National College of Ireland

Configuration Manual

MSc Research Project
MSCDAD SEP 23 B

Sandra Pereppadan Ignatious
Student ID: x21195463

School of Computing
National College of Ireland

Supervisor: Dr Ahmed Makki

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sandra Pereppadan Ignatious
Student ID:	X21195463
Programme:	MSCDAD SEP23 B **Year:** 2023
Module:	MSc. Research Project
Lecturer:	Dr Ahmed Makki
Submission Due Date:	14 August 2023
Project Title:	Liver Segmentation on CT Images for Tumor Detection Using Hybrid Modelling of U-Net, Inceptionv3 and ResNet18
Word Count:	402 **Page Count:** 5

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:	Sandra Pereppadan Ignatious
Date:	17 September 2023

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

Sandra Pereppadan Ignatious
Student ID: x21195463

1 Introduction

The study conducted on liver segmentation tumor detection using Hybrid modelling, configuration steps are discussed below in detail. Each section covers the details like software used and hardware details and the data collection details. This also describes the execution details of the code in detail. By following the below steps the code can executed in a right way.

2 System hardware and software details

The study has used some specific requirements in terms of software and hardware which has clearly explained below:

The Hardware details:

RAM space: 32GB
Hard Disk space: 1TB
Operating system: Windows 10
Processor: Intel i7, 12th Gen

The Software details:

Platform used (IDE): Kaggle Notebook
Language: Python 3.8
Database: Kaggle DB
Browser: Google Chrome

I have used Jupyter Notebook to save the code from Kaggle to an executable format.

3 Steps to Run Project

The first step of running project is to import the dataset to the Kaggle environment and the dataset has been taken from the IRCAD website. The Kaggle account was created and the Kaggle GPO was used to run the project as the dataset was pretty heavy to run this in local machine or google colab.

a) Data Collection and loading

The was taken from the IRCAD website and was loaded to the Kaggle environment and shows like below:

