SKIN CANCER DETECTION USING TRANSFER LEARNING & ENSEMBLE MODELLING

Under the Guidance of

Dr. Agilandeeswari L

(Associate Professor, School of Information Technology and Engineering)

For the Course

ITE1015 – Soft Computing

For Winter Semester 2020-21

Project Team:

Aashish Bansal 19BIT0346

Perumalla Sasank 19BIT0338

Keerthi Yasasvi 19BIT0335

DECLARATION

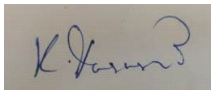
We, hereby, declare that the thesis entitled "Skin Cancer Detection Using Transfer Learning and Ensemble Modelling" submitted by us, for the award of the degree of *Bachelor of Technology in Information Technology* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Agilandeeswari L.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other Degree or Diploma in this Institute or any other Institute or University.

Place: Vellore

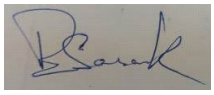
Date: 17th May 2021

15-05-2021

X 

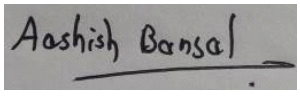
Keerthi Yasasvi 19BIT0335
Interface Designer and Developer
Signed by: 47316915-f1de-4b88-8c64-c2cc8414067c

15-05-2021

X 

Perumalla Sasank 19BIT0338
Interface Designer and Developer
Signed by: 47316915-f1de-4b88-8c64-c2cc8414067c

15-05-2021

X 

Aashish Bansal 19BIT0346
Learning Model Developer and Project Lead
Signed by: 47316915-f1de-4b88-8c64-c2cc8414067c

Signature of the Candidates

ACKNOWLEDGEMENTS

We would like to thank our mentor and the faculty in-charge for the course ITE1015 – Soft Computing, Dr. Agilandeeshwari L. for approving the project and being a mentor and guide for the project and helping us create, improve and bring out the best of our abilities to improve our skill and knowledge for learning, implementing and executing the project.

For the Project Team, we would to thank each other for helping each other in the hardships faced and sharing the joy of reaching a milestone and being a true support for each other at every step of the project. We had,

Aashish Bansal for Preparing and Training the Model and Documentation

Perumalla Sasank and Keerthi Yasasvi for Interface Development, Design and Documentation

EXECUTIVE SUMMARY

Vellore Institute of Technology, Vellore

Faculty Name: Agilandeeshwari L.

10th May, 2021

EXECUTIVE SUMMARY: SKIN CANCER

Skin Cancer is categorized into two categories namely: malignant melanoma(lethal) and benign(non-lethal/treatable). Evolution in technology has brought upon many boon's as well as banes and Cancer is one of them, skin cancer becoming a common side effect due to exposure to a lethal level of radiation or due to genetic disorder from exposure to radiation. The opportunity for machine learning models to be implemented, recommendations and costs are mentioned in this article.

OPPORTUNITY

With boons of advancement in technology machine learning models can be implemented and trained to detect, diagnose and possibly prevent any benign case from turning into a malignant case thus, advancing the medical infrastructure. Currently the main challenges are the cost-effective models to train, economic viability and error detection rates varying from ethnicity/race and a few specific cases.

SOLUTION/RECOMMENDATION

Training of a model with Mobilenet, Inception and Xception Architecture gives an economically viable solution and a detection rate of above 60% which as per the IEEE standards is viable for use practically. Recurring training of various Skin Cancer training data based on specific individuals with previously known other skin disorders and others on their race/ethnicity as to increase the detection rate without overfitting the model.

Table of Contents

TABLE OF FIGURES

TABLE OF TABULAR INFORMATION

ABBREVIATIONS

- CNN – Convolution Neural network
- GUI – Graphics User interface
- UV light – Ultra Violet light
- OS – Operating System
- ResNet – Residential Energy Services Network
- CSV – Comma Separated Values
- ROC – Receiver Operating Circuit
- EDA – Exploratory Data Analysis
- MCAR - Missing Completely At Random
- MAR - Missing At Random
- MNAR - Missing Not at Random
- ISIC – international Standard industrial classification
- SIIM – Society for Imaging Informatics in Medicine
- U.S. – United States of America
- SPF – Sun Protection Factors
- Exe File – Executable File
- VGG - Visual Geometry Group
- ROC - Receiving Operating Characteristic Curve
- EDA – Exploratory Data Analysis
- np – Numpy
- 4D – Four Dimensional
- ML – Machine Learning
- H5 - Hierarchical Data Format
- TFLITE – Tensorflow Lite
- API – Application Programming Interface

SYMBOLS AND NOTATIONS

Symbol	Meaning
-	Subtraction
+	Addition
/	Division
()/x	Multiplication (multiplication of enclosed value and value outside the enclosed bracket)

SKIN CANCER DETECTION USING TRANSFER LEARNING AND ENSEMBLE MODELING

PROPOSED ARCHITECTURE

ABSTRACT

The project is a Transfer Learning and Ensemble Learning trained model which consists of the MobileNet, Inception and Xception Learning pre-trained Models which can predict whether the patient has a suffering from Skin Cancer or not by checking the scanning and predicting on the images of the infected areas on the body. The model has been trained on a variety of images through which it predicts the required.

In this project, the image file of the patient is upload into a general public use software, which is a GUI-based interface, developed with the help of Tkinter and Python, and it consists of the model saved as a file and the software passes the image through the function and uses the results obtained that to analyse the image and give the prediction which can help people and doctors to start with the medication way earlier instead of waiting for the laboratory tests and reports for the confirmation, which are also very expensive.

We know about Skin Cancer that Skin cancer is an abnormal growth of skin cells. Most skin cancers are caused by exposure to ultraviolet (UV) light. When the skin is not protected, UV rays from sunlight or tanning beds can damage and alter skin's DNA that leads to the cancer.

Now, Deep learning model has been built to classify and identify the binary diagnostic group of melanocytic images obtained through dermoscopy. And based on the model, disease detection through dermal cell images has been investigated, and classifications on dermal cell images have been performed.

INTRODUCTION

OBJECTIVE

In this project, we plan on taking the image of the skin disease of a patient and upload and pass it through the machine learning model which we have created and and based on the analysis of the image by the model, we plan on predicting whether the person is suffering from Skin Cancer or not.

We intent to make a free software in the interest of the benefit of the general public and can be used by anyone and everyone and they can use it to check for the possibility of Skin Cancer with the ease of being at there home by uploading a clear image of the infected area.

The image file of the patient is upload into a software, which is GUI-based interface, developed with the help of Tkinter, and it consists of the model saved as a file and the software uses that to analyse the image and give the prediction which can help doctors to start with the medication way faster instead of waiting for the laboratory reports for the confirmation.

MOTIVATION

Skin cancer is an alarming disease for mankind. The necessity of early diagnosis of the skin cancer have been increased because of the rapid growth rate of Melanoma skin cancer, its high treatment costs, and death rate. This cancer cells are detected manually and it takes time to cure in most of the cases. We even intend to help people in the pandemic time so that people do not have to risk there lives on getting the laboratory tests.

The features of the affected skin cells are extracted after the segmentation of the dermoscopic images using feature extraction technique.

We want to build an application which we can deploy in the production for the general public and can be used by anyone and everyone for their benefit to start early with the medication and precautions.

BACKGROUND

Skin cancer is an abnormal growth of skin cells. Most skin cancers are caused by exposure to ultraviolet (UV) light. When the skin is not protected, UV rays from sunlight or tanning beds can damage and alter skin's DNA that leads to the cancer.

Deep learning model has been built to classify and identify the binary diagnostic group of melanocytic images obtained through dermoscopy. Based on the model, disease detection through dermal cell images has been investigated, and classifications on dermal cell images have been performed.

ABOUT THE DISEASE

Skin cancer is the most prevalent type of cancer. Melanoma, specifically, is responsible for 75% of skin cancer deaths, despite being the least common skin cancer. The American Cancer Society estimates over 100,000 new melanoma cases will be diagnosed in 2020. It's also expected that almost 7,000 people will die from the disease. As with other cancers, early and accurate detection—potentially aided by data science—can make treatment more effective.

Currently, dermatologists evaluate every one of a patient's moles to identify outlier lesions or “ugly ducklings” that are most likely to be melanoma. Existing AI approaches have not adequately considered this clinical frame of reference. Dermatologists could enhance their diagnostic accuracy if detection algorithms take into account “contextual” images within the same patient to determine which images represent a melanoma. If successful, classifiers would be more accurate and could better support dermatological clinic work.

As the leading healthcare organization for informatics in medical imaging, the Society for Imaging Informatics in Medicine (SIIM)'s mission is to advance medical imaging informatics through education, research, and innovation in a multi-disciplinary community. SIIM is joined by the International Skin Imaging Collaboration (ISIC), an international effort to improve melanoma diagnosis. The ISIC Archive contains the largest publicly available collection of quality-controlled dermoscopic images of skin lesions.

In this competition, you’ll identify melanoma in images of skin lesions. In particular, you’ll use images within the same patient and determine which are likely to represent a melanoma. Using patient-level contextual information may help the development of image analysis tools, which could better support clinical dermatologists.

Melanoma is a deadly disease, but if caught early, most melanomas can be cured with minor surgery. Image analysis tools that automate the diagnosis of melanoma will improve dermatologists' diagnostic accuracy. Better detection of melanoma has the opportunity to positively impact millions of people.

FACTS ON MELANOMA SKIN CANCER

- It's estimated that the number of new melanoma cases diagnosed in 2021 will increase by 5.8 percent.
- The number of melanoma deaths is expected to increase by 4.8 percent in 2021.
- An estimated 207,390 cases of melanoma will be diagnosed in the U.S. in 2021. Of those, 106,110 cases will be in situ (non-invasive), confined to the epidermis (the top layer of skin), and 101,280 cases will be invasive, penetrating the epidermis into the skin's second layer (the dermis). Of the invasive cases, 62,260 will be men and 43,850 will be women.
- In the past decade (2011 – 2021), the number of new invasive melanoma cases diagnosed annually increased by 44 percent
- An estimated 7,180 people will die of melanoma in 2021. Of those, 4,600 will be men and 2,580 will be women.
- The vast majority of melanomas are caused by the sun. In fact, one UK study found that about 86 percent of melanomas can be attributed to exposure to ultraviolet (UV) radiation from the sun.
- Compared with stage I melanoma patients treated within 30 days of being biopsied, those treated 30 to 59 days after biopsy have a 5 percent higher risk of dying from the disease, and those treated more than 119 days after biopsy have a 41 percent higher risk.
- Across all stages of melanoma, the average five-year survival rate in the U.S. is 93 percent. The estimated five-year survival rate for patients whose melanoma is detected early is about 99 percent. The survival rate falls to 66 percent when the disease reaches the lymph nodes and 27 percent when the disease metastasizes to distant organs.
- Only 20 to 30 percent of melanomas are found in existing moles, while 70 to 80 percent arise on apparently normal skin.
- On average, a person's risk for melanoma doubles if they have had more than five sunburns, but just one blistering sunburn in childhood or adolescence more than doubles a person's chances of developing melanoma later in life.
- Regular daily use of an SPF 15 or higher sunscreen reduces the risk of developing melanoma by 50 percent.
- Melanoma accounts for 6 percent of new cancer cases in men, and 5 percent of new cancer cases in women.
- Men age 49 and under have a higher probability of developing melanoma than any other cancer.
- From ages 15 to 39, men are 55 percent more likely to die of melanoma than women in the same age group.
- Women age 49 and under are more likely to develop melanoma than any other cancer except breast and thyroid cancers.
- From age 50 on, significantly more men develop melanoma than women. The majority of people who develop melanoma are white men over age 55. But until age 49, significantly more non-Hispanic white women develop melanoma than white men (one in 156 women versus one in 230 men). Overall, one in 27 white men and one in 40 white women will develop melanoma in their lifetime.

STATISTICS ON SKIN CANCER

SKIN CANCER RATES: BOTH GENDERS

Australia had the highest rate of melanoma in 2018, followed by New Zealand.

Table 1: Skin Cancer Rates: Both Gender

Rank	Country	Age-standardised rate per 100,000
1	Australia	33.6
2	New Zealand	33.3
3	Norway	29.6
4	Denmark	27.6
5	Netherlands	25.7
6	Sweden	24.7
7	Germany	21.6
8	Switzerland	21.3
9	Belgium	19.9
10	Slovenia	18.6
11	Luxembourg	16.5
12	Ireland	16.3
13	Finland	15.8
14	UK	15.0
15=	Austria	13.6
15=	France (metropolitan)	13.6
17	US	12.7

18	Czech Republic	12.6
19=	Canada	12.4
19=	Italy	12.4

SKIN CANCER RATES IN MEN

Australia had the highest rate of melanoma in men in 2018, followed by New Zealand.

Table 2: Skin Cancer Rates in Men

Rank	Country	Age-standardised rate per 100,000
1	Australia	40.4
2	**New Zealand**	**35.8**
3	Norway	29.0
4	**Netherlands**	**26.4**
5	Sweden	23.5
6	**Switzerland**	**23.4**
7	Denmark	22.4
8	**Germany**	**19.6**
9	Luxembourg	18.1
10	**Slovenia**	**18.0**
11=	Belgium	16.2
11=	**Finland**	**16.2**
13=	Austria	15.0
13=	**UK**	**15.0**
15	US	14.9

16	**France (metropolitan)**	**14.4**
17	Italy	14.0
18	**Ireland**	**13.6**
19	Canada	13.4
20	**Czech Republic**	**13.3**

SKIN CANCER RATES IN WOMEN

Denmark had the highest rate of melanoma in women in 2018, followed by New Zealand.

Table 3: Skin Cancer Rates in Women

Rank	Country	Age-standardised rate per 100,000
1	Denmark	33.1
2	New Zealand	31.1
3	Norway	30.7
4	Australia	27.5
5	Sweden	26.2
6	Netherlands	25.4
7	Germany	24.0
8	Belgium	23.9
9	Slovenia	19.7
10	Switzerland	19.5
11	Ireland	19.0
12	Finland	15.9
13	Luxembourg	15.4

14	UK	15.3
15	France (metropolitan)	12.9
16	Austria	12.6
17	Czech Republic	12.4
18=	Canada	11.7
18=	Iceland	11.7
20	Estonia	11.4
21=	Italy	11.0
21=	US	11.0
23	Greece	10.3
24	Hungary	10.1
25	Lithuania	9.0

PROJECT DESCRIPTION AND GOALS

As mentioned before, in this project, the image file of the patient is upload into a software, which is GUI-based interface, developed with the help of Tkinter, and it consists of the model saved as a file and the software uses that to analyze the image and give the prediction which can help doctors to start with the medication way faster instead of waiting for the laboratory reports for the confirmation.

The learning model is created with the help of the following pre-trained architectures: MobileNet, Inception and Xception.

The MobileNet Architecture is based on a streamlined architecture that uses a depth-wise separable convolution to build a light-weight deep neural network. We introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem.

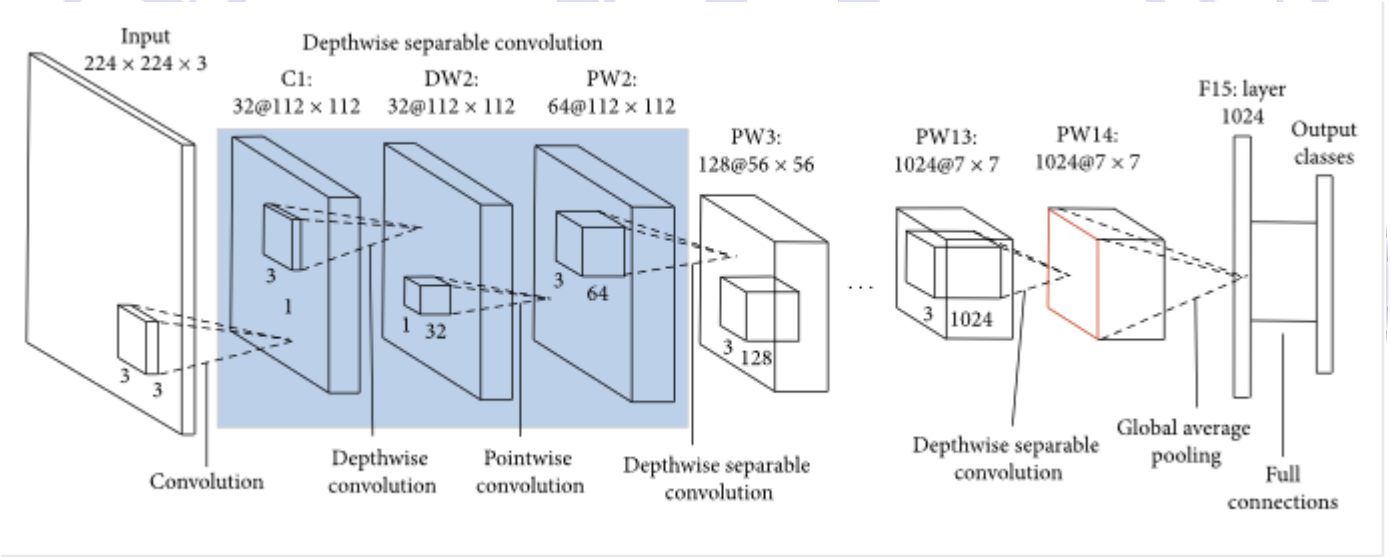


Figure 1: MobileNet Architeure (Source: Analytics Vidya)

The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

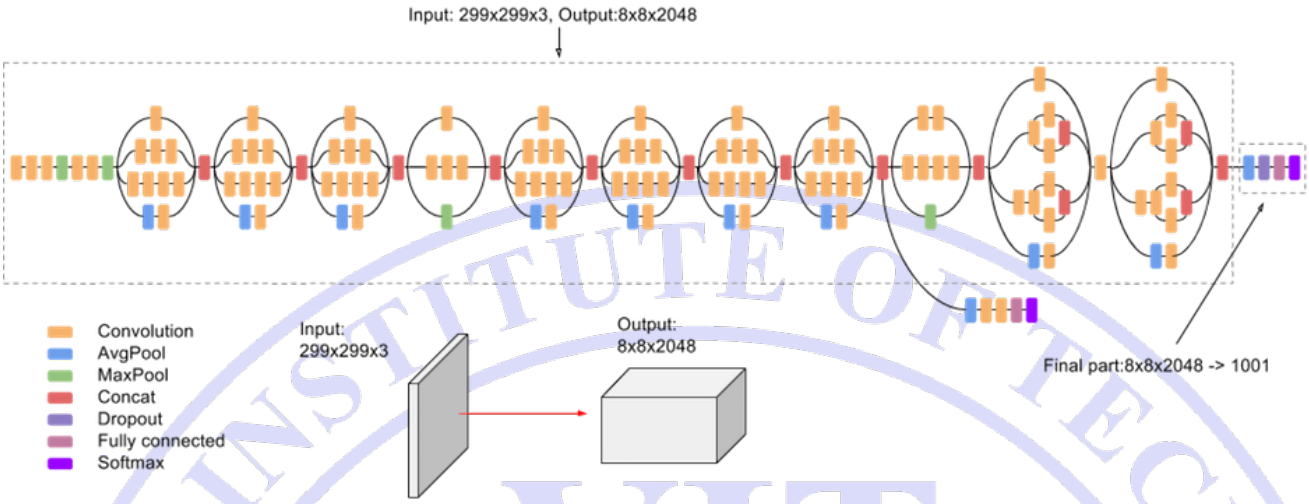


Figure 2: InceptionV3 Architecture (Source: Google Cloud)

Xception stands for "extreme inception." Rather like our previous two architectures, it reframes the way we look at neural nets — conv nets in particular. And, as the name suggests, it takes the principles of Inception to an extreme. Here's the hypothesis: "*cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly*."

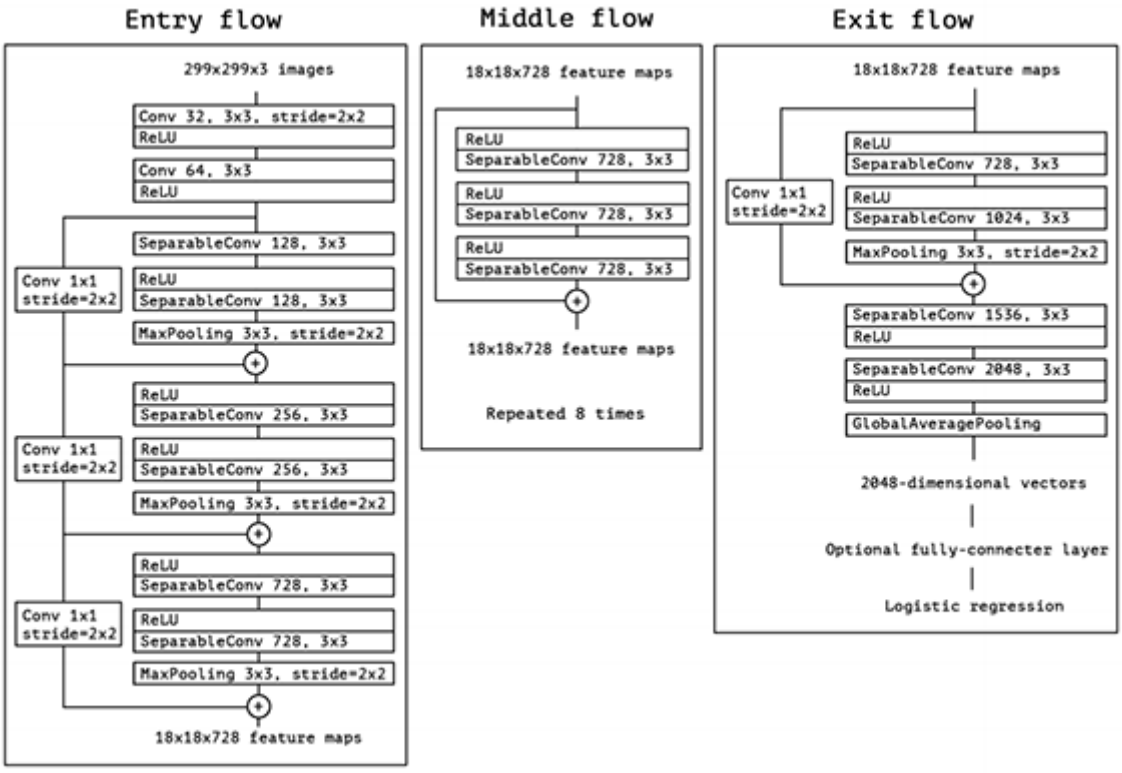


Figure 3: Xception Architecture (Source: pyimagesearch.com)

The goal of this project is in the favour of the help of the general public as anyone and everyone can use this to check personally so that the medications and precautions can be started early.

TECHNICAL SPECIFICATIONS

MODEL DEVELOPMENT AND TRAINING

Development of the Machine Learning/Deep Learning Model requires the knowledge and skill of using Jupyter Notebook (either installed via Anaconda Studio or directly as an individual component) or Google Colab along with high-powered Graphical Processing Unit for better, reduced and efficient processing time.

