# Configuration Manual

MSc Research Project
Data Analytics


Ninad Mohite
Student ID: x18203591

School of Computing
National College of Ireland


Supervisor: Dr. Manaz Kaleel

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Ninad Mohite |
| **Student ID:** | x18203591 |
| **Programme:** | Data Analytics |
| **Year:** | 2020 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Dr. Manaz Kaleel |
| **Submission Due Date:** | 17/08/2020 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 723 |
| **Page Count:** | 8 |

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:** | |
| **Date:** | 15th August 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

Ninad Mohite
x18203591

## 1 Introduction

This configuration manual presents a step by step guide to implement Vertebra Segmentation from CT images using Volumetric Network which is presented in the report. The main aim of this manual is to assist in reproducing the results which are presented in the report. The project is developed using various technologies, high-end hardware and cloud systems.

### 1.1 Project Overview

The goal of the project is to implement vertebra segmentation using the volumetric network. Segmentation helps surgeons to detect problems related to the spine and vertebra at the early stage. The results obtained from the model can be used to develop a system that can be deployed to segment vertebra automatically.

## 2 Pre-requisites

The pre-requirements to execute the project are as follows. The hardware and software mentioned below were used during the implementation and are subject to change as per the availability. The GPU is a must for the training of the model.

### 2.1 Hardware Requirements

- Processor Required: 2.3 GHz Dual-Core Intel Core i5
- RAM: 8 GB 2133 MHz LPDDR3
- GPU: Nvidia Tesla K80 (Google Colab)
- Memory: Minimum 100GB on local system as well as on Google Drive
- Operating System: macOS Catalina

### 2.2 Software Requirements

- Programming Language: Python
- Development Tool and IDE: Jupyter Notebook, PyCharm, Google Colab
- Other Software: ITKSnap

# 3 Software Installation Guide

Following are the installation steps for the software mentioned in section 2.

## 3.1 Anaconda Navigator and PyCharm for anaconda

- Download Anaconda Nevigator graphical Installer
- To start the installation double click on the installer.
- Address the prompts on the screens for Introduction, Read Me and License.
- Click install button and start installation in /opt directory or install it at the preferred location and on the screen click on install for me.
- Select pycharm for anaconda and click continue.

Follow the installation guide for graphical assistance[1].

## 3.2 Installation of ITKSnap

ITK-SNAP is a software framework for segmenting medical image structures in 3D format. Following are the steps to install ITKSnap:

- Download the installer from the official website [2].
- Start the installation by double clicking on the icon.
- Accept License
- Drag the icon to applications folder or double click on it.
- Select the Applications folder and move the ITK-SNAP.app button onto the dock to attach the ITK-SNAP application to the Desktop.
- To launch program click on ITK-SNAP icon.

Follow the installation guide for graphical assistance [3].

ITK-SNAP is freely available software used to open .mhd files. It requires both .mhd and .raw files to open and image. To open a 3D scan, select open workspace and the browse to scan and select the scan. Detail information can be found in the official documentation [4]. Below snapshot in figure 1 shows the ITK-SNAP project work-space when a scan is loaded.

[1] https://docs.anaconda.com/anaconda/install/mac-os/
[2] http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.SNAP3
[3] http://www.itksnap.org/pmwiki/pmwiki.php?n=Documentation.TutorialSectionInstallation
[4] http://www.itksnap.org/pmwiki/pmwiki.php?n=Documentation.SNAP3