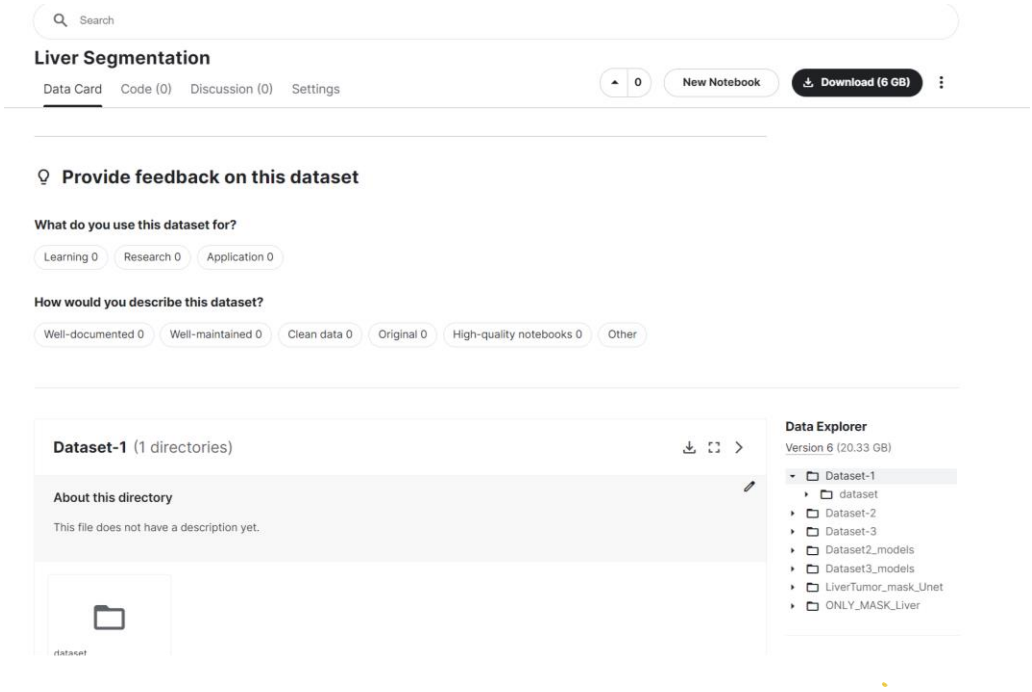


Fig 1: Data setting up in environment

b) Importing required libraries

The important libraries like NumPy, pandas, matplotlib etc are imported to the code for the later usage.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from sklearn import preprocessing
import random
import tensorflow as tf
import warnings
warnings.filterwarnings("ignore")
#!pip install visualkeras
import PIL
from PIL import Image
import numpy as np
import pandas as pd

import imageio
import imgaug as ia
import imgaug.augmenters as iaa
import numpy as np
import pandas as pd
```

Fig 2: Librairies importing

c) Loading the image and data into code

The image and the whole data were loaded in to the code and passed in to an array

Notebook Input Output Logs Comments (0) Settings

In [5]:

```
def hasmask(y):
    return not np.any(y)
def classify_data(x,y):
    Y = []
    X=[]
    for i in tqdm(range(y.shape[0])):
        if hasmask(y[i,:,:]) == False:
            X.append(x[i,:,:])
            Y.append(y[i,:,:])
    Y = np.array(Y)
    X = np.array(X)
    Y = Y.astype('float32')
    X = X.astype('float32')
    return X, Y
def load_df(org):
    print(org)
    DIR = '../input/liver-segmentation/Dataset-1/dataset/3Dircadb1/3Dircadb1/3Dircadb1.1/'
    df = pd.DataFrame()
    for i in os.listdir(DIR+'PATIENT_DICOM/PATIENT_DICOM/'):
        df=  df.append({
            'Image':DIR+'PATIENT_DICOM/PATIENT_DICOM/'+i,
            'Mask':DIR+'MASKS_DICOM/MASKS_DICOM/'+org+'/'+i,
            'Mask_of_liver':DIR+'MASKS_DICOM/MASKS_DICOM/liver/'+i
        },ignore_index=True)
    return df
df = load_df(organ)
```

```
livertumor01
```

Fig 3: data loading

d) Pre-processing steps

In this step the array was loaded and checked if it is correctly loaded

```
def ld_dimcon(path):
    dicom = pydicom.read_file(path)
    data = dicom.pixel_array
    return data
def load_dataset(df):
    X = []
    Y = []
    Y_liver = []
    for i in tqdm(df.index):
        X.append(ld_dimcon(df.iloc[i,0]))
        Y.append(ld_dimcon(df.iloc[i,1]))
        Y_liver.append(ld_dimcon(df.iloc[i,2]))
    X = np.array(X)
    Y =np.array(Y)
    Y_liver = np.array(Y_liver)
    print('With Empty masks',X.shape, Y.shape,Y_liver.shape)
    for i in range(5):
        ind = random.randint(0,X.shape[0] - 1)
        pltin(ind,X,Y,Y_liver)
    return X,Y,Y_liver,df
X,Y,Y_liver,df = load_dataset(load_df(organ))
```

```
livertumor01

100%|██████████| 129/129 [00:04<00:00, 31.21it/s]

With Empty masks (129, 512, 512) (129, 512, 512) (129, 512, 512)
```

Fig 4: Array initlisation

In [8]:
```
X.shape, Y.shape
```

Out[8]:
```
((129, 512, 512), (129, 512, 512))
```

In [9]:
```
X = X.reshape(X.shape[0],X.shape[1],X.shape[2],1)
Y = Y.reshape(Y.shape[0],Y.shape[1],Y.shape[2],1)
X = np.float32(X)
Y = np.float32(Y)
```

In [10]:
```
X.shape, Y.shape
```

Out[10]:
```
((129, 512, 512, 1), (129, 512, 512, 1))
```

Fig 5: Vectorisation

e) Modelling steps

The Hybrid model is being developed here, as the below screenshot showing the application of InceptionV3 and U-Net with the application of the optimiser as well. The models also applied in the same way with U-Net and later GWO and PSO applied for the selected model.

Notebook Input Output Logs Comments (0) Settings