Use of Google Colab gives more advantages for writing and execution of code in Python, sharing notebook, Integrating Libraries and Dependencies and also to increase the available processing power through Cloud Computing. The advantage of Cloud Computing allows the model to still train even if the system crashes due to some unforeseen reason and does not cause a halt to the training.

INTERFACE DEVELOPMENT AND TESTING

The interface is converted into an executable file (exe File) so that the general public regardless of with or without any technical skills can simply install the software on their system and use the model to check if the patient suffering from the disease has Skin Cancer or not, so that the patient can think and take action at an early stage and start with the precautions and medications which can help save his/her life.

After uploading an image of the infected area, the final result is that whether the person is suffering from skin cancer or not. The final result is shown through an alert box.

INTERFACE REQUIREMENTS

There are two interfaces namely – User Interface and Software Interface.

- User Interface: The user interface will be implemented using any desktop running on Windows OS. This interface will be very user friendly so that people from different strata can use it to detect their disease without any difficulty by just uploading their medical test image.
- Software Interface: A software interface running on Windows OS. It should have Python compiler.

PLATFORM, CONFIGURATION AND SPECIFICATION

Platform: Google Colab (Pro version recommended)

Standard RAM: 12.69 GB

High-RAM Configuration: 25.46 GB (recommended with Pro version)

Normal Disk Space: 89.15 GB

Pro Version Disk Space: 147.15 GB

Engine: Google 3 Python Compute Engine backed (GPU)

GPU: NVIDIA Tesla K80

DESIGN APPROACH AND DETAILS

DESIGN APPROACH/MATERIALS AND METHODS

As suggested by the mentor to use the pre-trained models, for example, VGG, ResNet, MobileNet, etc., we planned on using these models and building an architecture. The proposed model includes the use of MobileNet, Inception and Xception model which are very well-known pre-trained architectures. These architectures have been brought together using Ensemble Modelling in which we have created a list of these models through which we join the Neural Net and further join their weights for one single model and to used and updated.

CODES AND STANDARDS

For the implementation of this model, the code was written using the programming language Python. Python also has several advantages due to its hundreds of open-source libraries which are available for almost every functionality to improve and increase the performance. The program for interface is also written using Python.

PROJECT SOURCE CODE

SOURCE CODE IN THE FORM OF A PY FILE

-*- coding: utf-8 -*-

"""WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

Automatically generated by Colaboratory.

Original file is located at: https://colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki

CODE:

```
# <center><b>Skin Cancer Classification for Detecting Melanoma <br> Using <br> Transfer Learning and Ensemble Modeling</b></center>

<center>

<h2>J-Component - Soft Computing</h2>

<p>Made under the guidance of <b>Dr. Agilandeeswari L</b><br>

</center>

By:<br>

Aashish Bansal 19BIT0346<br>

Keerthi Yasasvi 19BIT03<br>

Perumalla Sasank 19BIT03<br>

</p>
```

The project is a Transfer Learning and CNN trained model which can predict whether the patient has a suffering from Cancer or not by checking the images of the infected areas on the body. The model has been trained on a variety of images through which it predicts the required.

In this project, the image file of the patient is upload into a software, which is GUI-based interface, developed with the help of Tkinter, and it consists of the model saved as a file and the software uses that to analyze the image and give the prediction which can help doctors to start with the medication way faster instead of waiting for the laboratory reports for the confirmation.

So basically,

* Skin cancer is an abnormal growth of skin cells. Most skin cancers are caused by exposure to ultraviolet (UV) light. When the skin is not protected, UV rays from sunlight or tanning beds can damage and alter skin's DNA that leads to the cancer.

* Deep learning model has been built to classify and identify the binary diagnostic group of melanocytic images obtained through dermoscopy.

* Based on the model, disease detection through dermal cell images has been investigated, and classifications on dermal cell images have been performed.

Ensembles are predictive models that combine predictions from two or more other models. Ensemble learning methods are popular and the goto technique when the best performance on a predictive modeling project is the most important outcome. Nevertheless, they are not always the most appropriate technique to use and beginners the field of applied machine learning have the expectation that ensembles or a specific ensemble method are always the best method to use. Ensembles offer two specific benefits on a predictive modeling project, and it is important to know what these benefits are and how to measure them to ensure that using an ensemble is the right decision on your project. In this tutorial, you will discover the benefits of using ensemble methods for machine learning. After reading this tutorial, you will know:

* A minimum benefit of using ensembles is to reduce the spread in the average skill of a predictive model.

* A key benefit of using ensembles is to improve the average prediction performance over any contributing member in the ensemble.

* The mechanism for improved performance with ensembles is often the reduction in the variance component of prediction errors made by the contributing models.