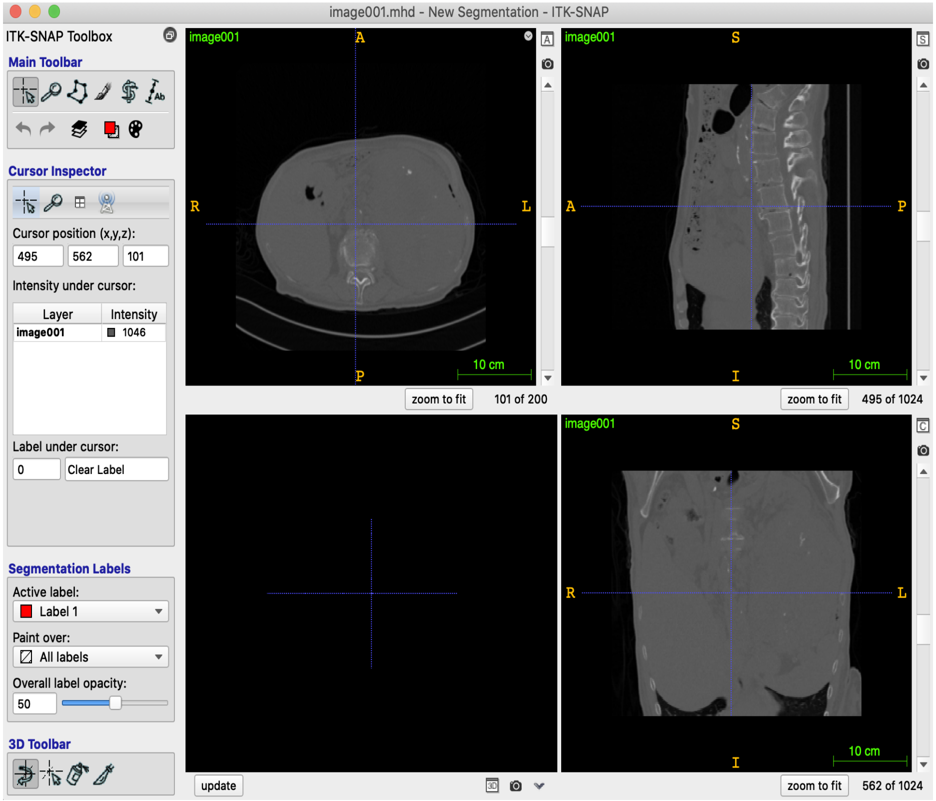


Figure 1: Screenshot of ITK-SNAP

# 4 Project Implementation Guide

This section talks about the implementation of the entire project. To install python packages use the command "pip install 'package_name' ".

## 4.1 Data Understanding and Preprocessing

### 4.1.1 Data Understanding and Generating Ground Truth

The most important part of any deep learning project is to understand the data you are working with. For exploratory data analysis and generation of masks (ground truth) run GenerateMask.ipynb. Run each cell in the notebook. Below figure shows the snapshot of the code.

### Import Packages

In [1]:
```
%matplotlib inline

import numpy as np
import pydicom
import os
import SimpleITK as sitk
import matplotlib.pyplot as plt
from glob import glob
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import scipy.ndimage
from skimage import morphology
from skimage import measure
from skimage.transform import resize
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly.tools import FigureFactory as FF
from plotly.graph_objs import *
init_notebook_mode(connected=True)
import seaborn as sns
```

In [2]:
```
data_path = '/Volumes/Samsung_T5/Vert/xVertSeg-1.v1/Data2/images/image025.mhd'
output_path = working_path = "/Users/ninadmohite/Desktop/Output/"
```

### Helper Functions

Here we make two helper functions.

- load_itk will load .mhd images from a folder.
- The voxel values in the images are raw. get_pixels_hu converts raw values into Houndsfeld units
- The transformation is linear. Therefore, so long as we have a slope and an intercept, we can rescale a voxel value to HU.

Both the rescale intercept and rescale slope are stored in the header at the time of image acquisition (these values are scanner-dependent, so you will need external information).