```
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
```

In [12]:
```
import segmentation_models as sm
BACKBONE = 'inceptionv3'
preprocess_input = sm.get_preprocessing(BACKBONE)
dim = 512
w,h =dim ,dim
BATCH_SIZE = 8
CLASSES = ['Liver']
LR = 0.0001
EPOCHS = 40
preprocess_input = sm.get_preprocessing(BACKBONE)
# define network parameters
n_classes = 1 if len(CLASSES) == 1 else (len(CLASSES) + 1)  # case for binary and multiclass segmenta
tion
activation = 'sigmoid' if n_classes == 1 else 'softmax'
#create model
model = sm.Unet(BACKBONE,input_shape=(w,h, 1),encoder_weights=None, classes=n_classes, activation=ac
tivation)
```

```
Segmentation Models: using `keras` framework.
```

In [13]:
```
# define optomizer
optim = tf.keras.optimizers.Adam(LR)
# Segmentation models losses can be combined together by '+' and scaled by integer or float factor
dice_loss = sm.losses.DiceLoss()
focal_loss = sm.losses.BinaryFocalLoss() if n_classes == 1 else sm.losses.CategoricalFocalLoss()
```

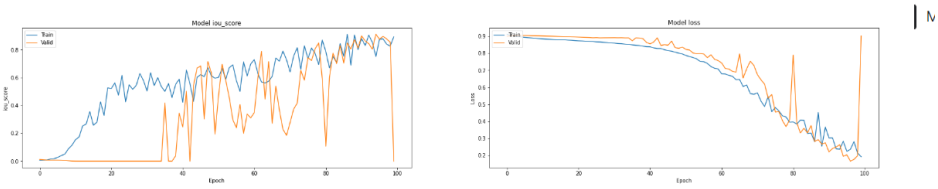
Fig 6: Model appliaction of InceptionV3 in U-Net

f) Training of the model and graphs

In [13]:

```
# define optomizer
optim = tf.keras.optimizers.Adam(LR)
# Segmentation models losses can be combined together by '+' and scaled by integer or float factor
dice_loss = sm.losses.DiceLoss()
focal_loss = sm.losses.BinaryFocalLoss() if n_classes == 1 else sm.losses.CategoricalFocalLoss()
total_loss = 0.9*dice_loss + (0.1 * focal_loss)
# actulally total_loss can be imported directly from library, above example just show you how to manip
ulate with losses
# total_loss = sm.losses.binary_focal_dice_loss # or sm.losses.categorical_focal_dice_loss
metrics = [sm.metrics.IOUScore(threshold=0.5), sm.metrics.FScore(threshold=0.5)]
# compile keras model with defined optimozer, loss and metrics
model.compile(optim, total_loss, metrics)
from sklearn.model_selection import train_test_split
X = np.float32(X)
X = np.resize(X,(X.shape[0],dim,dim,1))
Y_dash = np.float32(Y)
Y_dash = Y_dash/255
Y_dash = np.resize(Y_dash,(Y_dash.shape[0],dim,dim,1))
X_train, X_test, y_train, y_test = train_test_split(X, Y_dash, test_size=0.20, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.15, random_state
=42)
X_train.shape,X_valid.shape, X_test.shape, y_train.shape, y_valid.shape, y_test.shape
```

Out[13]:



In [18]:

```
from tensorflow.keras.models import model_from_json
model_json = model.to_json()
name = BACKBONE+'_D'
with open(name+".json", "w") as json_file:
    json_file.write(model_json)
model.save_weights(name+".h5")
import pandas as pd
hist_df = pd.DataFrame(history.history)
hist_df.to_csv(name+'.csv',index = False)
hist_df.tail()
```

Fig 7: Model training and graph

The model running is finished, and the later experiments are done based on the same manner to produce the outputs.