1. Preprocessing

1.1. Loading Libraries

```
#### 1.1.1. Importing Libraries

"""


import numpy as np

import pandas as pd

import os

import cv2

import csv


import matplotlib.pyplot as plt

# import matplotlib.pylab as plt


from sklearn.datasets import load_files


import keras

from keras.preprocessing import image

from keras.preprocessing.image import img_to_array

from keras.preprocessing.image import load_img

from keras.utils import np_utils

from keras.models import Model

from keras.layers import Dense, GlobalAveragePooling2D, Flatten, BatchNormalization,
Activation, Dropout

from keras.callbacks import ModelCheckpoint, TensorBoard


from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

from tensorflow.python.keras.layers import
Dense,Conv2D,Flatten,MaxPooling2D,GlobalAveragePooling2D,Activation,BatchNormalization,
Dropout

from tensorflow.python.keras import Sequential,backend,optimizers
```

```
from numpy import *


from tkinter import *


from PIL import ImageTk

from PIL import ImageFile

from PIL import Image


from tqdm import tqdm


from glob import glob


ImageFile.LOAD_TRUNCATED_IMAGES = True


"""### 1.2. Loading Data


#### 1.2.1. Mounting Google Drive
"""


from google.colab import drive

drive.mount('/content/drive')


"""#### 1.2.2. Training Data


##### 1.2.2.1. Importing Training Data
"""


# Load text files with categories as subfolder names.

path_training_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/ISIC-2017_Training_Part3_GroundTruth.csv"

training_data = pd.read_csv(path_training_data)
```

```
"""##### 1.2.2.2 Display the Training Data of the CSV File"""


training_data


print("Filename: \n", training_data['image_id'][:5])


print("Targets: \n", training_data['melanoma'][:5])


"""##### 1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding"""


# Getting the labels

target = np_utils.to_categorical(np.array(training_data['melanoma']), 2)

target


"""##### 1.2.2.4 Checking the Number of Training Images"""


len(training_data['image_id'])


"""##### 1.2.2.5 Loading the Image Filenames"""


# Splitting the data into the training and validation set

#load_dataset('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Training_Data/Data Images JPG/')

train_files, train_targets = training_data['image_id'][:2000], target[:2000]


train_files


train_targets


"""#### 1.2.3 Validation Data
```

```
##### 1.2.3.1. Importing Validation Dataset

"""


# Load text files with categories as subfolder names.

path_validation_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/ISIC-2017_Validation_Part3_GroundTruth.csv"

validation_data = pd.read_csv(path_validation_data)


"""##### 1.2.3.2 Display the Validation Data of the CSV File"""


validation_data


print("Filename: \n", validation_data['image_id'][:5])


print("Targets: \n", validation_data['melanoma'][:5])


"""##### 1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding"""


# Getting the labels

target = np_utils.to_categorical(np.array(validation_data['melanoma']), 2)

target


"""##### 1.2.3.4. Checking the Number of Validation Images"""


len(validation_data['image_id'])


"""##### 1.2.3.5 Loading the Images Filenames of the Validation Data"""


# train_files, train_targets = training_data['image_id'][:2000], target[:2000]

valid_files, valid_targets = validation_data['image_id'][:150], target[:150]
```

```
valid_files


valid_targets


"""#### 1.2.4. Testing Data


##### 1.2.4.1.  Importing Testing Dataset

"""


# Load text files with categories as subfolder names.

path_testing_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Part3_GroundTruth.csv"

testing_data = pd.read_csv(path_testing_data)


"""##### 1.2.4.2. Display the Testing Data of the CSV File"""


testing_data


print("Filename: \n", testing_data['image_id'][:5])


print("Targets: \n", testing_data['melanoma'][:5])


"""##### 1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding"""


# Getting the labels

test_target = np_utils.to_categorical(np.array(testing_data['melanoma']), 2)

test_target


"""##### 1.2.4.4. Checking the Number of Testing Images"""
```

```
len(testing_data['image_id'])


"""##### 1.2.4.5. Loading the Images Filenames of the Testing Data"""


# train_files, train_targets = training_data['image_id'][:2000], target[:2000]
test_files, test_targets = testing_data['image_id'][:600], test_target[:600]


test_files


len(test_targets)


test_targets


"""### 1.3. Image preprocessing"""


def path_to_tensor(img_path):
    """
    Getting a tensor from a given path.
    """
    # Loading the image
    img = image.load_img(img_path, target_size=(512, 512))
    # Converting the image to numpy array
    x = image.img_to_array(img)
    # convert 3D tensor to 4D tensor with shape (1, 512, 512, 3)
    return np.expand_dims(x, axis=0)


def paths_to_tensor(img_paths):
    """
    # Getting a list of tensors from a given path directory.
    """
```

```
    list_of_tensors = [path_to_tensor(img_path) for img_path in tqdm(img_paths)]

    return np.vstack(list_of_tensors)


"""#### 1.3.1 Training Data


##### 1.3.1.1. Display the Filenames of Training Data

"""


train_files


"""##### 1.3.1.2. DIsplay all the Filenames present in the Dataset Directory"""


import os

os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Training_Data/Data Images JPG/")

!ls


"""##### 1.3.1.3 Joining the Filenames with the Directory Path of Every File"""


# pre-process the data for Keras

# Training Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Training_Data/Data Images JPG

# os.path.join(folder, file)

dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Training_Data/Data Images JPG')

files = []

for root, d_names, f_names in dt:

    for filename in f_names:

        files.append(os.path.join(root, filename))


files
```

```
"""##### 1.3.1.4. Converting the Complete Filenames to Tensors"""


train_tensors = paths_to_tensor(files).astype('float32')/255


"""#### 1.3.2. Validation Data


##### 1.3.2.1. Display the Filenames of Validation Data
"""


valid_files


"""##### 1.3.2.2. DIsplay all the Filenames present in the Dataset Directory"""


import os
os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Validation_Data/Data Image JPG/")
!ls


"""##### 1.3.2.3 Joining the Filenames with the Directory Path of Every File"""


dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Validation_Data/Data Image JPG')
validation_files = []
for root, d_names, f_names in dt:
    for filename in f_names:
        validation_files.append(os.path.join(root, filename))


validation_files


"""##### 1.3.2.4. Converting the Complete Filenames to Tensors"""
```

```
valid_tensors = paths_to_tensor(validation_files).astype('float32')/255


"""#### 1.3.3. Test Data


##### 1.3.3.1. Display the Filenames of Testing Data

"""


test_files


"""##### 1.3.3.2. DIsplay all the Filenames present in the Dataset Directory"""


import os

os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Test_v2_Data/Data Image JPG/")

!ls


"""##### 1.3.3.3 Joining the Filenames with the Directory Path of Every File"""


# pre-process the data for Keras

# Training Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Training_Data/Data Images JPG

# os.path.join(folder, file)

dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Test_v2_Data/Data Image JPG')

test_files = []

for root, d_names, f_names in dt:

    for filename in f_names:

        test_files.append(os.path.join(root, filename))


len(test_files)
```

```
test_files


"""##### 1.3.3.4. Converting the Complete Filenames to Tensors"""


test_tensors = paths_to_tensor(test_files).astype('float32')/255


"""### 1.4. Saving Tensor Files


#### 1.4.1 Saving Files into Drive

"""


# Saving the data

np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image
tensors/augmented_training_tensors.npy", train_tensors)

np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image
tensors/augmented_validation_tensors.npy", valid_tensors)

np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image
tensors/augmented_testing_tensors.npy", test_tensors)


"""#### 1.4.2 Loading the Tensor Files into Model"""


# Loading the data

train_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved image tensors/augmented_training_tensors.npy")

valid_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved image tensors/augmented_validation_tensors.npy")

test_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved image tensors/augmented_testing_tensors.npy")


"""## 2. Training the model


### 2.1. MobileNet Architecture
```

```
#### 2.1.1 Defining the MobileNet Architecture Function

"""


def mobilenet_architecture():
    """
    Pre-build architecture of mobilenet for our dataset.
    """
    # Imprting the model
    from keras.applications.mobilenet import MobileNet


    # Pre-build model
    base_model = MobileNet(include_top = False, weights = None, input_shape = (512,
512, 3))


    # Adding output layers
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model
    mobilenet_model = Model(base_model.input, output)


    # Getting the summary of architecture
    mobilenet_model.summary()


    # Compiling the model
    mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                            loss = 'categorical_crossentropy',
                            metrics = ['accuracy'])
```

```
    return mobilenet_model


# Getting the mobilenet

mobilenet_model = mobilenet_architecture()


"""#### 2.1.2. Creating a Checkpoint for the Model"""


checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.mobilenet.hdf5',
                               verbose=1,
                               save_best_only=True)


"""#### 2.1.3. Fitting into the Model"""


mobilenet_model.fit(train_tensors,
                    train_targets,
                    batch_size = 8,
                    validation_data = (valid_tensors, valid_targets),
                    epochs = 5,
                    callbacks=[checkpointer],
                    verbose=1)


"""#### 2.1.4. Loading the Weights for the MobileNet"""


# Loading the weights

mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.mobilenet.hdf5")


"""### 2.2. Inception Architecture


#### 2.2.1 Defining the Inception Architecture Function
```

```
"""


def inception_architecture():
    """
    Pre-build architecture of inception for our dataset.
    """
    # Imprting the model
    from keras.applications.inception_v3 import InceptionV3

    # Pre-build model
    base_model = InceptionV3(include_top = False, weights = None, input_shape = (512,
512, 3))

    # Adding output layers
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    output = Dense(units = 2, activation = 'softmax')(x)

    # Creating the whole model
    inception_model = Model(base_model.input, output)

    # Summary of the model
    inception_model.summary()

    # Compiling the model
    inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                            loss = 'categorical_crossentropy',
                            metrics = ['accuracy'])

    return inception_model
```

```
# Getting the inception

inception_model = inception_architecture()


"""#### 2.2.2 Creating a Checkpoint for the Model"""


checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.InceptionV3.hdf5',

                               verbose=1,

                               save_best_only=True)


"""#### 2.2.3 Fitting into the Model"""


inception_model.fit(train_tensors,

                    train_targets,

                    batch_size = 8,

                    validation_data = (valid_tensors, valid_targets),

                    epochs = 5,

                    callbacks=[checkpointer],

                    verbose=1)


"""#### 2.2.4 Loading the Weights for the Inception"""


# Loading the weights

inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.InceptionV3.hdf5")


"""### 2.3 Xception architecture


#### 2.3.1 Defining the Xception Architechture

"""
```

```
def xception_architecture():
    """
    Pre-build architecture of inception for our dataset.
    """
    # Imprting the model
    from keras.applications.xception import Xception

    # Pre-build model
    base_model = Xception(include_top = False, weights = None, input_shape = (512, 512,
3))

    # Adding output layers
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    output = Dense(units = 2, activation = 'softmax')(x)

    # Creating the whole model
    xception_model = Model(base_model.input, output)

    # Summary of the model
    xception_model.summary()

    # Compiling the model
    xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                           loss = 'categorical_crossentropy',
                           metrics = ['accuracy'])

    return xception_model


# Getting the xception
xception_model = xception_architecture()
```

```
"""#### 2.3.2 Creating a Checkpoint for the Model"""


tensor_board = TensorBoard(log_dir='./logs', histogram_freq = 0, batch_size = 8)


checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/xception_weights.hdf5',
                               verbose=1,
                               save_best_only=True)


"""#### 2.3.3 Fitting into the Model"""


xception_model.fit(train_tensors,
                   train_targets,
                   batch_size = 8,
                   validation_data = (valid_tensors, valid_targets),
                   epochs = 2,
                   callbacks=[checkpointer, tensor_board],
                   verbose=1)


"""#### 2.3.4 Loading the Weights"""


# Loading the weights

xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/xception_weights.hdf5")


"""## 3. Prediction


#### Declaring some Variables

"""
```

```
predict_0_0 = 0

predict_0_1 = 0


model_architecture = None

weight_path = ""


"""#### Defining the Prediction Function"""


def predict(img_path,
            model_architecture = model_architecture,
            path_model_weight = weight_path):
    # Printing the information passed to the Predict Function
    print("Image Path: "+img_path)
    print("Arhitecture Used:")
    print(model_architecture)
    print("Path for Model Weights: ")
    print(path_model_weight)
    # Getting the tensor of image
    image_to_predict = path_to_tensor(img_path).astype('float32')/255
    # Getting the model's architecture
    model = model_architecture
    # Loading the weights
    model.load_weights(path_model_weight)
    # printing the weights
    print("Model Weights: ")
    print(model.load_weights(path_model_weight))
    # Predicting
    pred = model.predict(image_to_predict)
    print("Prediction..." + " Melanoma : ", pred[0][0], " | Other : ", pred[0][1])
    predict_0_0 = pred[0][0]
```

```
    predict_0_1 = pred[0][1]

    if np.argmax(pred) == 0:

        return [1., 0.]

    elif np.argmax(pred) == 1:

        return [0., 1.]


"""### 3.1 MobileNet Architecture


#### 3.1.1 Loading the Model and Weights

"""


model_architecture = mobilenet_architecture()

weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved
Models/weights.best.mobilenet.hdf5"


"""#### 3.1.3 Predicting for a Sample Image Using MobileNet"""


predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)


"""### 3.2 Inception Architecture


#### 3.2.1 Loading the Model and Weights

"""


model_architecture = inception_architecture()

weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved
Models/weights.best.InceptionV3.hdf5"


"""#### 3.2.2 Predicting for a Sample Image Using InceptionV3"""
```

```
predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)


"""### 3.3 Xception Architecture


#### 3.3.1 Loading the Model and Weights

"""


model_architecture = xception_architecture()

weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5"


"""#### 3.3.2 Predicting a Sample Image using Xception"""


predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)


"""## 4. Evaluating the Models Individually on Validation Data"""


# Commented out IPython magic to ensure Python compatibility.

# Importing the libraries

from sklearn.metrics import roc_curve, auc

import tqdm

import matplotlib.pyplot as plt

# %matplotlib inline


"""#### Defining Function to calculatae Receiving Operating Characteristic curve"""


def compute_roc(y_true, y_score):

    """

    Computing the "Receiving Operating Characteristic curve" and area
```

```
    """
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)
    auroc = auc(false_positive_rate, true_positive_rate)
    return false_positive_rate, true_positive_rate, auroc


"""#### Defining Function for Plotting the Receiving Operating Characteristic curve"""


def plot_roc(y_true, y_score):
    """
    Ploting the Receiving Operating Characteristic curve
    """
    false_positive_rate, true_positive_rate, auroc = compute_roc(y_true, y_score)
    plt.figure(figsize=(10,6))
    plt.grid()
    plt.plot(false_positive_rate,
             true_positive_rate,
             color='darkorange',
             lw=2,
             label='ROC curve (area = {:.2f})'.format(auroc))
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate', fontsize=12)
    plt.ylabel('True Positive Rate', fontsize=12)
    plt.title('Receiver operating characteristic example', fontsize=15)
    plt.legend(loc="lower right", fontsize=14)
    plt.show()


plt.style.available
```

```
plt.style.use("seaborn-white")


"""### 4.1 MobileNet Architecture


#### 4.1.1 Computing Test Set Predictions

"""


# Compute test set predictions

#model_architecture,path_model_weight

NUMBER_TEST_SAMPLES = 150


mobilenet_architecture_function = mobilenet_architecture()

mobilenet_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.mobilenet.hdf5"


y_true_MobileNet = valid_targets[:NUMBER_TEST_SAMPLES]

y_score_MobileNet = []

for index in range(NUMBER_TEST_SAMPLES): #compute one at a time due to memory
constraints

    probs = predict(img_path = validation_files[index], model_architecture =
mobilenet_architecture_function, path_model_weight =
mobilenet_architecture_weight_path)

    print("Real values..." + "Melanoma : ", valid_targets[index][0], " | Other : ",
valid_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_MobileNet.append(probs)


correct_MobileNet = np.array(y_true_MobileNet) == np.array(y_score_MobileNet)


print("Accuracy = %2.2f%%" % (np.mean(correct_MobileNet)*100))


"""#### 4.1.2 Re-ordering Actual y for ROC"""
```

```
# Re-ordering the actual y (for ROC)

y_true_2_MobileNet = []

for i in range(len(y_true_MobileNet)):

    y_true_2_MobileNet.append(y_true_MobileNet[i][0])


"""#### 4.1.3 Re-ordering Predicte y for ROC"""


# Re-ordering the predicte y (for ROC)

y_score_2_MobileNet = []

for i in range(len(y_score_MobileNet)):

    y_score_2_MobileNet.append(y_score_MobileNet[i][0])


"""#### 4.1.4 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_MobileNet, y_score_2_MobileNet)


"""#### 4.1.5 Confusion Matrix


##### 4.1.5.1 Defining the Confusion Matrix Function

"""


def positive_negative_measurement(y_true, y_score):

    # Initialization

    TRUE_POSITIVE = 0

    FALSE_POSITIVE = 0

    TRUE_NEGATIVE = 0

    FALSE_NEGATIVE = 0


    # Calculating the model
```

```
    for i in range(len(y_score)):

        if y_true[i] == y_score[i] == 1:

            TRUE_POSITIVE += 1

        if (y_score[i] == 1) and (y_true[i] != y_score[i]):

            FALSE_POSITIVE += 1

        if y_true[i] == y_score[i] == 0:

            TRUE_NEGATIVE += 1

        if (y_score[i] == 0) and (y_true[i] != y_score[i]):

            FALSE_NEGATIVE += 1


    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet, TRUE_NEGATIVE_MobileNet,
FALSE_NEGATIVE_MobileNet = positive_negative_measurement(y_true_2_MobileNet,
y_score_2_MobileNet)

postives_negatives_MobileNet = [[TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet],
                                [FALSE_NEGATIVE_MobileNet, TRUE_NEGATIVE_MobileNet]]


postives_negatives_MobileNet


"""##### 4.1.5.2 Obtaining Labels"""


import seaborn as sns

sns.set()

labels_MobileNet =  np.array([['True positive: ' + str(TRUE_POSITIVE_MobileNet),
                               'False positive: ' + str(FALSE_POSITIVE_MobileNet)],
                               ['False negative: ' + str(FALSE_NEGATIVE_MobileNet),
                               'True negative: ' + str(TRUE_NEGATIVE_MobileNet)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_MobileNet, annot = labels_MobileNet, linewidths = 0.1,
fmt="", cmap = 'RdYlGn')
```

```
labels_MobileNet


"""##### 4.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet +
FALSE_NEGATIVE_MobileNet)

print("Sensitivity: ", sensitivity_MobileNet)


"""##### 4.1.5.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet +
FALSE_NEGATIVE_MobileNet)

    print("Specifity: ", specifity_MobileNet)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""##### 4.1.5.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet +
FALSE_POSITIVE_MobileNet)

print("Precision: ", predcision_MobileNet)


"""##### 4.1.5.6 Calculating Negative Predictive Value"""


# Negative predictive value (NPV)

try:
```

```
    npv_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet +
FALSE_NEGATIVE_MobileNet)

    print("Negative predictive value: ", npv_MobileNet)

except:

    print("0 Negative Predictions")


"""##### 4.1.5.7 Calculating Accuracy"""


# Accuracy

accuracy_MobileNet = (TRUE_POSITIVE_MobileNet + TRUE_NEGATIVE_MobileNet) /
(TRUE_POSITIVE_MobileNet + FALSE_POSITIVE_MobileNet + TRUE_NEGATIVE_MobileNet +
FALSE_NEGATIVE_MobileNet)

print("Accuracy: ", accuracy_MobileNet)


"""### 4.2 Inception Architecture


#### 4.2.1 Compute Test Set Predictions

"""


# Compute test set predictions

#model_architecture,path_model_weight

NUMBER_TEST_SAMPLES_Inception = 150


inception_architecture_function = inception_architecture()

inception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.InceptionV3.hdf5"


y_true_Inception = valid_targets[:NUMBER_TEST_SAMPLES_Inception]

y_score_Inception = []

for index in range(NUMBER_TEST_SAMPLES_Inception): #compute one at a time due to memory
constraints
```

```
    probs_Inception = predict(img_path = validation_files[index], model_architecture =
inception_architecture_function, path_model_weight =
inception_architecture_weight_path)

    print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0],
" | Other : ", valid_targets[index][1])

    print("-------------------------------------------------------------------------------
")

    y_score_Inception.append(probs_Inception)


correct_Inception = np.array(y_true_Inception) == np.array(y_score_Inception)


print("Accuracy = %2.2f%%" % (np.mean(correct_Inception)*100))


"""#### 4.2.2 Evaluating the Model


##### 4.2.2.1 Re-ordering the Actual y for ROC
"""

# Re-ordering the actual y (for ROC)
y_true_2_Inception = []
for i in range(len(y_true_Inception)):
    y_true_2_Inception.append(y_true_Inception[i][0])


"""##### 4.2.2.2 Re-ordering the Predicte y for ROC"""


# Re-ordering the predicte y (for ROC)
y_score_2_Inception = []
for i in range(len(y_score_Inception)):
    y_score_2_Inception.append(y_score_Inception[i][0])


"""##### 4.2.2.3 Plotting the Re-ordered ROC"""
```

```
plot_roc(y_true_2_Inception, y_score_2_Inception)


"""##### 4.2.2.4 Confusion Matrix


###### 4.2.2.4.1 Defining the Confusion Matrix Function

"""


def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0


    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1


    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


"""###### 4.2.2.4.2 Obtaining Labels"""
```

```
TRUE_POSITIVE_Inception, FALSE_POSITIVE_Inception, TRUE_NEGATIVE_Inception,
FALSE_NEGATIVE_Inception = positive_negative_measurement(y_true_2_Inception,
y_score_2_Inception)

postives_negatives_Inception = [[TRUE_POSITIVE_Inception, FALSE_POSITIVE_Inception],

                                [FALSE_NEGATIVE_Inception, TRUE_NEGATIVE_Inception]]


import seaborn as sns

sns.set()

labels_Inception =  np.array([['True positive: ' + str(TRUE_POSITIVE_Inception),

                  'False positive: ' + str(FALSE_POSITIVE_Inception)],

                 ['False negative: ' + str(FALSE_NEGATIVE_Inception),

                  'True negative: ' + str(TRUE_POSITIVE_Inception)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_Inception, annot = labels_Inception, linewidths = 0.1,
fmt="", cmap = 'RdYlGn')


"""###### 4.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception +
FALSE_NEGATIVE_Inception)

print("Sensitivity: ", sensitivity_Inception)


"""###### 4.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception +
FALSE_NEGATIVE_Inception)

    print("Specifity: ", specifity_Inception)

except:

    print("No Specificity due to NO NEGATIVE results.")
```

```
"""###### 4.2.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception +
FALSE_POSITIVE_Inception)

print("Precision: ", predcision_Inception)


"""###### 4.2.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception +
FALSE_NEGATIVE_Inception)

    print("Negative predictive value: ", npv_Inception)

except:

    print("0 Negative Predictions")


"""###### 4.2.2.4.7 Calculating Accuracy"""


# Accuracy

accuracy_Inception = (TRUE_POSITIVE_Inception + TRUE_NEGATIVE_Inception) /
(TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception + TRUE_NEGATIVE_Inception +
FALSE_NEGATIVE_Inception)

print("Accuracy: ", accuracy_Inception)


"""### 4.3 Xception Architecture


#### 4.3.1 Compute Test Set Predictions

"""


# Compute test set predictions
```

```
#model_architecture,path_model_weight

NUMBER_TEST_SAMPLES_Xception = 150


xception_architecture_function = xception_architecture()

xception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/xception_weights.hdf5"


y_true_Xception = valid_targets[:NUMBER_TEST_SAMPLES_Xception]

y_score_Xception = []

for index in range(NUMBER_TEST_SAMPLES_Xception): #compute one at a time due to memory
constraints

    probs_Xception = predict(img_path = validation_files[index], model_architecture =
xception_architecture_function, path_model_weight = xception_architecture_weight_path)

    print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0],
" | Other : ", valid_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_Xception.append(probs_Xception)


correct_Xception = np.array(y_true_Xception) == np.array(y_score_Xception)


print("Accuracy = %2.2f%%" % (np.mean(correct_Xception)*100))


"""#### 4.3.2 Evaluating the Model


##### 4.3.2.1 Re-ordering the Actual y for ROC

"""


# Re-ordering the actual y (for ROC)

y_true_2_Xception = []

for i in range(len(y_true_Xception)):

    y_true_2_Xception.append(y_true_Xception[i][0])
```

```
"""##### 4.3.2.2 Re-ordering the Predict y for ROC"""


# Re-ordering the predicte y (for ROC)
y_score_2_Xception = []
for i in range(len(y_score_Xception)):
    y_score_2_Xception.append(y_score_Xception[i][0])


"""##### 4.3.2.3 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_Xception, y_score_2_Xception)


"""##### 4.3.2.4 Confusion Matrix


###### 4.3.2.4.1 Defining the Confusion Matrix Function
"""


def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0


    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
```

```
        if y_true[i] == y_score[i] == 0:

            TRUE_NEGATIVE += 1

        if (y_score[i] == 0) and (y_true[i] != y_score[i]):

            FALSE_NEGATIVE += 1


    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception, TRUE_NEGATIVE_Xception,
FALSE_NEGATIVE_Xception = positive_negative_measurement(y_true_2_Xception,
y_score_2_Xception)

postives_negatives_Xception = [[TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception],

                               [FALSE_NEGATIVE_Xception, TRUE_NEGATIVE_Xception]]


"""###### 4.3.2.4.2 Obtaining the Labels"""


import seaborn as sns

sns.set()

labels_Xception =  np.array([['True positive: ' + str(TRUE_POSITIVE_Xception),

                'False positive: ' + str(FALSE_POSITIVE_Xception)],

                ['False negative: ' + str(FALSE_NEGATIVE_Xception),

                'True negative: ' + str(TRUE_POSITIVE_Xception)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_Xception, annot = labels_Xception, linewidths = 0.1,
fmt="", cmap = 'RdYlGn')


"""###### 4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception +
FALSE_NEGATIVE_Xception)

print("Sensitivity: ", sensitivity_Xception)
```

```
"""###### 4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception +
FALSE_NEGATIVE_Xception)

    print("Specifity: ", specifity_Xception)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""###### 4.3.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception +
FALSE_POSITIVE_Xception)

print("Precision: ", predcision_Xception)


"""###### 4.3.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception +
FALSE_NEGATIVE_Xception)

    print("Negative predictive value: ", npv_Xception)

except:

    print("0 Negative Predictions")


"""###### 4.3.2.4.7 Calculating Accuracy"""


# Accuracy
```

```
accuracy_Xception = (TRUE_POSITIVE_Xception + TRUE_NEGATIVE_Xception) /
(TRUE_POSITIVE_Xception + FALSE_POSITIVE_Xception + TRUE_NEGATIVE_Xception +
FALSE_NEGATIVE_Xception)

print("Accuracy: ", accuracy_Xception)


"""## 5. Evaluating the Models Together on Validation Data - Ensembling the models"""


from keras.layers import Input


"""### 5.1 Defining the Input Shape"""


# Single input for multiple models

model_input = Input(shape=(512, 512, 3))


"""### 5.2 Defining all the Models"""


def mobilenet_architecture():

    """

    Pre-build architecture of mobilenet for our dataset.

    """

    # Imprting the model

    from keras.applications.mobilenet import MobileNet


    # Pre-build model

    base_model = MobileNet(include_top = False, weights = None, input_tensor =
model_input)


    # Adding output layers

    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)
```

```
    # Creating the whole model
    mobilenet_model = Model(base_model.input, output)

    # Getting the summary of architecture
    mobilenet_model.summary()

    # Compiling the model
    mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                            loss = 'categorical_crossentropy',
                            metrics = ['accuracy'])

    return mobilenet_model

# Model 1
mobilenet_model = mobilenet_architecture()
mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.mobilenet.hdf5")

def inception_architecture():
    """
    Pre-build architecture of inception for our dataset.
    """
    # Imprting the model
    from keras.applications.inception_v3 import InceptionV3

    # Pre-build model
    base_model = InceptionV3(include_top = False, weights = None, input_tensor =
model_input)

    # Adding output layers
```

```
    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model

    inception_model = Model(base_model.input, output)


    # Summary of the model

    inception_model.summary()


    # Compiling the model

    inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),

                            loss = 'categorical_crossentropy',

                            metrics = ['accuracy'])


    return inception_model


# Model 2

inception_model = inception_architecture()

inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.InceptionV3.hdf5")


def xception_architecture():

    """

    Pre-build architecture of inception for our dataset.

    """

    # Imprting the model

    from keras.applications.xception import Xception


    # Pre-build model
```

```
    base_model = Xception(include_top = False, weights = None, input_tensor =
model_input)


    # Adding output layers

    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model

    xception_model = Model(base_model.input, output)


    # Summary of the model

    xception_model.summary()


    # Compiling the model

    xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                           loss = 'categorical_crossentropy',
                           metrics = ['accuracy'])


    return xception_model


# Model 3

xception_model = xception_architecture()

xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/xception_weights.hdf5")


"""### 5.3 Appending All the Models"""


# Appending all models

models = [mobilenet_model, inception_model, xception_model]
```

```
"""### 5.4 Defining the Ensembling Function"""


def ensemble(models, model_input):
    outputs = [model.outputs[0] for model in models]
    print("Outputs: ")
    print(outputs)
    y = keras.layers.Average()(outputs)
    print("y: ")
    print(y)
    model = Model(model_input, y, name='ensemble')
    print("Model: ")
    print(model)
    return model


# Getting ensemble model
ensemble_model = ensemble(models, model_input)


"""### 5.5 Obtaing the Weights of all the Models combined"""


# average the weights of multiple loaded models
from keras.models import load_model
from keras.models import clone_model
from numpy import average
from numpy import array


# load models from file
def load_all_models(weight_file_names_with_path):
      all_models = list()
      for epoch in range(0,len(weight_file_names_with_path)):
            # define filename for this ensemble
```

```
            filename = weight_file_names_with_path[epoch] #'model_' + str(epoch) +
'.h5'

            # load model from file

            model = load_model(filename)

            # add to list of members

            all_models.append(model)

            print('>loaded %s' % filename)

      return all_models


weight_file_names_with_path = ["/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved
Models/weights.best.mobilenet.hdf5","/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved
Models/weights.best.InceptionV3.hdf5","/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/xception_weights.hdf5"]


# load all models into memory

members = load_all_models(weight_file_names_with_path)

print('Loaded %d models' % len(members))


members


# prepare an array of equal weights

n_models = len(members)

total_weights = [1/n_models for i in range(1, n_models+1)] #[1/3 for i in
range(1,(3+1))]


n_models


total_weights


for weight in total_weights:

    weight = float(weight)
```

```
total_weights


type(total_weights)


type(total_weights[0])


new_model = None


weights_of_MobileNet = members[0].get_weights()
print(weights_of_MobileNet)


weights_of_Inception = members[1].get_weights()
print(weights_of_Inception)


weights_of_Xception = members[2].get_weights()
print(weights_of_Xception)


# # np.concatenate(weights_of_MobileNet,weights_of_Inception)
# # using naive method to concat
# for i in weights_of_Inception :
#     weights_of_MobileNet.append(i)
# # weights_of_MobileNet_and_Inception =
weights_of_MobileNet.append(weights_of_Inception)
# print("weights_of_MobileNet_and_Inception: ")
# print(weights_of_MobileNet)


# # np.concatenate(weights_of_MobileNet,weights_of_Inception)
# # using naive method to concat
# for i in weights_of_Xception :
#     weights_of_MobileNet.append(i)
```

```
# # weights_of_MobileNet_and_Inception =
weights_of_MobileNet.append(weights_of_Inception)

# print("weights_of_MobileNet_and_Inception: ")

# print(weights_of_MobileNet)


print(type(weights_of_MobileNet))


weights_of_MobileNet_and_Inception = np.concatenate((weights_of_MobileNet[0],
weights_of_Inception[0]))

print(type(weights_of_MobileNet_and_Inception))


print(weights_of_MobileNet_and_Inception.shape)


weights_of_MobileNet_Inception_and_Xception = c =
np.concatenate((weights_of_MobileNet_and_Inception, weights_of_Xception[0]))
#np.stack((weights_of_MobileNet_and_Inception,b), axis=3)

print(type(weights_of_MobileNet_Inception_and_Xception))

print(weights_of_MobileNet_Inception_and_Xception.shape)


from keras.models import Sequential

from keras.layers import Dense

from keras.models import model_from_json


# serialize model to JSON

model_weights_for_json = ensemble_model.to_json()

with open("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved
Models/Ensemble Model/ensemble_model.json", "w") as json_file:

    json_file.write(model_weights_for_json)

# serialize weights to HDF5

ensemble_model.save_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/Ensemble Model/ensemble_model.h5")

print("Saved model to disk")
```

```
# #create a model from the weights of multiple models

# def model_weight_ensemble(members, weights):

#     #determine how many layers need to be averaged

#     n_layers = len(members[0].get_weights())

#     print("No. of Layers which need to be averaged: ")

#     print(n_layers)

#     print("The weights of the Member 0 are: ")

#     print(members[0].get_weights())

#     # avg_model_weights = list()

#     # for weight in weights:

#     #     weight = float(weight)

#     # for layer in range(n_layers):

#     #     #collect this layer from each model

#     #     layer_weights = array([model.get_weights()[layer] for model in members])

#     #     # weighted average of weights for this layer

#     #     avg_layer_weights = average(layer_weights, axis = 0, weights=weights)

#     #     #store average layer weights

#     #     avg_model_weights.append(avg_layer_weights)

#     # # create a new model with the same structure

#     # model = clone_model(members[0])

#     # # set the weights in the new

#     # model.set_weights(avg_model_weights)

#     # model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

#     return model


# create a new model with the weighted average of all model weights

# new_model = model_weight_ensemble(members, weights)


# new_model
```

```
# # summarize the created model

# new_model.summary()


"""### 5.5 Evaluating ensemble model"""


# Compute test set predictions

NUMBER_TEST_SAMPLES_Ensemble = 150


y_true_Ensemble = valid_targets[:NUMBER_TEST_SAMPLES_Ensemble]

y_score_Ensemble = []

for index in range(NUMBER_TEST_SAMPLES_Ensemble): #compute one at a time due to memory
constraints

    image_to_predict_Ensemble =
path_to_tensor(validation_files[index]).astype("float32")/255.

    probs_Ensemble = ensemble_model.predict(image_to_predict_Ensemble,)

    if np.argmax(probs_Ensemble) == 0:

        y_score_Ensemble.append([1., 0.])

    elif np.argmax(probs_Ensemble) == 1:

        y_score_Ensemble.append([0., 1.])

    print("Predicted value {}... ".format(index+1) + " Melanoma : ",
probs_Ensemble[0][0], "  | Other : ", probs_Ensemble[0][1])

    print("Real values {}...".format(index+1) + "      Melanoma : ",
valid_targets[index][0], "      | Other : ", valid_targets[index][1])

    print("-------------------------------------------------------------------------
")



correct_Ensemble = np.array(y_true_Ensemble) == np.array(y_score_Ensemble)


print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble)*100))
```

```
image_to_predict_Ensemble = path_to_tensor("/content/drive/MyDrive/dataset/ISIC
Challenge 2017
Organized/Output_melanoma/ISIC_0000000_180_angle_flipped.jpg").astype('float32')/255.

ensemble_model.predict(image_to_predict_Ensemble)


"""#### 5.5.1 Compute Test Set Predictions"""


def predict_ensemble(img_path,
            model_architecture = model_architecture,
            path_model_weight = weight_path):
    # Printing the information passed to the Predict Function
    print("Image Path: ")
    print(img_path)
    print("Arhitecture Used:")
    print(model_architecture)
    print("Path for Model Weights: ")
    print(path_model_weight)
    # Getting the tensor of image
    image_to_predict = path_to_tensor(img_path).astype('float32')/255
    # Getting the model's architecture
    model = model_architecture
    # Loading the weights
    model.load_weights(path_model_weight)
    # printing the weights
    print("Model Weights: ")
    print(model.load_weights(path_model_weight))
    # Predicting
    pred = model.predict(image_to_predict)
    print("Prediction..." + " Melanoma : ", pred[0][0], " | Other : ", pred[0][1])
    predict_0_0 = pred[0][0]
    predict_0_1 = pred[0][1]
```

```
    if np.argmax(pred) == 0:

        return [1., 0.]

    elif np.argmax(pred) == 1:

        return [0., 1.]


# Compute test set predictions

#model_architecture,path_model_weight

NUMBER_TEST_SAMPLES_Ensemble = 150


all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception


y_true_Ensemble = valid_targets[:NUMBER_TEST_SAMPLES_Ensemble]

y_score_Ensemble = []

for index in range(NUMBER_TEST_SAMPLES_Ensemble): #compute one at a time due to memory
constraints

    probs_Ensemble = predict_ensemble(img_path = validation_files[index],
model_architecture = ensemble_model, path_model_weight =
"/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble
Model/ensemble_model.h5")

    print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0],
" | Other : ", valid_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_Ensemble.append(probs_Ensemble)


correct_Ensemble = np.array(y_true_Ensemble) == np.array(y_score_Ensemble)


print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble)*100))


"""#### 5.5.2 Evaluating the Model


##### 5.5.2.1 Re-ordering the Actual y for ROC

"""
```

```
# Re-ordering the actual y (for ROC)
y_true_2_Ensemble = []
for i in range(len(y_true_Ensemble)):
    y_true_2_Ensemble.append(y_true_Ensemble[i][0])

"""##### 5.5.2.2 Re-ordering the Predict y for ROC"""

# Re-ordering the predicte y (for ROC)
y_score_2_Ensemble = []
for i in range(len(y_score_Ensemble)):
    y_score_2_Ensemble.append(y_score_Ensemble[i][0])

"""##### 5.5.2.3 Plotting the Re-ordered ROC"""

plot_roc(y_true_2_Ensemble, y_score_2_Ensemble)

"""##### 5.5.2.4 Confusion Matrix

###### 5.5.2.4.1 Defining the Confusion Matrix Function
"""

def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0

    # Calculating the model
```

```
    for i in range(len(y_score)):

        if y_true[i] == y_score[i] == 1:

            TRUE_POSITIVE += 1

        if (y_score[i] == 1) and (y_true[i] != y_score[i]):

            FALSE_POSITIVE += 1

        if y_true[i] == y_score[i] == 0:

            TRUE_NEGATIVE += 1

        if (y_score[i] == 0) and (y_true[i] != y_score[i]):

            FALSE_NEGATIVE += 1


    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_Ensemble, FALSE_POSITIVE_Ensemble, TRUE_NEGATIVE_Ensemble,
FALSE_NEGATIVE_Ensemble = positive_negative_measurement(y_true_2_Ensemble,
y_score_2_Ensemble)

postives_negatives_Ensemble = [[TRUE_POSITIVE_Ensemble, FALSE_POSITIVE_Ensemble],

                               [FALSE_NEGATIVE_Ensemble, TRUE_NEGATIVE_Ensemble]]


"""###### 5.5.2.4.2 Obtaining the Labels"""


import seaborn as sns

sns.set()

labels_Ensemble =  np.array([['True positive: ' + str(TRUE_POSITIVE_Ensemble),

                  'False positive: ' + str(FALSE_POSITIVE_Ensemble)],

                 ['False negative: ' + str(FALSE_NEGATIVE_Ensemble),

                  'True negative: ' + str(TRUE_POSITIVE_Ensemble)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_Ensemble, annot = labels_Ensemble, linewidths = 0.1,
fmt="", cmap = 'RdYlGn')


"""###### 5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""
```

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble +
FALSE_NEGATIVE_Ensemble)

print("Sensitivity: ", sensitivity_Ensemble)


"""###### 5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble +
FALSE_NEGATIVE_Ensemble)

    print("Specifity: ", specifity_Ensemble)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""###### 5.5.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble +
FALSE_POSITIVE_Ensemble)

print("Precision: ", predcision_Ensemble)


"""###### 5.5.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble +
FALSE_NEGATIVE_Ensemble)

    print("Negative predictive value: ", npv_Ensemble)

except:

    print("0 Negative Predictions")
```

```
"""###### 5.5.2.4.7 Calculating Accuracy"""


# Accuracy

accuracy_Ensemble = (TRUE_POSITIVE_Ensemble + TRUE_NEGATIVE_Ensemble) /
(TRUE_POSITIVE_Ensemble + FALSE_POSITIVE_Ensemble + TRUE_NEGATIVE_Ensemble +
FALSE_NEGATIVE_Ensemble)

print("Accuracy: ", accuracy_Ensemble)


"""## 6. Evaluating the Models Individually on Testing Data"""


# Commented out IPython magic to ensure Python compatibility.

# Importing the libraries

from sklearn.metrics import roc_curve, auc

import tqdm

import matplotlib.pyplot as plt

# %matplotlib inline


"""#### Defining Function to calculatae Receiving Operating Characteristic curve"""


def compute_roc(y_true, y_score):

