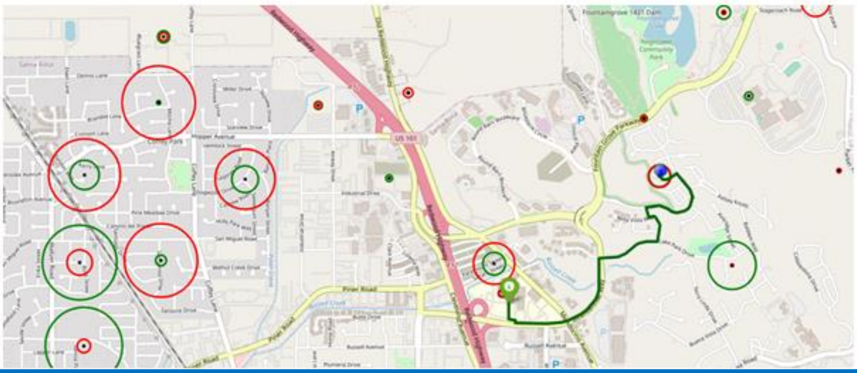


Figure 32: Visualization of Disaster Affected Areas