In [9]:
```
import SimpleITK as sitk
import numpy as np
'''
This funciton reads a '.mhd' file using SimpleITK and return the image array, origin and spacing of the image.
'''

def load_itk(filename):
    # Reads the image using SimpleITK
    itkimage = sitk.ReadImage(filename)

    # Convert the image to a  numpy array first and then shuffle the dimensions to get axis in the order z,y,x
    ct_scan = sitk.GetArrayFromImage(itkimage)

    # Read the origin of the ct_scan, will be used to convert the coordinates from world to voxel and vice versa.
    origin = np.array(list(reversed(itkimage.GetOrigin())))

    # Read the spacing along each dimension
    spacing = np.array(list(reversed(itkimage.GetSpacing())))

    return ct_scan, origin, spacing
```

Figure 2: Screenshot of GenerateMask.ipynb

### 4.1.2 Extract 2D slices from each 3D Scan

2D slices of each scans are saved as .bmp file, these are used to generate 3D patches of shape (128*128*128). To save slices from each scan run get2DScans.py

```python
from __future__ import print_function, division
import os
import SimpleITK as sitk
import cv2
import numpy as np


def getRangImageDepth(image):
    """
    :param image:
    :return:rang of image depth
    """
    # start, end = np.where(image)[0][[0, -1]]
    fistflag = True
    startposition = 0
    endposition = 0
    for z in range(image.shape[0]):
        notzeroflag = np.max(image[z])
        if notzeroflag and fistflag:
            startposition = z
            fistflag = False
        if notzeroflag:
            endposition = z
    return startposition, endposition


def resize_image_itk(itkimage, newSpacing, resamplemethod=sitk.sitkNearestNeighbor):
    """
    image resize withe sitk resampleImageFilter
    :param itkimage:
    :param newSpacing:such as [1,1,1]
    :param resamplemethod:
    :return:
    """
    newSpacing = np.array(newSpacing, float)
    originSpcaing = itkimage.GetSpacing()
    resampler = sitk.ResampleImageFilter()
    originSize = itkimage.GetSize()
    factor = newSpacing / originSpcaing
    newSize = originSize / factor
    newSize = newSize.astype(np.int)
    resampler.SetReferenceImage(itkimage)
    resampler.SetOutputSpacing(newSpacing.tolist())
    resampler.SetSize(newSize.tolist())
    resampler.SetTransform(sitk.Transform(3, sitk.sitkIdentity))
```

Figure 3: Screenshot of get2DScans.py

## 4.2 Generate 3D patches from the slices using

It is not possible to practically put entire 3D scan in a neural network for training. Hence, 3D patches are generated. To get 3D patches run prep_patches.py The 2D scans extracted using get2DScans.py is input to prep_patches.py.