    """

    Computing the "Receiving Operating Characteristic curve" and area

    """

    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)

    auroc = auc(false_positive_rate, true_positive_rate)

    return false_positive_rate, true_positive_rate, auroc


"""#### Defining Function for Plotting the Receiving Operating Characteristic curve"""


def plot_roc(y_true, y_score):
```

```
    """
    Ploting the Receiving Operating Characteristic curve
    """
    false_positive_rate, true_positive_rate, auroc = compute_roc(y_true, y_score)
    plt.figure(figsize=(10,6))
    plt.grid()
    plt.plot(false_positive_rate,
             true_positive_rate,
             color='darkorange',
             lw=2,
             label='ROC curve (area = {:.2f})'.format(auroc))
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate', fontsize=12)
    plt.ylabel('True Positive Rate', fontsize=12)
    plt.title('Receiver operating characteristic example', fontsize=15)
    plt.legend(loc="lower right", fontsize=14)
    plt.show()


plt.style.available


plt.style.use("seaborn-white")


"""### 6.1 MobileNet Architecture


#### 6.1.1 Computing Test Set Predictions
"""


print("No. of Files in Test Data")
```

```
print(len(test_files))

print("No. of Target Values in Test Data")

print(len(test_targets))


# Compute test set predictions

NUMBER_TEST_SAMPLES_MobileNet_Test = 600


mobilenet_architecture_function = mobilenet_architecture()

mobilenet_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.mobilenet.hdf5"


y_true_MobileNet_Test = test_targets[:NUMBER_TEST_SAMPLES_MobileNet_Test]

print(y_true_MobileNet_Test)


y_score_MobileNet_Test = []

count = 0

for index in range(NUMBER_TEST_SAMPLES_MobileNet_Test): #compute one at a time due to
memory constraints

    count = count+1

    print(count)

    probs_MobileNet_Test = predict(img_path = test_files[index], model_architecture =
mobilenet_architecture_function, path_model_weight =
mobilenet_architecture_weight_path)

    print("Real values..." + "Melanoma : ", test_targets[index][0], " | Other : ",
test_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_MobileNet_Test.append(probs_MobileNet_Test)


correct_MobileNet_Test = np.array(y_true_MobileNet_Test) ==
np.array(y_score_MobileNet_Test)


print("Accuracy = %2.2f%%" % (np.mean(correct_MobileNet_Test)*100))
```

```
"""#### 6.1.2 Re-ordering Actual y for ROC"""


# Re-ordering the actual y (for ROC)
y_true_2_MobileNet_Test = []
for i in range(len(y_true_MobileNet_Test)):
    y_true_2_MobileNet_Test.append(y_true_MobileNet_Test[i][0])


"""#### 6.1.3 Re-ordering Predicte y for ROC"""


# Re-ordering the predicte y (for ROC)
y_score_2_MobileNet_Test = []
for i in range(len(y_score_MobileNet_Test)):
    y_score_2_MobileNet_Test.append(y_score_MobileNet_Test[i][0])


"""#### 6.1.4 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_MobileNet_Test, y_score_2_MobileNet_Test)


"""#### 6.1.5 Confusion Matrix


##### 6.1.5.1 Defining the Confusion Matrix Function
"""


def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0
```

```
    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1

    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_MobileNet_Test, FALSE_POSITIVE_MobileNet_Test,
TRUE_NEGATIVE_MobileNet_Test, FALSE_NEGATIVE_MobileNet_Test =
positive_negative_measurement(y_true_2_MobileNet_Test, y_score_2_MobileNet_Test)

postives_negatives_MobileNet_Test = [[TRUE_POSITIVE_MobileNet_Test,
FALSE_POSITIVE_MobileNet_Test],
                                     [FALSE_NEGATIVE_MobileNet_Test,
TRUE_NEGATIVE_MobileNet_Test]]


postives_negatives_MobileNet_Test


"""##### 6.1.5.2 Obtaining Labels"""


import seaborn as sns

sns.set()

labels_MobileNet_Test =  np.array([['True positive: ' +
str(TRUE_POSITIVE_MobileNet_Test),
                                   'False positive: ' +
str(FALSE_POSITIVE_MobileNet_Test)],
```

```
                                    ['False negative: ' +
str(FALSE_NEGATIVE_MobileNet_Test),

                                     'True negative: ' +
str(TRUE_NEGATIVE_MobileNet_Test)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_MobileNet_Test, annot = labels_MobileNet_Test,
linewidths = 0.1, fmt="", cmap = 'RdYlGn')



labels_MobileNet_Test


"""##### 6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test /
(TRUE_POSITIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)

print("Sensitivity: ", sensitivity_MobileNet_Test)


"""##### 6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_MobileNet_Test = TRUE_NEGATIVE_MobileNet_Test /
(TRUE_NEGATIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)

    print("Specifity: ", specifity_MobileNet_Test)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""##### 6.1.5.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test /
(TRUE_POSITIVE_MobileNet_Test + FALSE_POSITIVE_MobileNet_Test)

print("Precision: ", predcision_MobileNet_Test)
```

```
"""##### 6.1.5.6 Calculating Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_MobileNet_Test = TRUE_NEGATIVE_MobileNet_Test / (TRUE_NEGATIVE_MobileNet_Test +
FALSE_NEGATIVE_MobileNet_Test)

    print("Negative predictive value: ", npv_MobileNet_Test)

except:

    print("0 Negative Predictions")


"""##### 6.1.5.7 Calculating Accuracy"""


# Accuracy

accuracy_MobileNet_Test = (TRUE_POSITIVE_MobileNet_Test + TRUE_NEGATIVE_MobileNet_Test)
/ (TRUE_POSITIVE_MobileNet_Test + FALSE_POSITIVE_MobileNet_Test +
TRUE_NEGATIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)

print("Accuracy: ", accuracy_MobileNet_Test)


"""### 6.2 Inception Architecture


#### 6.2.1 Compute Test Set Predictions

"""


# Compute test set predictions

NUMBER_TEST_SAMPLES_Inception_Test = 600


inception_architecture_function = inception_architecture()

inception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge
2017 Organized/Saved Models/weights.best.InceptionV3.hdf5"


y_true_Inception_Test = test_targets[:NUMBER_TEST_SAMPLES_Inception_Test]
```

```
y_score_Inception_Test = []

for index in range(NUMBER_TEST_SAMPLES_Inception_Test): #compute one at a time due to
memory constraints

    probs_Inception_Test = predict(img_path = test_files[index], model_architecture =
inception_architecture_function, path_model_weight =
inception_architecture_weight_path)

    print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0],
" | Other : ", test_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_Inception_Test.append(probs_Inception_Test)


correct_Inception_Test = np.array(y_true_Inception_Test) ==
np.array(y_score_Inception_Test)


print("Accuracy = %2.2f%%" % (np.mean(correct_Inception_Test)*100))


"""#### 6.2.2 Evaluating the Model


##### 6.2.2.1 Re-ordering the Actual y for ROC

"""


# Re-ordering the actual y (for ROC)

y_true_2_Inception_Test = []

for i in range(len(y_true_Inception_Test)):

    y_true_2_Inception_Test.append(y_true_Inception_Test[i][0])


"""##### 6.2.2.2 Re-ordering the Predicte y for ROC"""


# Re-ordering the predicte y (for ROC)

y_score_2_Inception_Test = []

for i in range(len(y_score_Inception_Test)):

    y_score_2_Inception_Test.append(y_score_Inception_Test[i][0])
```

```
"""##### 6.2.2.3 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_Inception_Test, y_score_2_Inception_Test)


"""##### 6.2.2.4 Confusion Matrix


###### 6.2.2.4.1 Defining the Confusion Matrix Function
"""


def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0

    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1

    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
```

```
"""###### 6.2.2.4.2 Obtaining Labels"""


TRUE_POSITIVE_Inception_Test, FALSE_POSITIVE_Inception_Test,
TRUE_NEGATIVE_Inception_Test, FALSE_NEGATIVE_Inception_Test =
positive_negative_measurement(y_true_2_Inception_Test, y_score_2_Inception_Test)

postives_negatives_Inception_Test = [[TRUE_POSITIVE_Inception_Test,
FALSE_POSITIVE_Inception_Test],

                                     [FALSE_NEGATIVE_Inception_Test,
TRUE_NEGATIVE_Inception_Test]]


import seaborn as sns

sns.set()

labels_Inception_Test =  np.array([['True positive: ' +
str(TRUE_POSITIVE_Inception_Test),

                                   'False positive: ' +
str(FALSE_POSITIVE_Inception_Test)],

                                   ['False negative: ' +
str(FALSE_NEGATIVE_Inception_Test),

                                   'True negative: ' +
str(TRUE_POSITIVE_Inception_Test)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_Inception_Test, annot = labels_Inception_Test,
linewidths = 0.1, fmt="", cmap = 'RdYlGn')


"""###### 6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Inception_Test = TRUE_POSITIVE_Inception_Test /
(TRUE_POSITIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)

print("Sensitivity: ", sensitivity_Inception_Test)


"""###### 6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:
```

```
    specifity_Inception_Test = TRUE_NEGATIVE_Inception_Test /
(TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)

    print("Specifity: ", specifity_Inception_Test)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""###### 6.2.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Inception_Test = TRUE_POSITIVE_Inception_Test /
(TRUE_POSITIVE_Inception_Test + FALSE_POSITIVE_Inception_Test)

print("Precision: ", predcision_Inception_Test)


"""###### 6.2.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_Inception_Test = TRUE_NEGATIVE_Inception_Test / (TRUE_NEGATIVE_Inception_Test +
FALSE_NEGATIVE_Inception_Test)

    print("Negative predictive value: ", npv_Inception_Test)

except:

    print("0 Negative Predictions")


"""###### 6.2.2.4.7 Calculating Accuracy"""


# Accuracy

accuracy_Inception_Test = (TRUE_POSITIVE_Inception_Test + TRUE_NEGATIVE_Inception_Test)
/ (TRUE_POSITIVE_Inception_Test + FALSE_POSITIVE_Inception_Test +
TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)

print("Accuracy: ", accuracy_Inception_Test)


"""### 6.3 Xception Architecture
```

```
#### 6.3.1 Compute Test Set Predictions

"""


# Compute test set predictions

NUMBER_TEST_SAMPLES_Xception_Test = 600


xception_architecture_function = xception_architecture()

xception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/xception_weights.hdf5"



y_true_Xception_Test = test_targets[:NUMBER_TEST_SAMPLES_Xception_Test]

y_score_Xception_Test = []

for index in range(NUMBER_TEST_SAMPLES_Xception_Test): #compute one at a time due to
memory constraints

    probs_Xception_Test = predict(img_path = test_files[index], model_architecture =
xception_architecture_function, path_model_weight = xception_architecture_weight_path)

    print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0],
" | Other : ", test_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_Xception_Test.append(probs_Xception_Test)


correct_Xception_Test = np.array(y_true_Xception_Test) ==
np.array(y_score_Xception_Test)


print("Accuracy = %2.2f%%" % (np.mean(correct_Xception_Test)*100))


"""#### 6.3.2 Evaluating the Model


##### 6.3.2.1 Re-ordering the Actual y for ROC

"""
```

```
# Re-ordering the actual y (for ROC)

y_true_2_Xception_Test = []

for i in range(len(y_true_Xception_Test)):

    y_true_2_Xception_Test.append(y_true_Xception_Test[i][0])


"""##### 6.3.2.2 Re-ordering the Predict y for ROC"""


# Re-ordering the predicte y (for ROC)

y_score_2_Xception_Test = []

for i in range(len(y_score_Xception_Test)):

    y_score_2_Xception_Test.append(y_score_Xception_Test[i][0])


"""##### 6.3.2.3 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_Xception_Test, y_score_2_Xception_Test)


"""##### 6.3.2.4 Confusion Matrix


###### 6.3.2.4.1 Defining the Confusion Matrix Function

"""


def positive_negative_measurement(y_true, y_score):

    # Initialization

    TRUE_POSITIVE = 0

    FALSE_POSITIVE = 0

    TRUE_NEGATIVE = 0

    FALSE_NEGATIVE = 0


    # Calculating the model
```

```
    for i in range(len(y_score)):

        if y_true[i] == y_score[i] == 1:

            TRUE_POSITIVE += 1

        if (y_score[i] == 1) and (y_true[i] != y_score[i]):

            FALSE_POSITIVE += 1

        if y_true[i] == y_score[i] == 0:

            TRUE_NEGATIVE += 1

        if (y_score[i] == 0) and (y_true[i] != y_score[i]):

            FALSE_NEGATIVE += 1


    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_Xception_Test, FALSE_POSITIVE_Xception_Test, TRUE_NEGATIVE_Xception_Test,
FALSE_NEGATIVE_Xception_Test = positive_negative_measurement(y_true_2_Xception_Test,
y_score_2_Xception_Test)

postives_negatives_Xception_Test = [[TRUE_POSITIVE_Xception_Test,
FALSE_POSITIVE_Xception_Test],

                                    [FALSE_NEGATIVE_Xception_Test,
TRUE_NEGATIVE_Xception_Test]]


"""###### 6.3.2.4.2 Obtaining the Labels"""


import seaborn as sns

sns.set()

labels_Xception_Test =  np.array([['True positive: ' +
str(TRUE_POSITIVE_Xception_Test),

                                  'False positive: ' +
str(FALSE_POSITIVE_Xception_Test)],

                                  ['False negative: ' +
str(FALSE_NEGATIVE_Xception_Test),

                                  'True negative: ' +
str(TRUE_POSITIVE_Xception_Test)]])

plt.figure(figsize = (13, 10))
```

```
sns.heatmap(postives_negatives_Xception_Test, annot = labels_Xception_Test, linewidths
= 0.1, fmt="", cmap = 'RdYlGn')


"""###### 6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test
+ FALSE_NEGATIVE_Xception_Test)

print("Sensitivity: ", sensitivity_Xception_Test)


"""###### 6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)

try:

    specifity_Xception_Test = TRUE_NEGATIVE_Xception_Test /
(TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)

    print("Specifity: ", specifity_Xception_Test)

except:

    print("No Specificity due to NO NEGATIVE results.")


"""###### 6.3.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test +
FALSE_POSITIVE_Xception_Test)

print("Precision: ", predcision_Xception_Test)


"""###### 6.3.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:
```

```
    npv_Xception_Test = TRUE_NEGATIVE_Xception_Test / (TRUE_NEGATIVE_Xception_Test +
FALSE_NEGATIVE_Xception_Test)

    print("Negative predictive value: ", npv_Xception_Test)

except:

    print("0 Negative Predictions")


"""###### 6.3.2.4.7 Calculating Accuracy"""


# Accuracy

accuracy_Xception_Test = (TRUE_POSITIVE_Xception_Test + TRUE_NEGATIVE_Xception_Test) /
(TRUE_POSITIVE_Xception_Test + FALSE_POSITIVE_Xception_Test +
TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)

print("Accuracy: ", accuracy_Xception_Test)


"""## 7. Evaluating the Models Together on Testing Data - Ensembling the models"""


from keras.layers import Input


"""### 7.1 Defining the Input Shape"""


# Single input for multiple models

model_input = Input(shape=(512, 512, 3))


"""### 7.2 Defining all the Models"""


def mobilenet_architecture():

    """

    Pre-build architecture of mobilenet for our dataset.

    """

    # Imprting the model

    from keras.applications.mobilenet import MobileNet
```

```
    # Pre-build model

    base_model = MobileNet(include_top = False, weights = None, input_tensor =
model_input)


    # Adding output layers

    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model

    mobilenet_model = Model(base_model.input, output)


    # Getting the summary of architecture

    mobilenet_model.summary()


    # Compiling the model

    mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),

                            loss = 'categorical_crossentropy',

                           metrics = ['accuracy'])


    return mobilenet_model


# Model 1

mobilenet_model = mobilenet_architecture()

mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.mobilenet.hdf5")


def inception_architecture():

    """

    Pre-build architecture of inception for our dataset.
```

```
    """

    # Imprting the model

    from keras.applications.inception_v3 import InceptionV3


    # Pre-build model

    base_model = InceptionV3(include_top = False, weights = None, input_tensor =
model_input)


    # Adding output layers

    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model

    inception_model = Model(base_model.input, output)


    # Summary of the model

    inception_model.summary()


    # Compiling the model

    inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),

                            loss = 'categorical_crossentropy',

                            metrics = ['accuracy'])


    return inception_model


# Model 2

inception_model = inception_architecture()

inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.InceptionV3.hdf5")
```

```
def xception_architecture():

    """

    Pre-build architecture of inception for our dataset.

    """

    # Imprting the model

    from keras.applications.xception import Xception


    # Pre-build model

    base_model = Xception(include_top = False, weights = None, input_tensor =
model_input)


    # Adding output layers

    x = base_model.output

    x = GlobalAveragePooling2D()(x)

    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model

    xception_model = Model(base_model.input, output)


    # Summary of the model

    xception_model.summary()


    # Compiling the model

    xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),

                           loss = 'categorical_crossentropy',

                           metrics = ['accuracy'])


    return xception_model


# Model 3

xception_model = xception_architecture()
```

```
xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5")


"""### 7.3 Appending All the Models"""


# Appending all models

models = [mobilenet_model, inception_model, xception_model]


"""### 7.4 Defining the Ensembling Function"""


def ensemble(models, model_input):

    outputs = [model.outputs[0] for model in models]

    y = keras.layers.Average()(outputs)

    model = Model(model_input, y, name='ensemble')

    return model


# Getting ensemble model

ensemble_model = ensemble(models, model_input)


"""### 7.5 Evaluating ensemble model"""


# Compute test set predictions

#model_architecture,path_model_weight

NUMBER_TEST_SAMPLES_Ensemble_Test = 600


all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception


y_true_Ensemble_Test = test_targets[:NUMBER_TEST_SAMPLES_Ensemble_Test]

y_score_Ensemble_Test = []

for index in range(NUMBER_TEST_SAMPLES_Ensemble_Test): #compute one at a time due to memory constraints
```

```
    probs_Ensemble_Test = predict_ensemble(img_path = test_files[index],
model_architecture = ensemble_model, path_model_weight =
"/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble
Model/ensemble_model.h5")

    print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0],
" | Other : ", test_targets[index][1])

    print("-------------------------------------------------------------------------
")

    y_score_Ensemble_Test.append(probs_Ensemble_Test)


correct_Ensemble_Test = np.array(y_true_Ensemble_Test) ==
np.array(y_score_Ensemble_Test)


print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))


image_to_predict_Ensemble_Test = path_to_tensor("/content/drive/MyDrive/dataset/ISIC
Challenge 2017
Organized/Output_melanoma/ISIC_0000000_180_angle_flipped.jpg").astype('float32')/255.

ensemble_model.predict(image_to_predict_Ensemble_Test)


"""#### 7.5.1 Compute Test Set Predictions"""


# Compute test set predictions

NUMBER_TEST_SAMPLES_Ensemble_Test = 600


all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception


y_true_Ensemble_Test = test_targets[:NUMBER_TEST_SAMPLES_Ensemble_Test]

y_score_Ensemble_Test = []

for index in range(NUMBER_TEST_SAMPLES_Ensemble_Test): #compute one at a time due to
memory constraints

    probs_Ensemble_Test = predict(img_path = test_files[index], model_architecture =
ensemble_model, path_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/Ensemble Model/ensemble_model.h5")
```

```
    print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0],
" | Other : ", test_targets[index][1])

    print("-------------------------------------------------------------------------------
")

    y_score_Ensemble_Test.append(probs_Ensemble_Test)


correct_Ensemble_Test = np.array(y_true_Ensemble_Test) ==
np.array(y_score_Ensemble_Test)


print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))


"""#### 7.5.2 Evaluating the Model


##### 7.5.2.1 Re-ordering the Actual y for ROC
"""

# Re-ordering the actual y (for ROC)
y_true_2_Ensemble_Test = []
for i in range(len(y_true_Ensemble_Test)):
    y_true_2_Ensemble_Test.append(y_true_Ensemble_Test[i][0])


"""##### 7.5.2.2 Re-ordering the Predict y for ROC"""


# Re-ordering the predicte y (for ROC)
y_score_2_Ensemble_Test = []
for i in range(len(y_score_Ensemble_Test)):
    y_score_2_Ensemble_Test.append(y_score_Ensemble_Test[i][0])


"""##### 7.5.2.3 Plotting the Re-ordered ROC"""


plot_roc(y_true_2_Ensemble_Test, y_score_2_Ensemble_Test)
```

```
"""##### 7.5.2.4 Confusion Matrix


###### 7.5.2.4.1 Defining the Confusion Matrix Function
"""


def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0

    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1

    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)


TRUE_POSITIVE_Ensemble_Test, FALSE_POSITIVE_Ensemble_Test, TRUE_NEGATIVE_Ensemble_Test,
FALSE_NEGATIVE_Ensemble_Test = positive_negative_measurement(y_true_2_Ensemble_Test,
y_score_2_Ensemble_Test)

postives_negatives_Ensemble_Test = [[TRUE_POSITIVE_Ensemble_Test,
FALSE_POSITIVE_Ensemble_Test],
```

```
                                    [FALSE_NEGATIVE_Ensemble_Test,
TRUE_NEGATIVE_Ensemble_Test]]


"""###### 7.5.2.4.2 Obtaining the Labels"""


import seaborn as sns

sns.set()

labels_Ensemble_Test =  np.array([['True positive: ' +
str(TRUE_POSITIVE_Ensemble_Test),

                                  'False positive: ' +
str(FALSE_POSITIVE_Ensemble_Test)],

                                  ['False negative: ' +
str(FALSE_NEGATIVE_Ensemble_Test),

                                  'True negative: ' +
str(TRUE_POSITIVE_Ensemble_Test)]])

plt.figure(figsize = (13, 10))

sns.heatmap(postives_negatives_Ensemble_Test, annot = labels_Ensemble_Test, linewidths
= 0.1, fmt="", cmap = 'RdYlGn')


"""###### 7.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate"""


# Sensitivity | Recall | hit rate | true positive rate (TPR)

sensitivity_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test
+ FALSE_NEGATIVE_Ensemble_Test)

print("Sensitivity: ", sensitivity_Ensemble_Test)


"""###### 7.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate"""


# Specificity | selectivity | true negative rate (TNR)
try:

    specifity_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test /
(TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)

    print("Specifity: ", specifity_Ensemble_Test)

except:
```

```
    print("No Specificity due to NO NEGATIVE results.")


"""###### 7.5.2.4.5 Calculating Precision/Positive Predictive Value"""


# Precision | positive predictive value (PPV)

predcision_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test +
FALSE_POSITIVE_Ensemble_Test)

print("Precision: ", predcision_Ensemble_Test)


"""###### 7.5.2.4.6 Negative Predictive Value"""


# Negative predictive value (NPV)

try:

    npv_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test / (TRUE_NEGATIVE_Ensemble_Test +
FALSE_NEGATIVE_Ensemble_Test)

    print("Negative predictive value: ", npv_Ensemble_Test)

except:

    print("0 Negative Predictions")


"""###### 7.5.2.4.7 Calculating Accuracy"""


# Accuracy

accuracy_Ensemble_Test = (TRUE_POSITIVE_Ensemble_Test + TRUE_NEGATIVE_Ensemble_Test) /
(TRUE_POSITIVE_Ensemble_Test + FALSE_POSITIVE_Ensemble_Test +
TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)

print("Accuracy: ", accuracy_Ensemble_Test)


"""## 8. Localization"""


# Importing the libraries

from keras.applications.mobilenet import preprocess_input

import scipy
```

```
import cv2


"""### 8.1 Obtaining the Path for the MobileNet Architecture Models"""


path_to_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017
Organized/Saved Models/weights.best.mobilenet.hdf5"


"""### 8.2 Sample Image Path"""


img_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-
2017_Test_v2_Data/Data Image JPG/ISIC_0012086.jpg"


"""### 8.3 Defining the function for the Layer Weights for MobileNet"""


def getting_two_layer_weights(path_model_weight = path_to_model_weight):
    # The model

    # Imprting the model
    from keras.applications.mobilenet import MobileNet


    # Pre-build model
    base_model = MobileNet(include_top = False, weights = None, input_shape = (512,
512, 3))


    # Adding output layers
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    output = Dense(units = 2, activation = 'softmax')(x)


    # Creating the whole model
    model = Model(base_model.input, output)
```

```
    #model.summary()


    # Compiling the model

    model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),

                  loss = 'categorical_crossentropy',

                  metrics = ['accuracy'])


    # loading the weights

    model.load_weights(path_model_weight)


    # Getting the AMP layer weight

    all_amp_layer_weights = model.layers[-1].get_weights()[0]


    # Extracting the wanted output

    mobilenet_model = Model(inputs = model.input, outputs = (model.layers[-3].output,
model.layers[-1].output))


    return mobilenet_model, all_amp_layer_weights


mobilenet_model, all_amp_layer_weights =
getting_two_layer_weights(path_to_model_weight)


"""### 8.4 Defining Class Activation Map Function"""


def mobilenet_CAM(img_path, model, all_amp_layer_weights):

    # Getting filtered images from last convolutional layer + model prediction output

    last_conv_output, predictions = model.predict(path_to_tensor(img_path)) #
last_conv_output.shape = (1, 16, 16, 1024)


    # Converting the dimension of last convolutional layer to 16 x 16 x 1024

    last_conv_output = np.squeeze(last_conv_output)
```

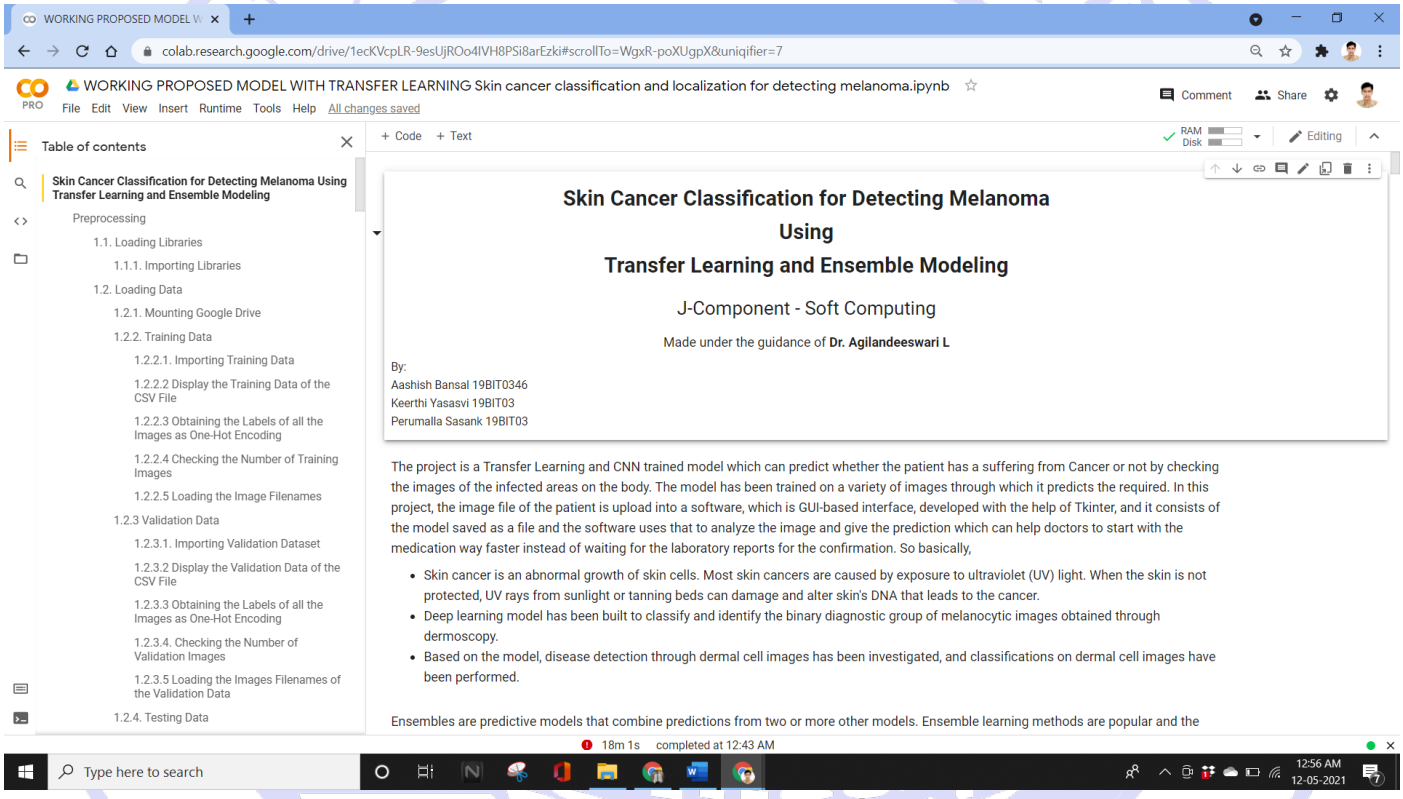
```
    # Model's prediction

    predicted_class = np.argmax(predictions)


    # Bilinear upsampling (resize each image to size of original image)

    mat_for_mult = scipy.ndimage.zoom(last_conv_output, (32, 32, 1), order = 1)  # dim
from (16, 16, 1024) to (512, 512, 1024)


    # Getting the AMP layer weights

    amp_layer_weights = all_amp_layer_weights[:, predicted_class] # dim: (1024,)


    # CAM for object class that is predicted to be in the image

    final_output = np.dot(mat_for_mult, amp_layer_weights) # dim: 512 x 512


    # Return class activation map (CAM)

    return final_output, predicted_class


final_output, predicted_class = mobilenet_CAM(img_path, mobilenet_model,
all_amp_layer_weights)


"""### 8.5 Plotting the Class Activation Function for MobileNet"""


def plot_CAM(img_path, ax, model, all_amp_layer_weights):

    # Loading the image / resizing to 512x512 / Converting BGR to RGB

    #im = cv2.resize(cv2.cvtColor(cv2.imread(img_path), cv2.COLOR_BGR2RGB), (512, 512))

    im = path_to_tensor(img_path).astype("float32")/255.


    # Plotting the image

    ax.imshow(im.squeeze(), vmin=0, vmax=255)


    # Getting the class activation map
```

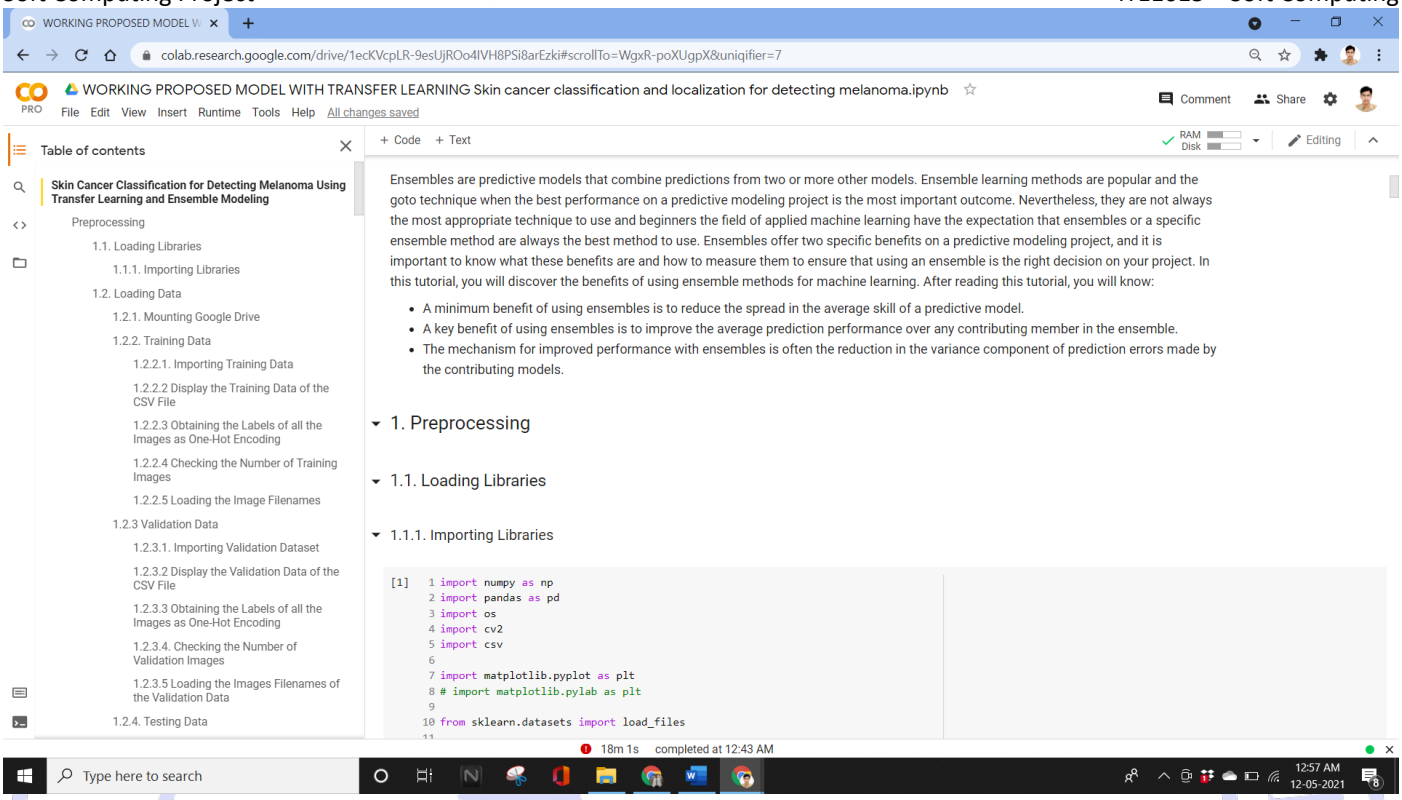
```
    CAM, pred = mobilenet_CAM(img_path, model, all_amp_layer_weights)


    CAM = (CAM - CAM.min()) / (CAM.max() - CAM.min())


    # Plotting the class activation map
    ax.imshow(CAM, cmap = "jet", alpha = 0.5, interpolation='nearest', vmin=0, vmax=1)


"""### 8.6 Visualizing the Images"""


# Visualizing images with and without localization
# Canvas
fig, ax = plt.subplots(nrows=1, ncols=2, figsize = (10, 10))
# Image without localization
ax[0].imshow((path_to_tensor(img_path).astype('float32')/255).squeeze())
# Image with localization
CAM = plot_CAM(img_path, ax[1], mobilenet_model, all_amp_layer_weights)
plt.show()


# predict_0_0 = 0.7350246
# predict_0_1 = 0.26497543


# Getting the image tensor
image_to_predict = path_to_tensor(img_path).astype('float32')/255


# Predicting the image
prediction = ensemble_model.predict(image_to_predict)
prediction_final = "Melanoma: " + str(np.round(predict_0_0*100, decimals = 4)) + "%" +
\
                   " | Other illness: " + str(np.round(predict_0_1*100, decimals = 4))
+ "%"
```

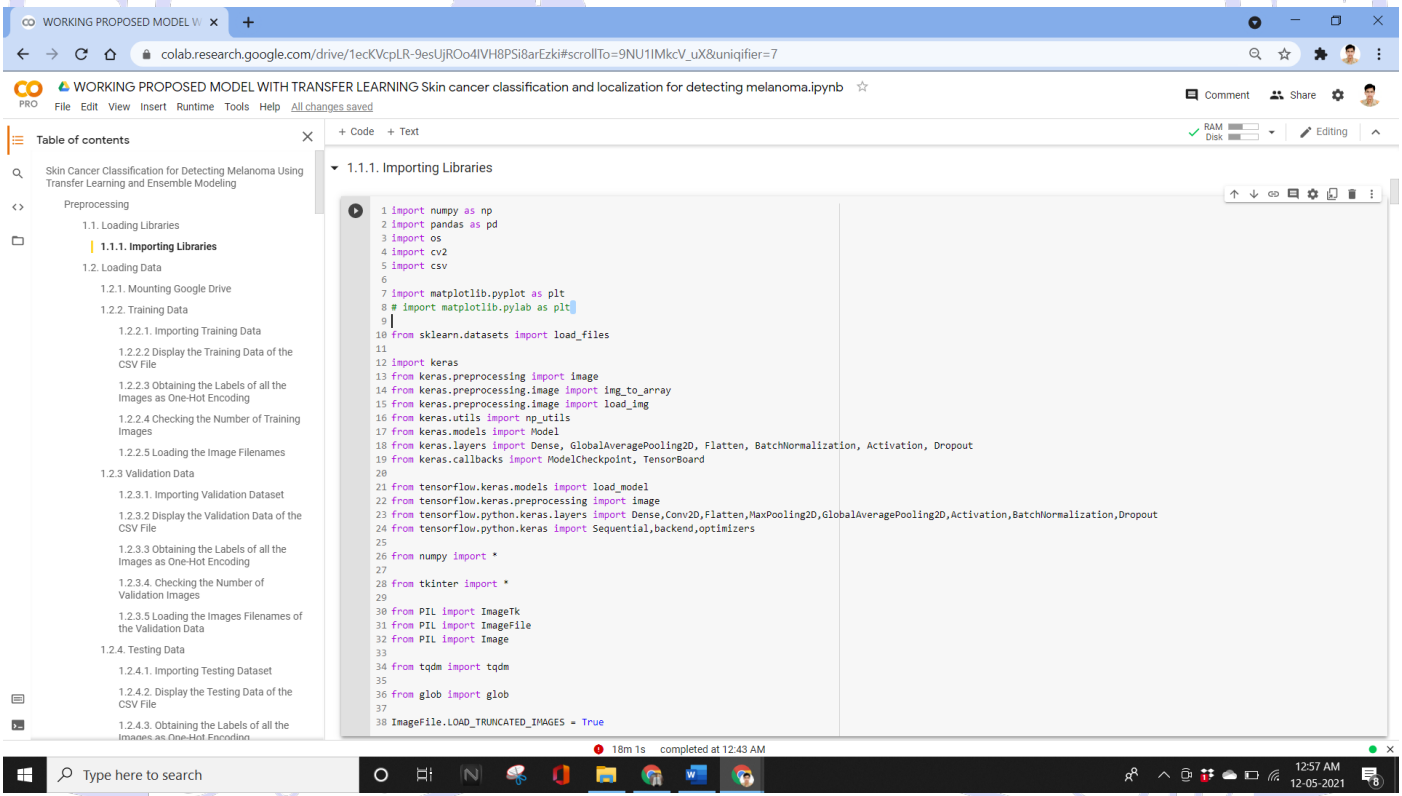
```
# Canvas initialization

fig = plt.figure(figsize = (10, 10))


# First image

ax = fig.add_subplot(121)

ax.imshow(image_to_predict.squeeze())

ax.text(0.3, 1.6, prediction_final)


# Second image

ax = fig.add_subplot(122)

CAM = plot_CAM(img_path, ax, mobilenet_model, all_amp_layer_weights)


plt.show()


"""## 9. Saving the Complete Model for the Python Interface Application"""


# Commented out IPython magic to ensure Python compatibility.

# %cd "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED
OUTPUTS/"

ensemble_model.save('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')

from tensorflow.keras.models import load_model

h5_saved_ensemble_model =
load_model('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')

h5_saved_ensemble_model.summary()


"""## 10. Converting the .h5 File to .tflite File for the Python Interface
Application"""


import tensorflow as tf
```

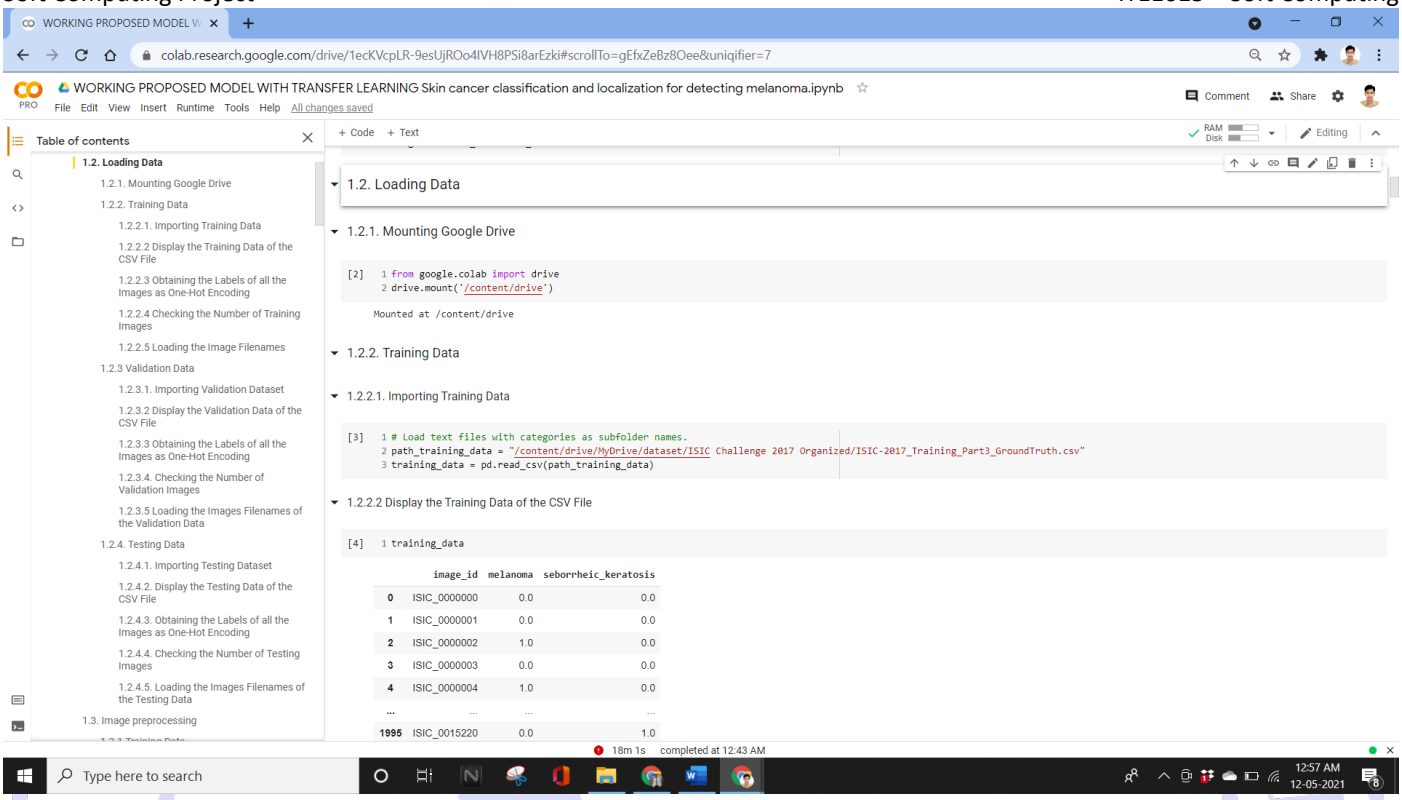
```
saved_ensemble_model = tf.keras.models.load_model('/content/drive/MyDrive/dataset/ISIC
Challenge 2017 Organized/FINAL SAVED
OUTPUTS/FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')

converter = tf.lite.TFLiteConverter.from_keras_model(saved_ensemble_model)

tflite_model = converter.convert()