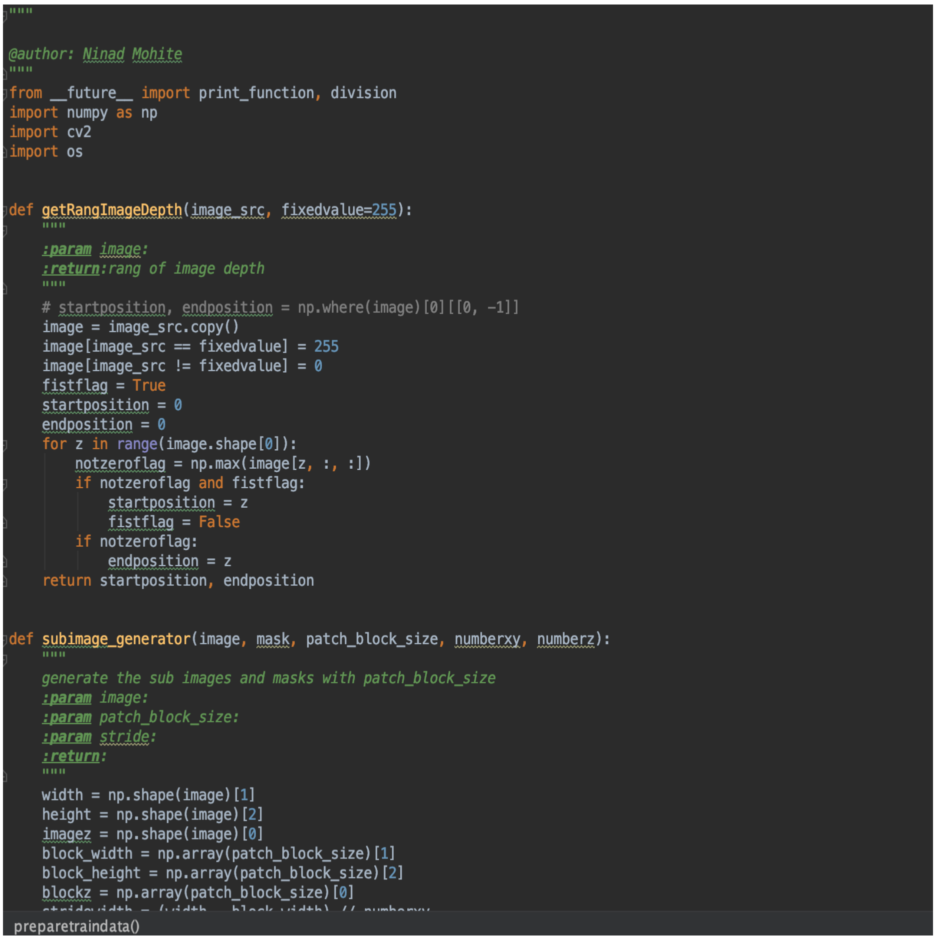


Figure 4: Screenshot of prep_patches.py

After generating patches run saveDetailsToCSV.py to save details of each patch in csv file. Upload the file and data in google drive a for training the V-Net model on google colab.

```python
import os

def file_name_path(file_dir, dir=True, file=True):
    """
    get root path,sub_dirs,all_sub_files
    :param file_dir:
    :return:
    """
    print(file_dir)

    for root, dirs, files in os.walk(file_dir):
        files.pop(0)
        print(root)
        print(dirs)
        if len(dirs) and dir:
            print("sub_dirs:", dirs)
            return dirs
        if len(files) and file:
            print("files:", len(files))
            return files


def save_file2csv(file_dir, file_name):
    """
    save file path to csv
    :param file_dir:preprocess data path
    :param file_name:output csv name
    :return:
    """
    out = open(file_name, 'w')
    image = "image"
    mask = "mask"
    file_image_dir = file_dir + "/" + image
    file_mask_dir = file_dir + "/" + mask
    file_paths = file_name_path(file_image_dir, dir=False, file=True)
    out.writelines("Image,Mask" + "\n")
    print(file_paths)
    for index in range(len(file_paths)):
        out_file_image_path = file_image_dir + "/" + file_paths[index]
        out_file_mask_path = file_mask_dir + "/" + file_paths[index]
        out.writelines(out_file_image_path + "," + out_file_mask_path + "\n")


save_file2csv("/Volumes/Samsung_T5/gen_image3d", "vertebra3dSegmentation.csv")
```

Figure 5: Screenshot of saveDetailsToCSV.py

## 4.3 Training the Model and Generating the results

Run VNet.py to train the model and generate a the results.

```python
import keras
import os
import h5py
from keras import backend as K
from keras.engine import Input, Model
from keras.layers import Conv3D, MaxPooling3D, UpSampling3D, Activation, BatchNormalization, PReLU, Conv3DTranspose
from keras.optimizers import Adam
from keras.layers.merge import concatenate
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from functools import partial
# Set the image shape to have the channels in the first dimension
K.set_image_data_format("channels_first")
```

```python
def dice_coefficient(y_true, y_pred, smooth=1.):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)


def dice_coefficient_loss(y_true, y_pred):
    return 1-dice_coefficient(y_true, y_pred)


def weighted_dice_coefficient(y_true, y_pred, axis= (0), smooth=1):
    """
    Weighted dice coefficient. Default axis assumes a "channels first" data structure
    :param smooth:
    :param y_true:
    :param y_pred:
    :param axis:
```

Figure 6: Screenshot of V-Net.py