open("FINAL_FILE_for_Interface_Soft_Computing_Project_Skin_Cancer.tflite",
"wb").write(tflite_model)
```

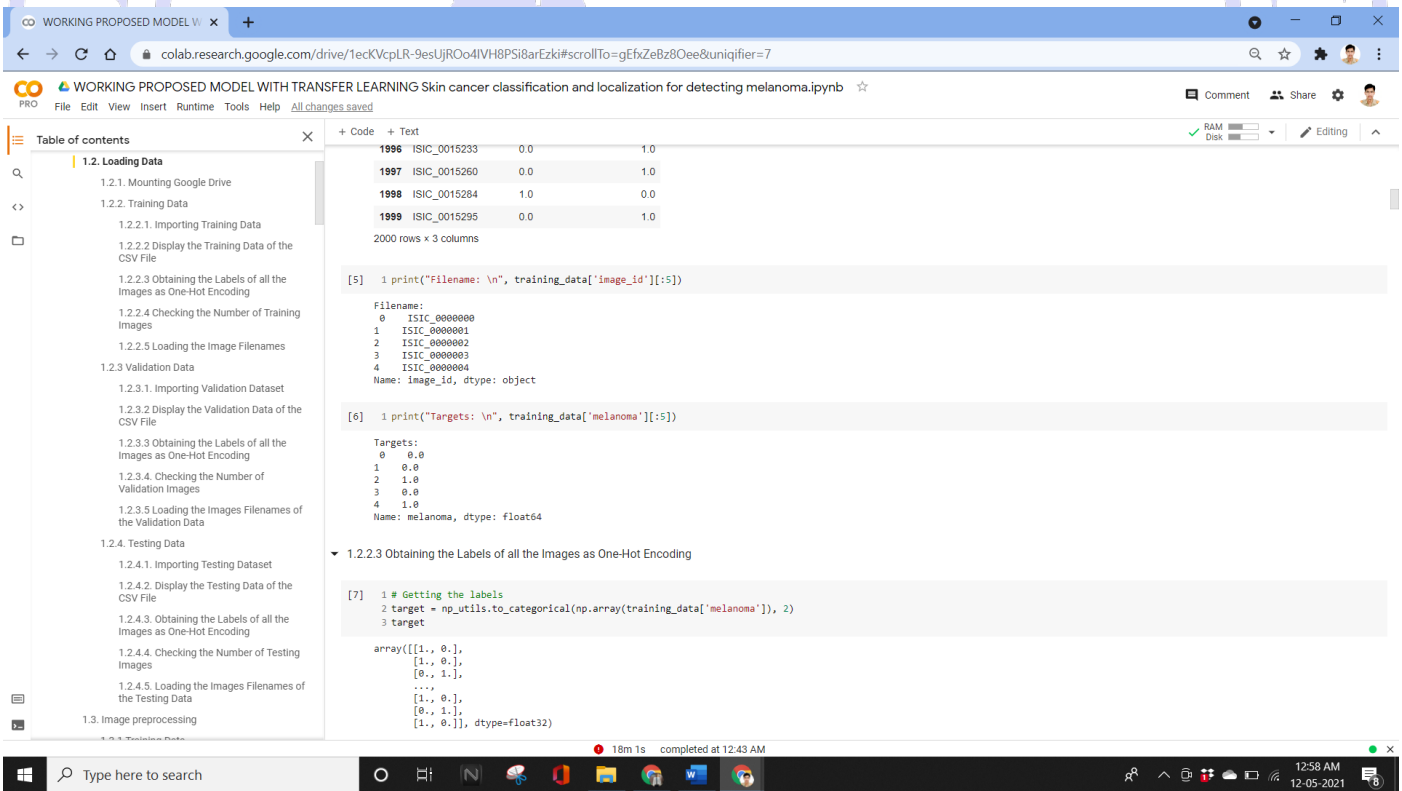
SOURCE CODE SCREENSHOTS IN COLAB

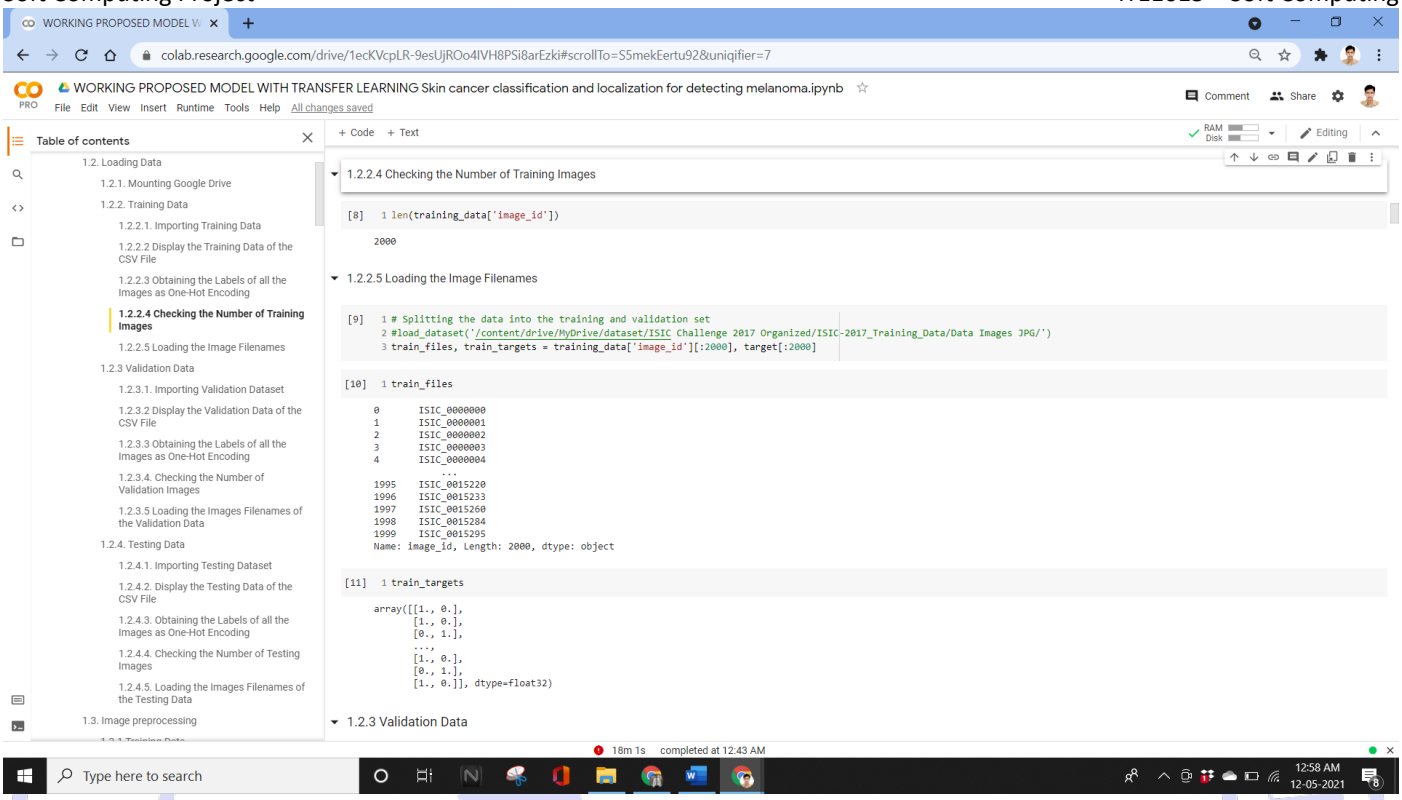


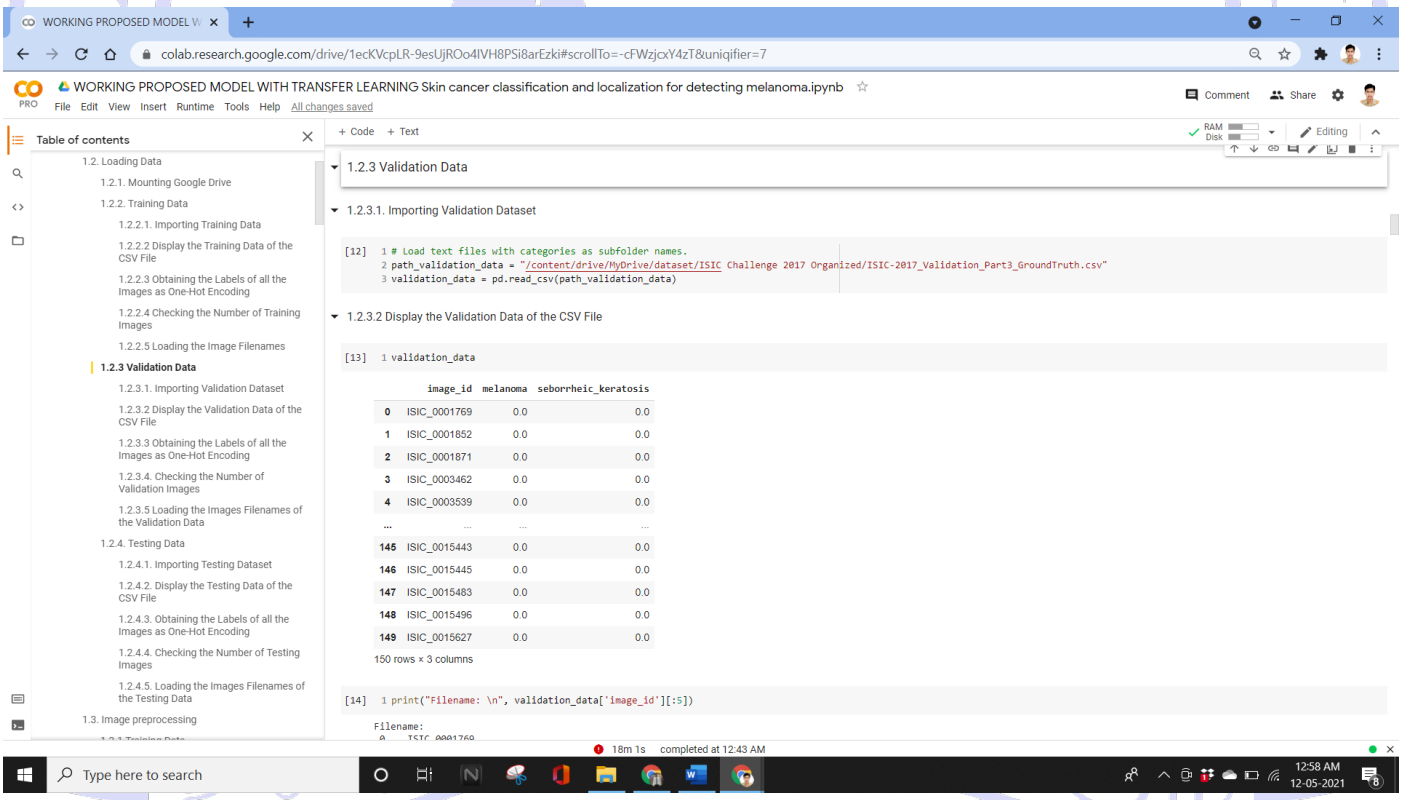

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=WgxR-poXUgpX&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Skin Cancer Classification for Detecting Melanoma Using Transfer Learning and Ensemble Modeling
Preprocessing
1.1. Loading Libraries
1.1.1. Importing Libraries
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
+ Code + Text
RAM Disk
Editing
Ensembles are predictive models that combine predictions from two or more other models. Ensemble learning methods are popular and the goto technique when the best performance on a predictive modeling project is the most important outcome. Nevertheless, they are not always the most appropriate technique to use and beginners the field of applied machine learning have the expectation that ensembles or a specific ensemble method are always the best method to use. Ensembles offer two specific benefits on a predictive modeling project, and it is important to know what these benefits are and how to measure them to ensure that using an ensemble is the right decision on your project. In this tutorial, you will discover the benefits of using ensemble methods for machine learning. After reading this tutorial, you will know:
A minimum benefit of using ensembles is to reduce the spread in the average skill of a predictive model.
A key benefit of using ensembles is to improve the average prediction performance over any contributing member in the ensemble.
The mechanism for improved performance with ensembles is often the reduction in the variance component of prediction errors made by the contributing models.
1. Preprocessing
1.1. Loading Libraries
1.1.1. Importing Libraries
[1]
import numpy as np
import pandas as pd
import os
import cv2
import csv
import matplotlib.pyplot as plt
# import matplotlib.pylab as plt
from sklearn.datasets import load_files
18m 1s completed at 12:43 AM
Type here to search
12:57 AM
12-05-2021

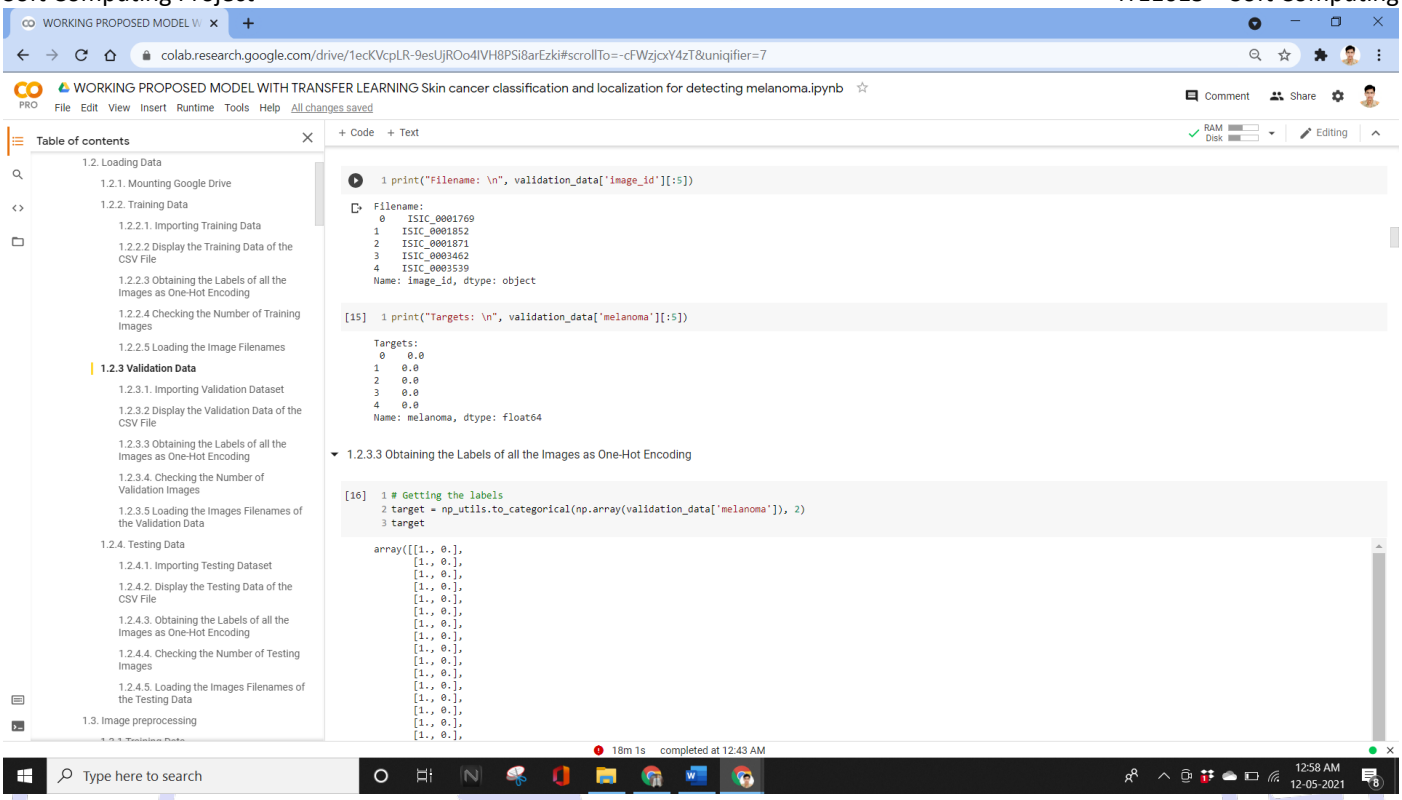

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=9NU1IMkcV_uX&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Skin Cancer Classification for Detecting Melanoma Using Transfer Learning and Ensemble Modeling
Preprocessing
1.1. Loading Libraries
1.1.1. Importing Libraries
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
+ Code + Text
RAM Disk
Editing
1.1.1. Importing Libraries
import numpy as np
import pandas as pd
import os
import cv2
import csv
import matplotlib.pyplot as plt
# import matplotlib.pylab as plt
from sklearn.datasets import load_files
import keras
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from keras.utils import np_utils
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D, Flatten, BatchNormalization, Activation, Dropout
from keras.callbacks import ModelCheckpoint, TensorBoard
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.python.keras.layers import Dense,Conv2D,Flatten,MaxPooling2D,GlobalAveragePooling2D,Activation,BatchNormalization,Dropout
from tensorflow.python.keras import Sequential,backend,optimizers
from numpy import *
from tkinter import *
from PIL import ImageTk
from PIL import ImageFile
from PIL import Image
from tqdm import tqdm
from glob import glob
ImageFile.LOAD_TRUNCATED_IMAGES = True
18m 1s completed at 12:43 AM
Type here to search
12:57 AM
12-05-2021

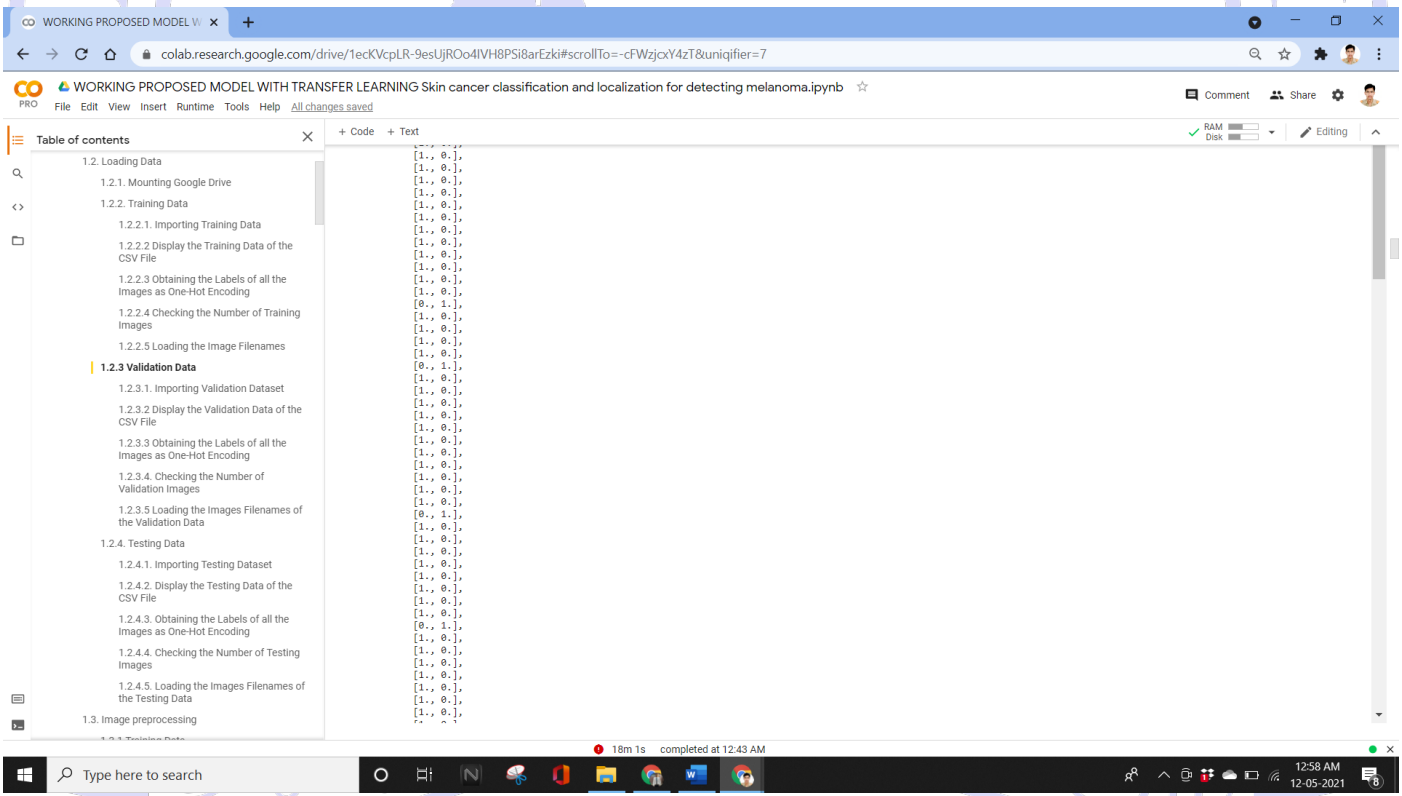

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=gEfxZeBz8Oee&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1.2. Loading Data
1.2.1. Mounting Google Drive
[2] 1 from google.colab import drive
2 drive.mount('/content/drive')
Mounted at /content/drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
[3] 1 # Load text files with categories as subfolder names.
2 path_training_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Part3_GroundTruth.csv"
3 training_data = pd.read_csv(path_training_data)
1.2.2.2 Display the Training Data of the CSV File
[4] 1 training_data
image_id melanoma seborrheic_keratosis
0 ISIC_0000000 0.0 0.0
1 ISIC_0000001 0.0 0.0
2 ISIC_0000002 1.0 0.0
3 ISIC_0000003 0.0 0.0
4 ISIC_0000004 1.0 0.0
... ...
1995 ISIC_0015220 0.0 1.0
18m 1s completed at 12:43 AM
Type here to search
12:57 AM
12-05-2021

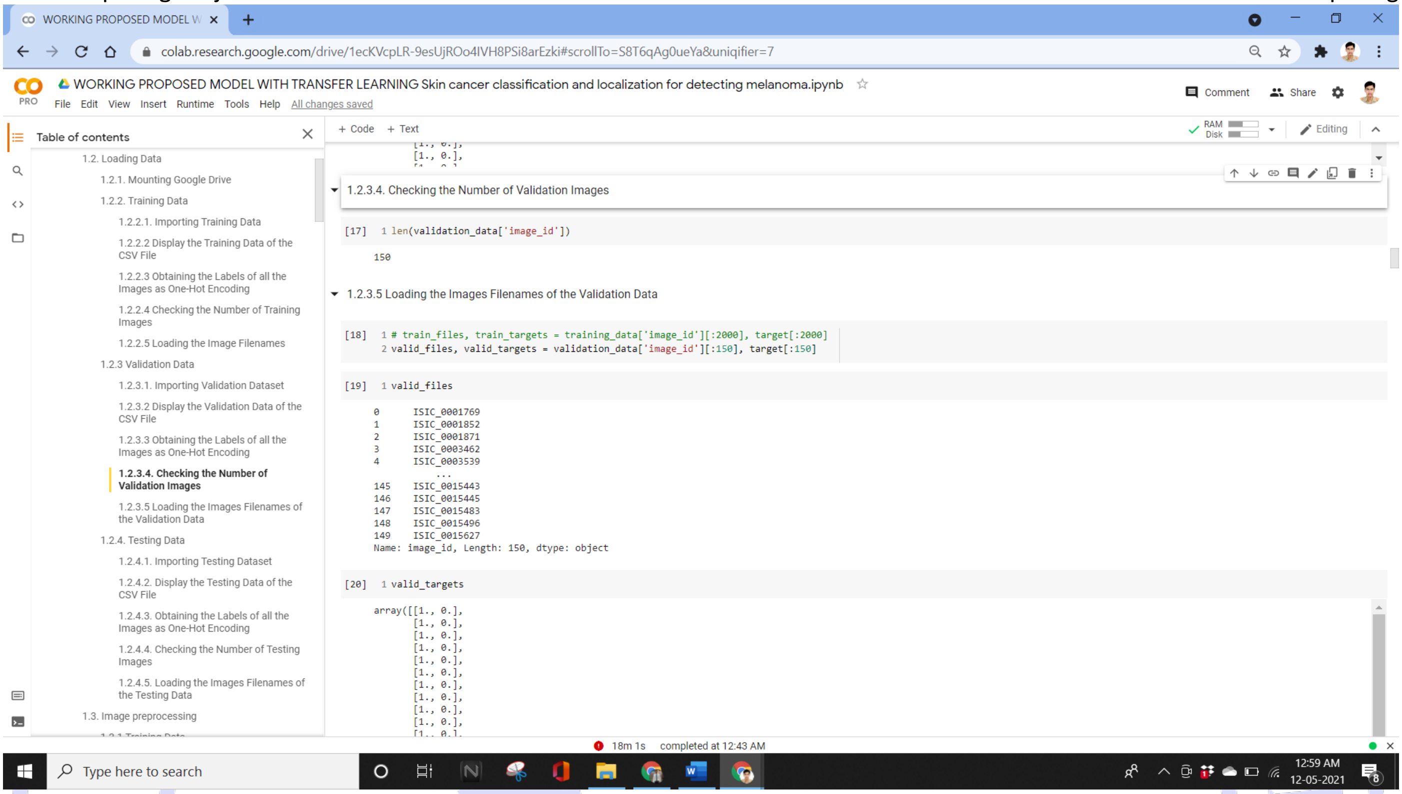

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=gEfxZeBz8Oee&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1996 ISIC_0015233 0.0 1.0
1997 ISIC_0015260 0.0 1.0
1998 ISIC_0015284 1.0 0.0
1999 ISIC_0015295 0.0 1.0
2000 rows × 3 columns
[5] 1 print("Filename: \n", training_data['image_id'][:5])
Filename:
0 ISIC_0000000
1 ISIC_0000001
2 ISIC_0000002
3 ISIC_0000003
4 ISIC_0000004
Name: image_id, dtype: object
[6] 1 print("Targets: \n", training_data['melanoma'][:5])
Targets:
0 0.0
1 0.0
2 1.0
3 0.0
4 1.0
Name: melanoma, dtype: float64
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
[7] 1 # Getting the labels
2 target = np_utils.to_categorical(np.array(training_data['melanoma']), 2)
3 target
array([[1., 0.],
[1., 0.],
[0., 1.],
...,
[1., 0.],
[0., 1.],
[1., 0.]], dtype=float32)
18m 1s completed at 12:43 AM
Type here to search
12:58 AM
12-05-2021

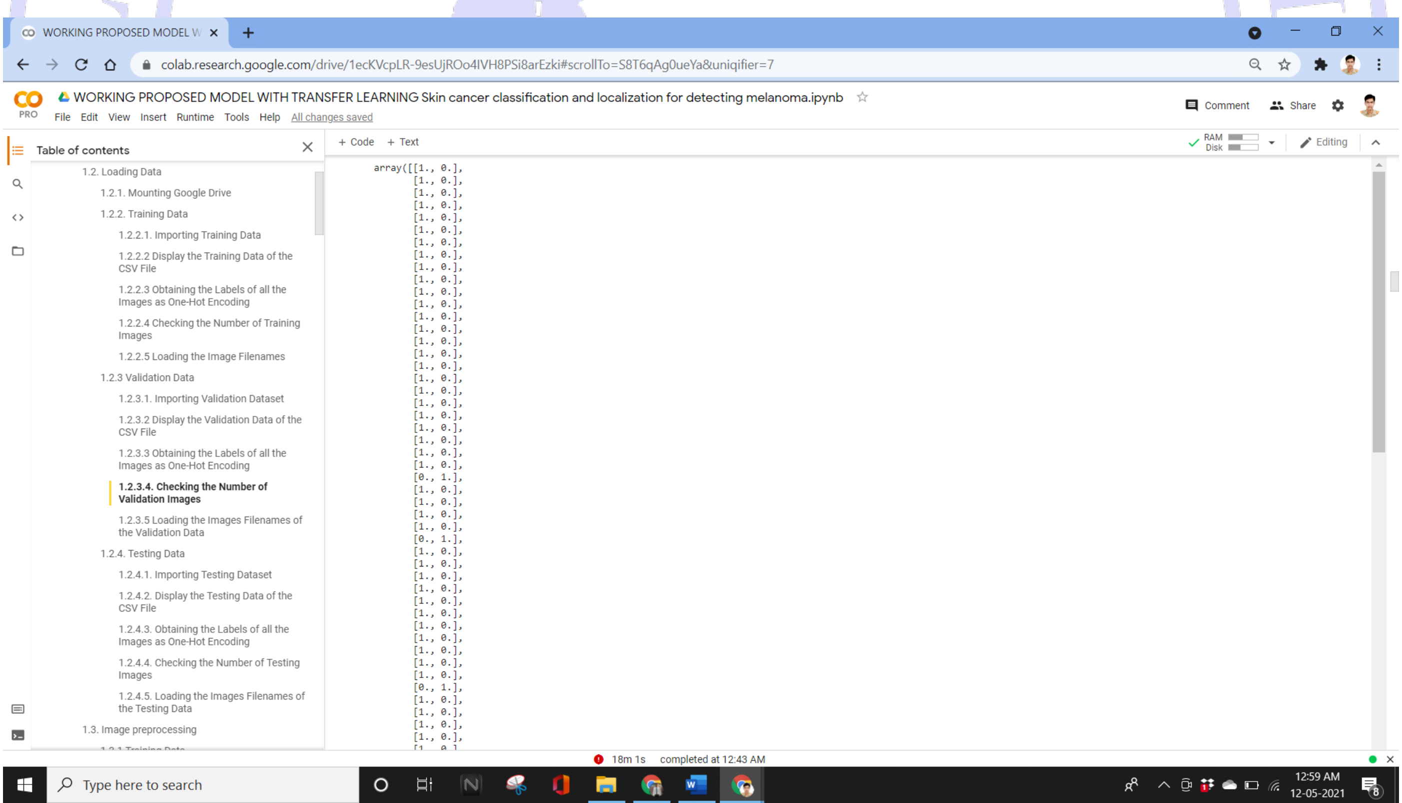

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=S5mekEertu92&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1.2.2.4 Checking the Number of Training Images
[8] 1 len(training_data['image_id'])
2000
1.2.2.5 Loading the Image Filenames
[9] 1 # Splitting the data into the training and validation set
2 #load_dataset('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/')
3 train_files, train_targets = training_data['image_id'][:2000], target[:2000]
[10] 1 train_files
0 ISIC_0000000
1 ISIC_0000001
2 ISIC_0000002
3 ISIC_0000003
4 ISIC_0000004
...
1995 ISIC_0015220
1996 ISIC_0015233
1997 ISIC_0015260
1998 ISIC_0015284
1999 ISIC_0015295
Name: image_id, Length: 2000, dtype: object
[11] 1 train_targets
array([[1., 0.],
[1., 0.],
[0., 1.],
...,
[1., 0.],
[0., 1.],
[1., 0.]], dtype=float32)
1.2.3 Validation Data
18m 1s completed at 12:43 AM
Type here to search
12:58 AM
12-05-2021

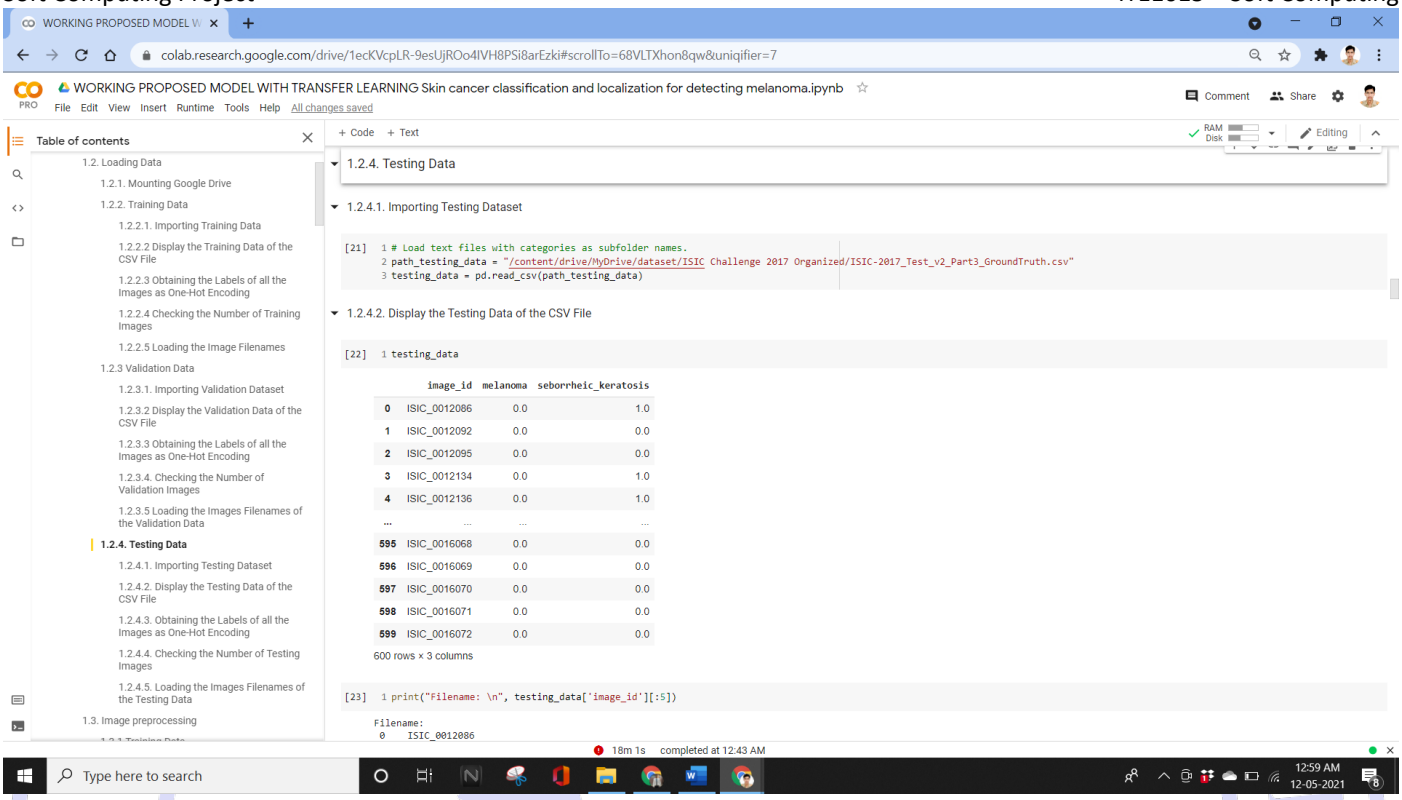

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=-cFWzjcxY4zT&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
[12] 1 # Load text files with categories as subfolder names.
2 path_validation_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Part3_GroundTruth.csv"
3 validation_data = pd.read_csv(path_validation_data)
1.2.3.2 Display the Validation Data of the CSV File
[13] 1 validation_data
image_id melanoma seborrheic_keratosis
0 ISIC_0001769 0.0 0.0
1 ISIC_0001852 0.0 0.0
2 ISIC_0001871 0.0 0.0
3 ISIC_0003462 0.0 0.0
4 ISIC_0003539 0.0 0.0
... ...
145 ISIC_0015443 0.0 0.0
146 ISIC_0015445 0.0 0.0
147 ISIC_0015483 0.0 0.0
148 ISIC_0015496 0.0 0.0
149 ISIC_0015627 0.0 0.0
150 rows × 3 columns
[14] 1 print("Filename: \n", validation_data['image_id'][:5])
Filename:
18m 1s completed at 12:43 AM
Type here to search
12:58 AM
12-05-2021

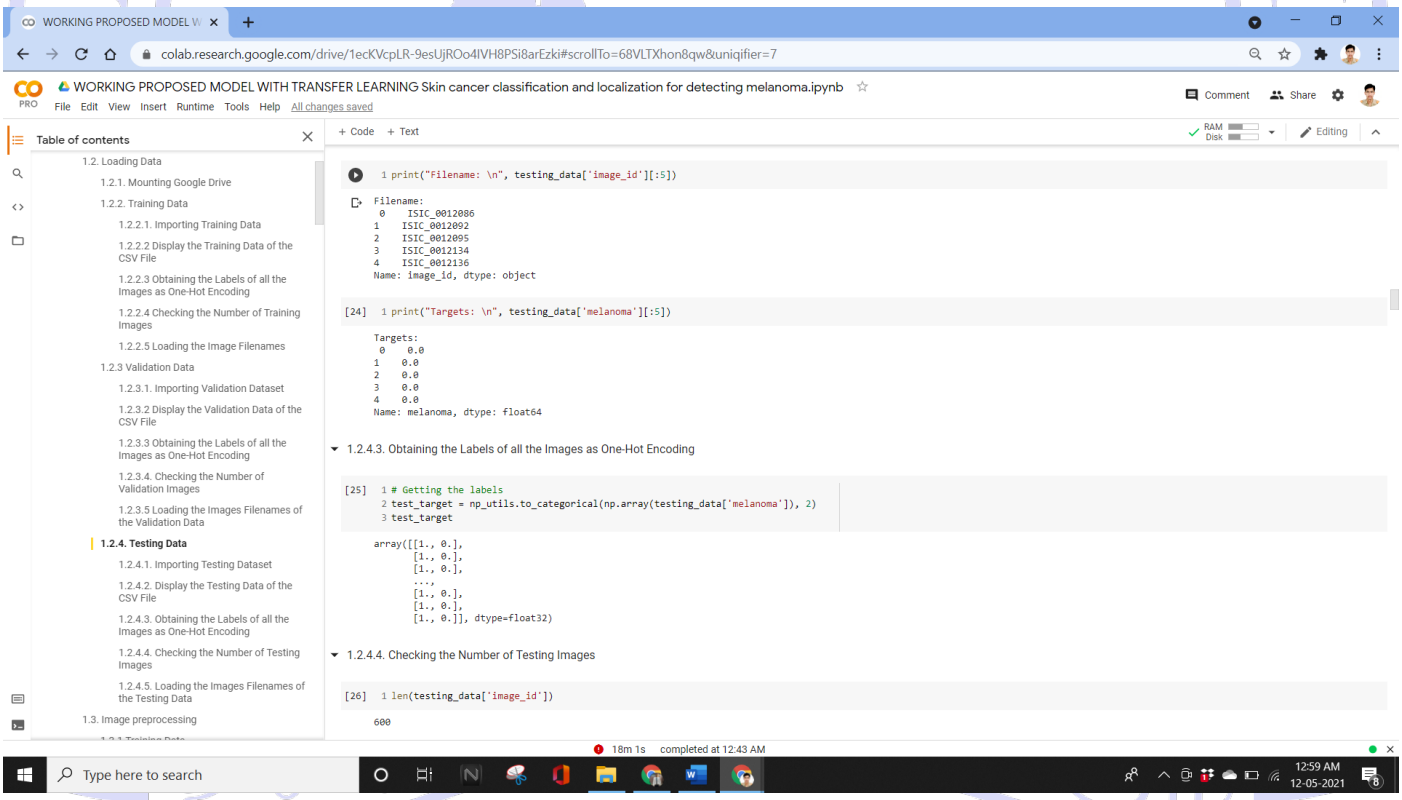

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=-cFWzjcxY4zT&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1 print("Filename: \n", validation_data['image_id'][:5])
Filename:
0 ISIC_0001769
1 ISIC_0001852
2 ISIC_0001871
3 ISIC_0003462
4 ISIC_0003539
Name: image_id, dtype: object
[15] 1 print("Targets: \n", validation_data['melanoma'][:5])
Targets:
0 0.0
1 0.0
2 0.0
3 0.0
4 0.0
Name: melanoma, dtype: float64
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
[16] 1 # Getting the labels
2 target = np_utils.to_categorical(np.array(validation_data['melanoma']), 2)
3 target
array([[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
18m 1s completed at 12:43 AM
Type here to search
12:58 AM
12-05-2021

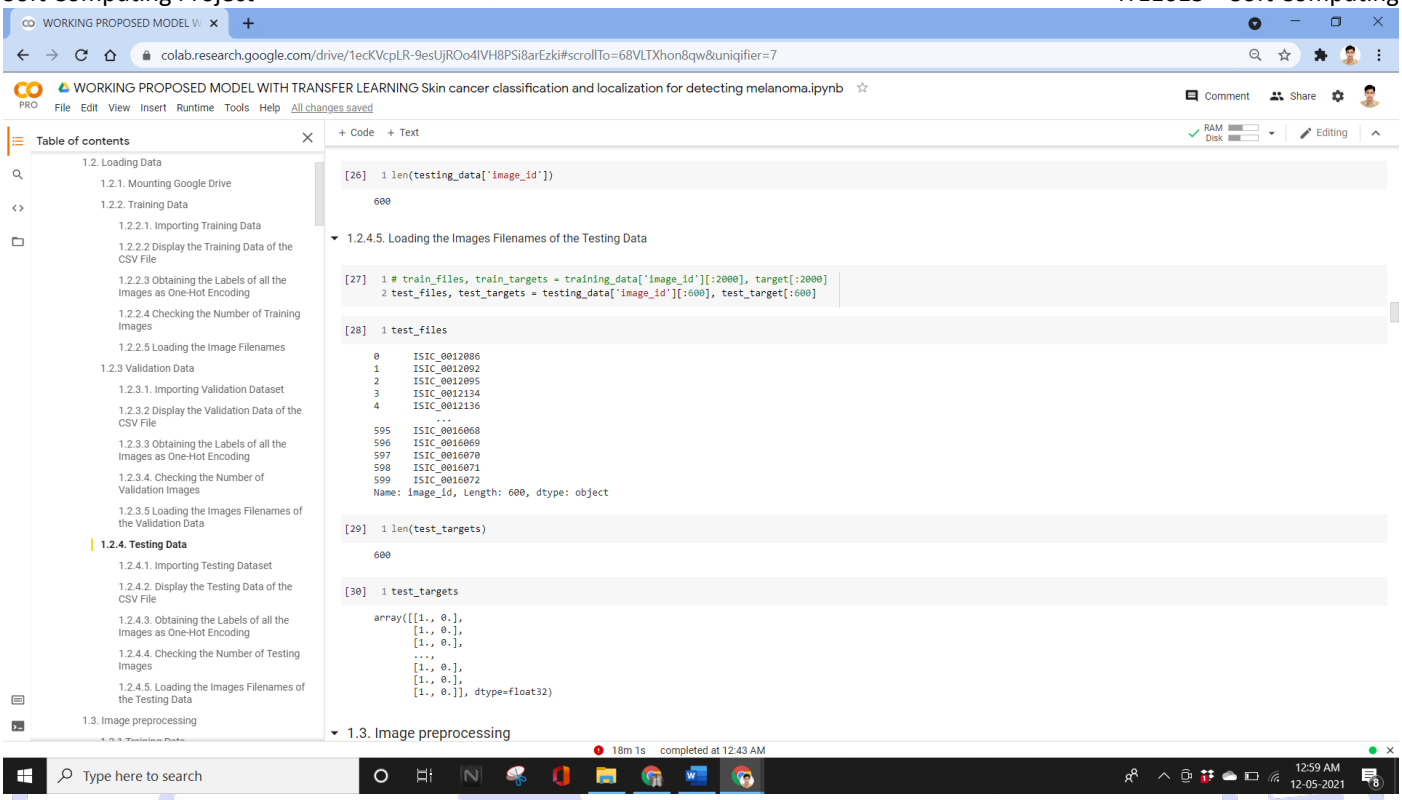

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=-cFWzjcxY4zT&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
18m 1s completed at 12:43 AM
Type here to search
12:58 AM
12-05-2021

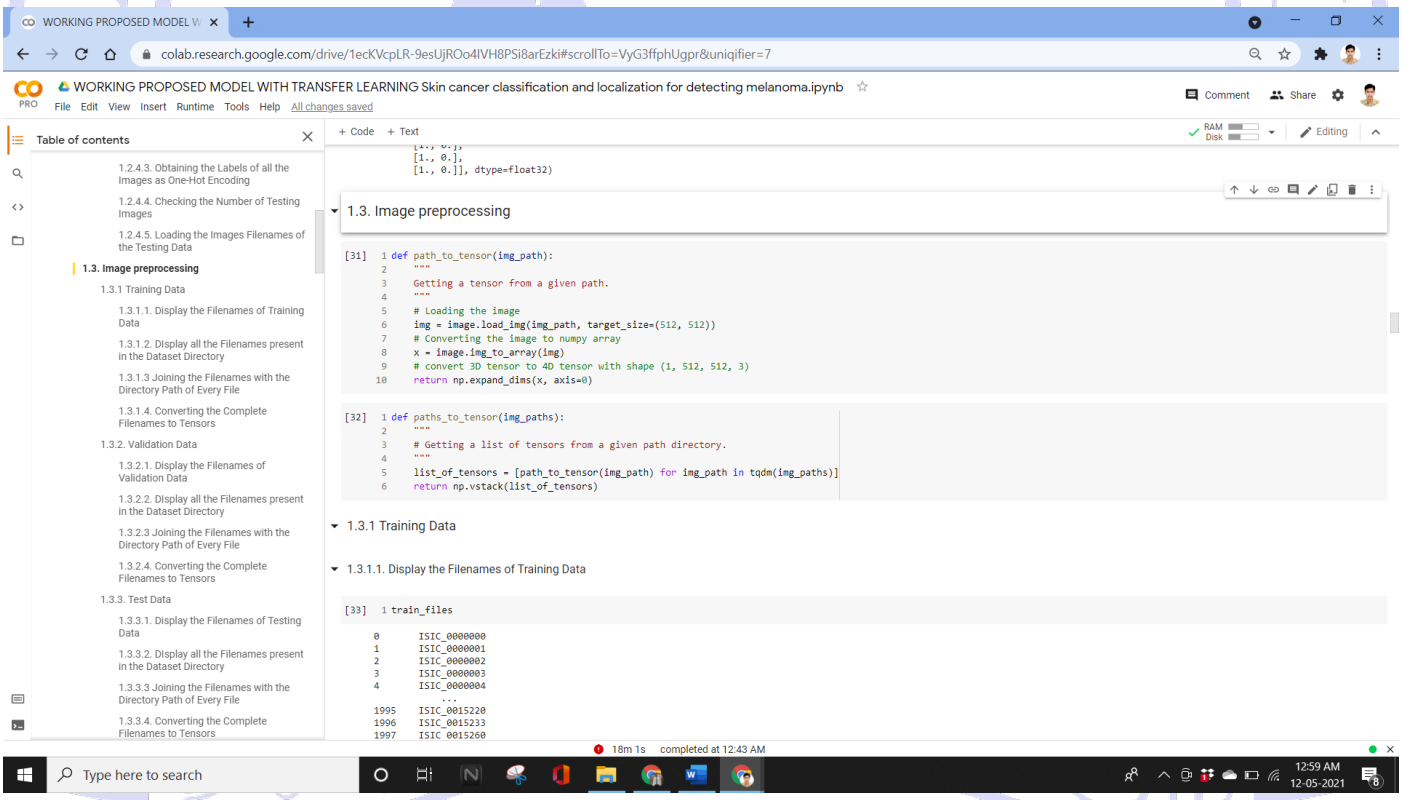

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4lVH8PSi8arEzki#scrollTo=S8T6qAg0ueYa&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
[1., 0.],
1.2.3.4. Checking the Number of Validation Images
[17] 1 len(validation_data['image_id'])
150
1.2.3.5 Loading the Images Filenames of the Validation Data
[18] 1 # train_files, train_targets = training_data['image_id'][:2000], target[:2000]
2 valid_files, valid_targets = validation_data['image_id'][:150], target[:150]
[19] 1 valid_files
0 ISIC_0001769
1 ISIC_0001852
2 ISIC_0001871
3 ISIC_0003462
4 ISIC_0003539
...
145 ISIC_0015443
146 ISIC_0015445
147 ISIC_0015483
148 ISIC_0015496
149 ISIC_0015627
Name: image_id, Length: 150, dtype: object
[20] 1 valid_targets
array([[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

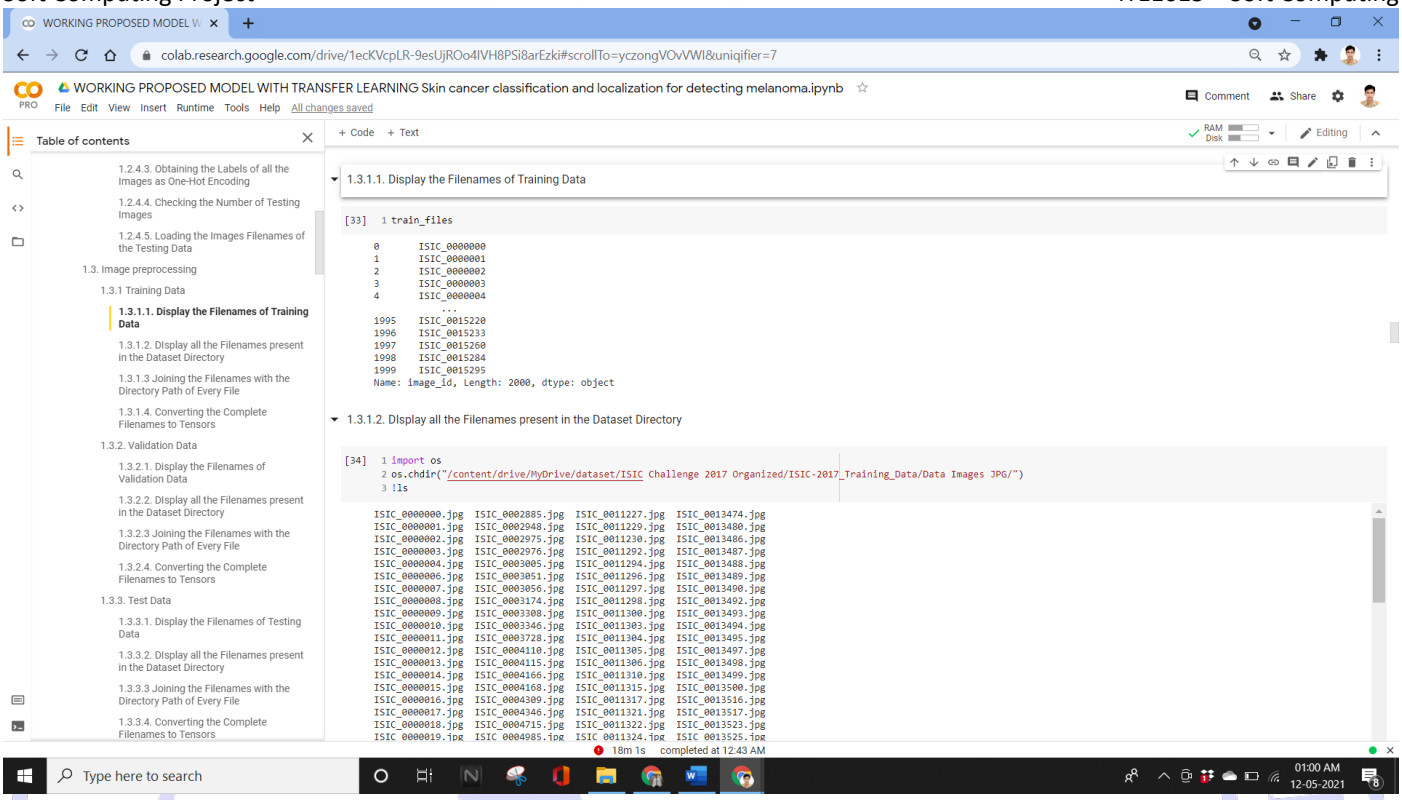

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4lVH8PSi8arEzki#scrollTo=S8T6qAg0ueYa&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
array([[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
[0., 1.],
[1., 0.],
[1., 0.],
[1., 0.],
[1., 0.],
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

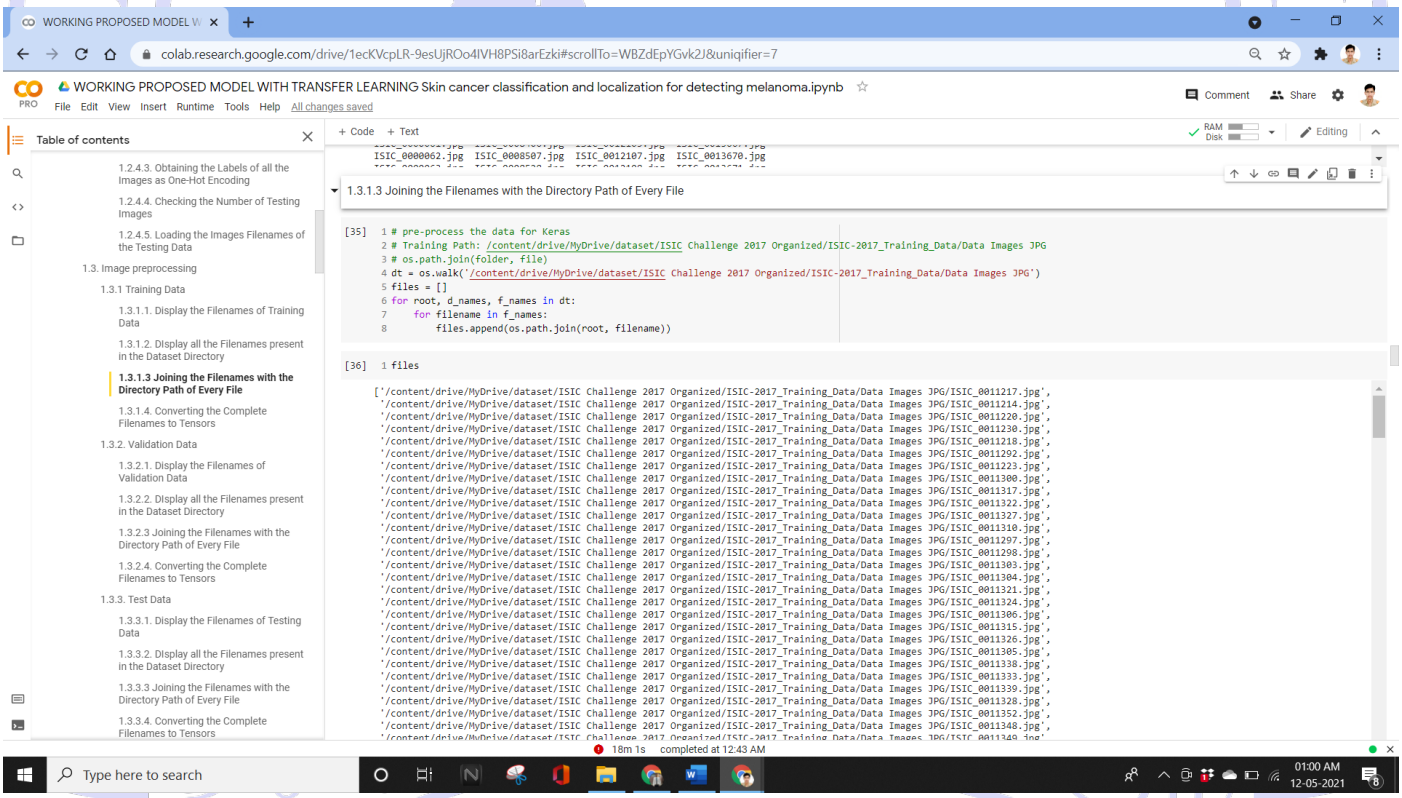

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=68VLTXhon8qw&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
[21] 1 # Load text files with categories as subfolder names.
2 path_testing_data = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Part3_GroundTruth.csv"
3 testing_data = pd.read_csv(path_testing_data)
1.2.4.2. Display the Testing Data of the CSV File
[22] 1 testing_data
image_id melanoma seborrheic_keratosis
0 ISIC_0012086 0.0 1.0
1 ISIC_0012092 0.0 0.0
2 ISIC_0012095 0.0 0.0
3 ISIC_0012134 0.0 1.0
4 ISIC_0012136 0.0 1.0
595 ISIC_0016068 0.0 0.0
596 ISIC_0016069 0.0 0.0
597 ISIC_0016070 0.0 0.0
598 ISIC_0016071 0.0 0.0
599 ISIC_0016072 0.0 0.0
600 rows × 3 columns
[23] 1 print("Filename: \n", testing_data['image_id'][:5])
Filename:
0 ISIC_0012086
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

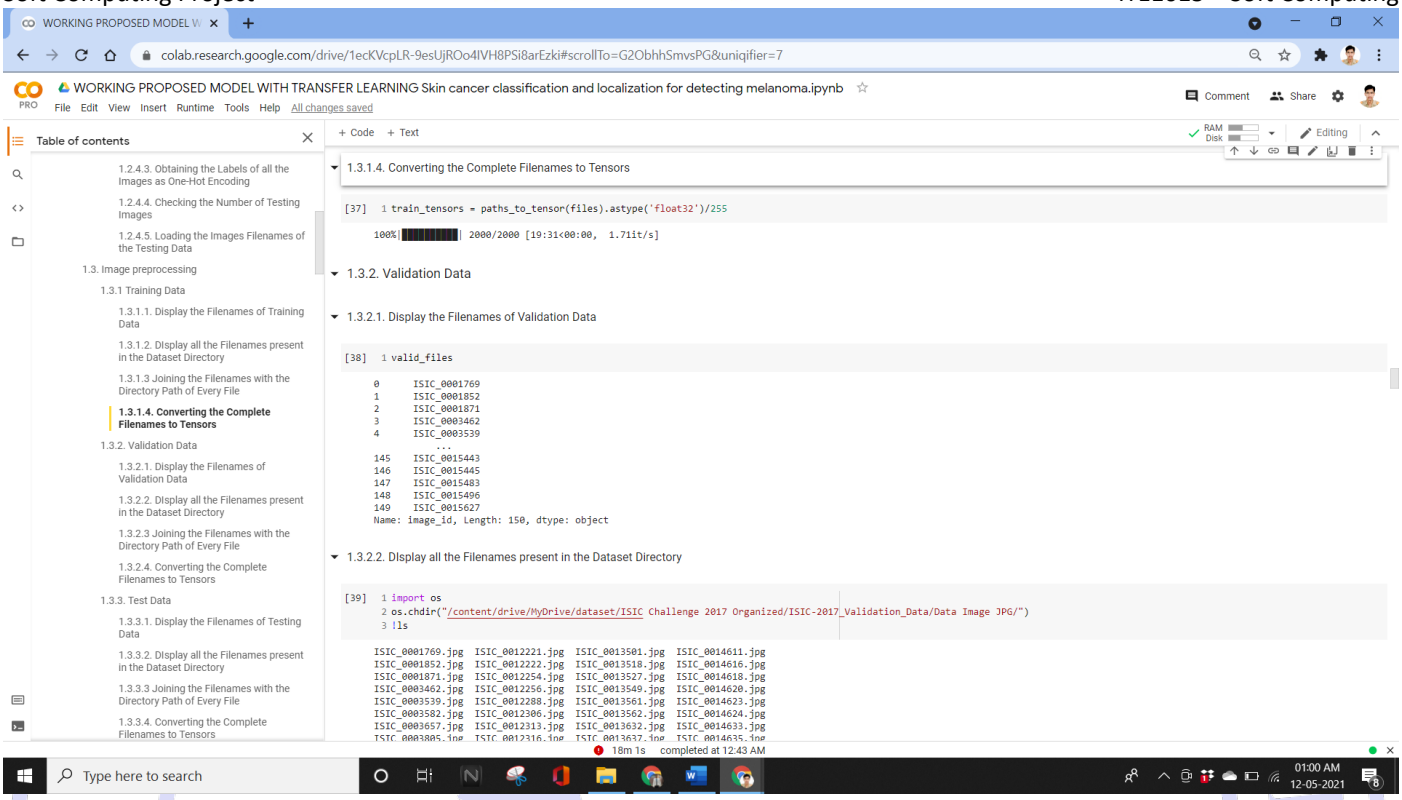

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=68VLTXhon8qw&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
1 print("Filename: \n", testing_data['image_id'][:5])
Filename:
0 ISIC_0012086
1 ISIC_0012092
2 ISIC_0012095
3 ISIC_0012134
4 ISIC_0012136
Name: image_id, dtype: object
[24] 1 print("Targets: \n", testing_data['melanoma'][:5])
Targets:
0 0.0
1 0.0
2 0.0
3 0.0
4 0.0
Name: melanoma, dtype: float64
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
[25] 1 # Getting the labels
2 test_target = np_utils.to_categorical(np.array(testing_data['melanoma']), 2)
3 test_target
array([[1., 0.],
[1., 0.],
[1., 0.],
...,
[1., 0.],
[1., 0.],
[1., 0.]], dtype=float32)
1.2.4.4. Checking the Number of Testing Images
[26] 1 len(testing_data['image_id'])
600
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

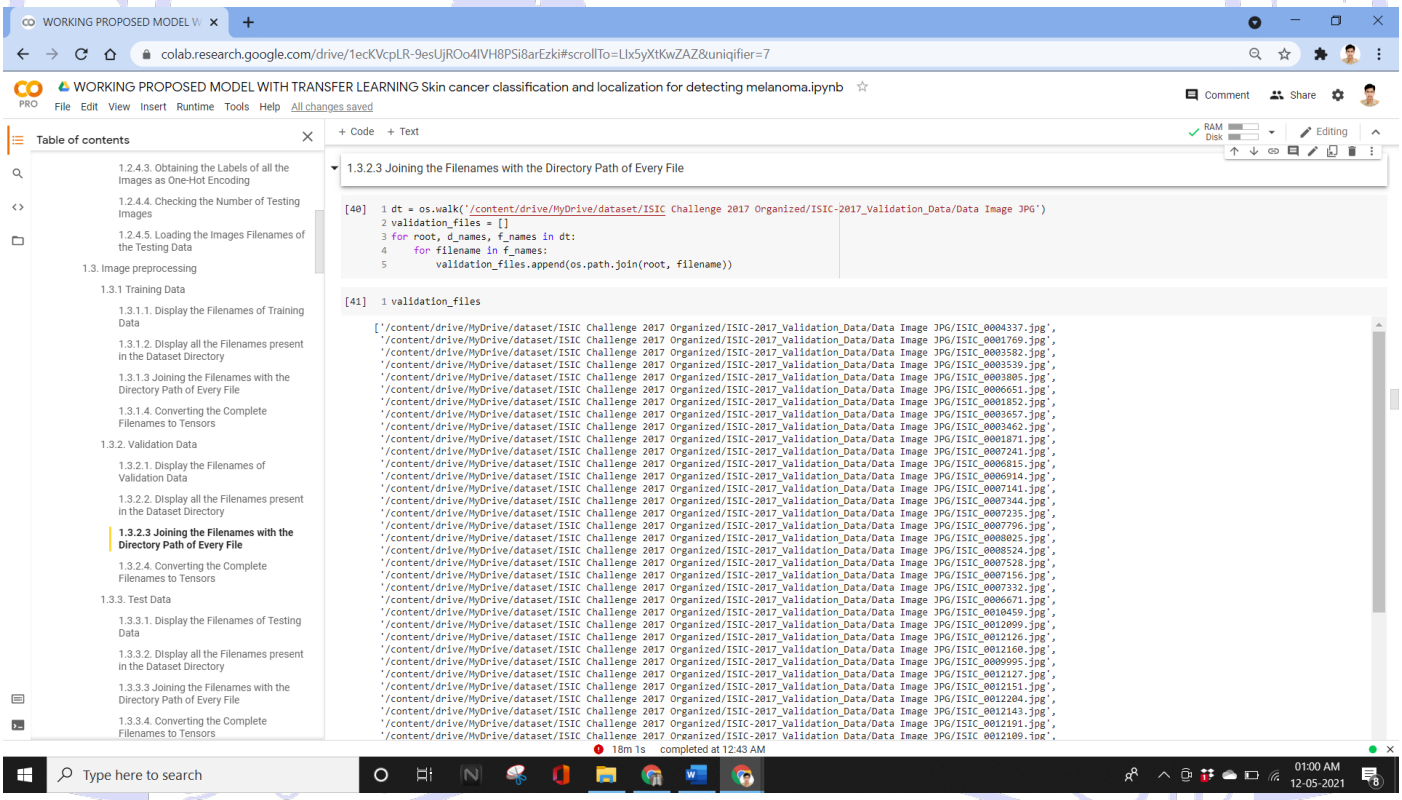

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=68VLTXhon8qw&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2. Loading Data
1.2.1. Mounting Google Drive
1.2.2. Training Data
1.2.2.1. Importing Training Data
1.2.2.2 Display the Training Data of the CSV File
1.2.2.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.2.4 Checking the Number of Training Images
1.2.2.5 Loading the Image Filenames
1.2.3 Validation Data
1.2.3.1. Importing Validation Dataset
1.2.3.2 Display the Validation Data of the CSV File
1.2.3.3 Obtaining the Labels of all the Images as One-Hot Encoding
1.2.3.4. Checking the Number of Validation Images
1.2.3.5 Loading the Images Filenames of the Validation Data
1.2.4. Testing Data
1.2.4.1. Importing Testing Dataset
1.2.4.2. Display the Testing Data of the CSV File
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
+ Code + Text
RAM Disk
Editing
[26] 1 len(testing_data['image_id'])
600
1.2.4.5. Loading the Images Filenames of the Testing Data
[27] 1 # train_files, train_targets = training_data['image_id'][:2000], target[:2000]
2 test_files, test_targets = testing_data['image_id'][:600], test_target[:600]
[28] 1 test_files
0 ISIC_0012086
1 ISIC_0012092
2 ISIC_0012095
3 ISIC_0012134
4 ISIC_0012136
...
595 ISIC_0016068
596 ISIC_0016069
597 ISIC_0016070
598 ISIC_0016071
599 ISIC_0016072
Name: image_id, Length: 600, dtype: object
[29] 1 len(test_targets)
600
[30] 1 test_targets
array([[1., 0.],
[1., 0.],
[1., 0.],
...,
[1., 0.],
[1., 0.],
[1., 0.]], dtype=float32)
1.3. Image preprocessing
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

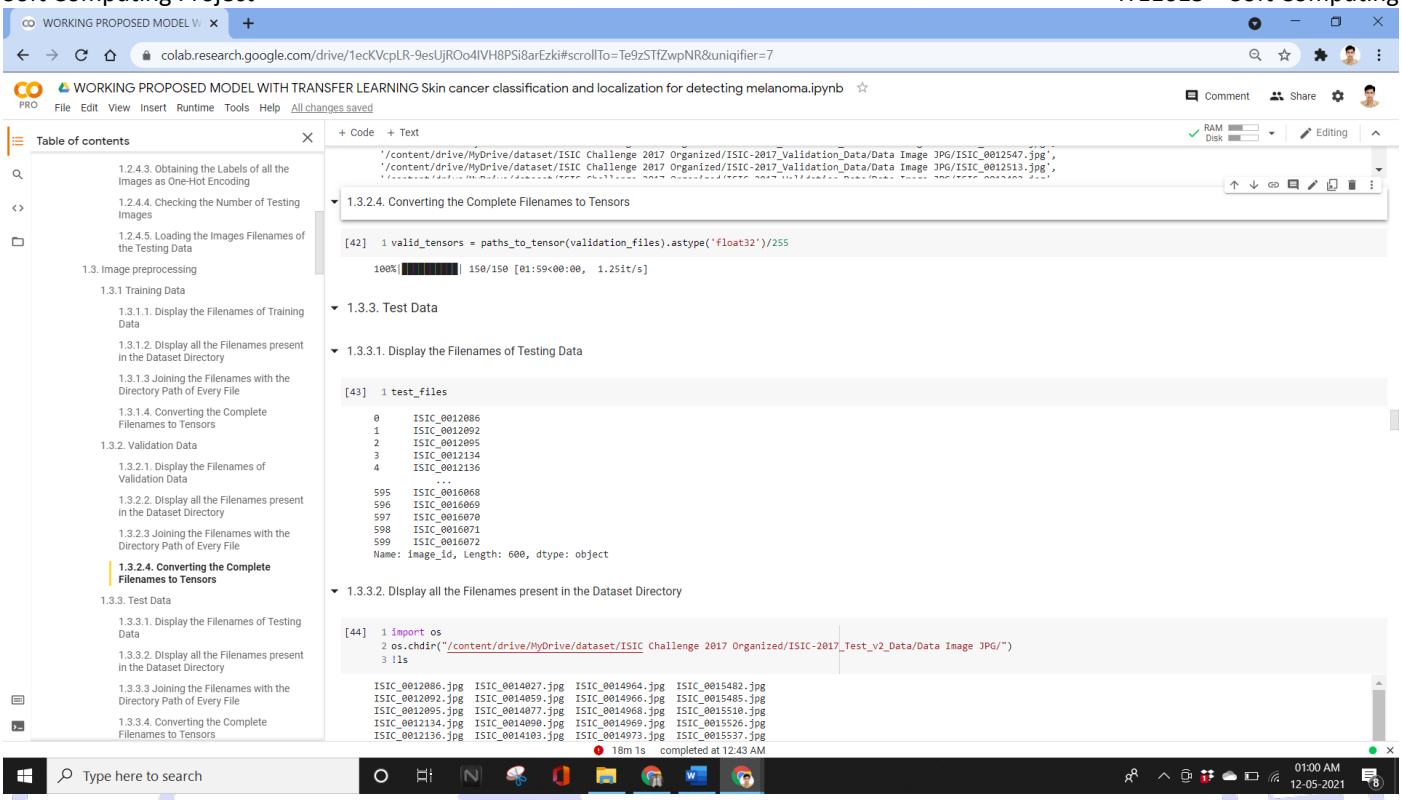

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=VyG3ffphUgpr&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
1.3.1.2. Display all the Filenames present in the Dataset Directory
1.3.1.3 Joining the Filenames with the Directory Path of Every File
1.3.1.4. Converting the Complete Filenames to Tensors
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
+ Code + Text
RAM Disk
Editing
[1., 0.],
[1., 0.]], dtype=float32)
1.3. Image preprocessing
[31] 1 def path_to_tensor(img_path):
2 """
3 Getting a tensor from a given path.
4 """
5 # Loading the image
6 img = image.load_img(img_path, target_size=(512, 512))
7 # Converting the image to numpy array
8 x = image.img_to_array(img)
9 # convert 3D tensor to 4D tensor with shape (1, 512, 512, 3)
10 return np.expand_dims(x, axis=0)
[32] 1 def paths_to_tensor(img_paths):
2 """
3 # Getting a list of tensors from a given path directory.
4 """
5 list_of_tensors = [path_to_tensor(img_path) for img_path in tqdm(img_paths)]
6 return np.vstack(list_of_tensors)
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
[33] 1 train_files
0 ISIC_0000000
1 ISIC_0000001
2 ISIC_0000002
3 ISIC_0000003
4 ISIC_0000004
...
1995 ISIC_0015220
1996 ISIC_0015233
1997 ISIC_0015260
18m 1s completed at 12:43 AM
Type here to search
12:59 AM
12-05-2021

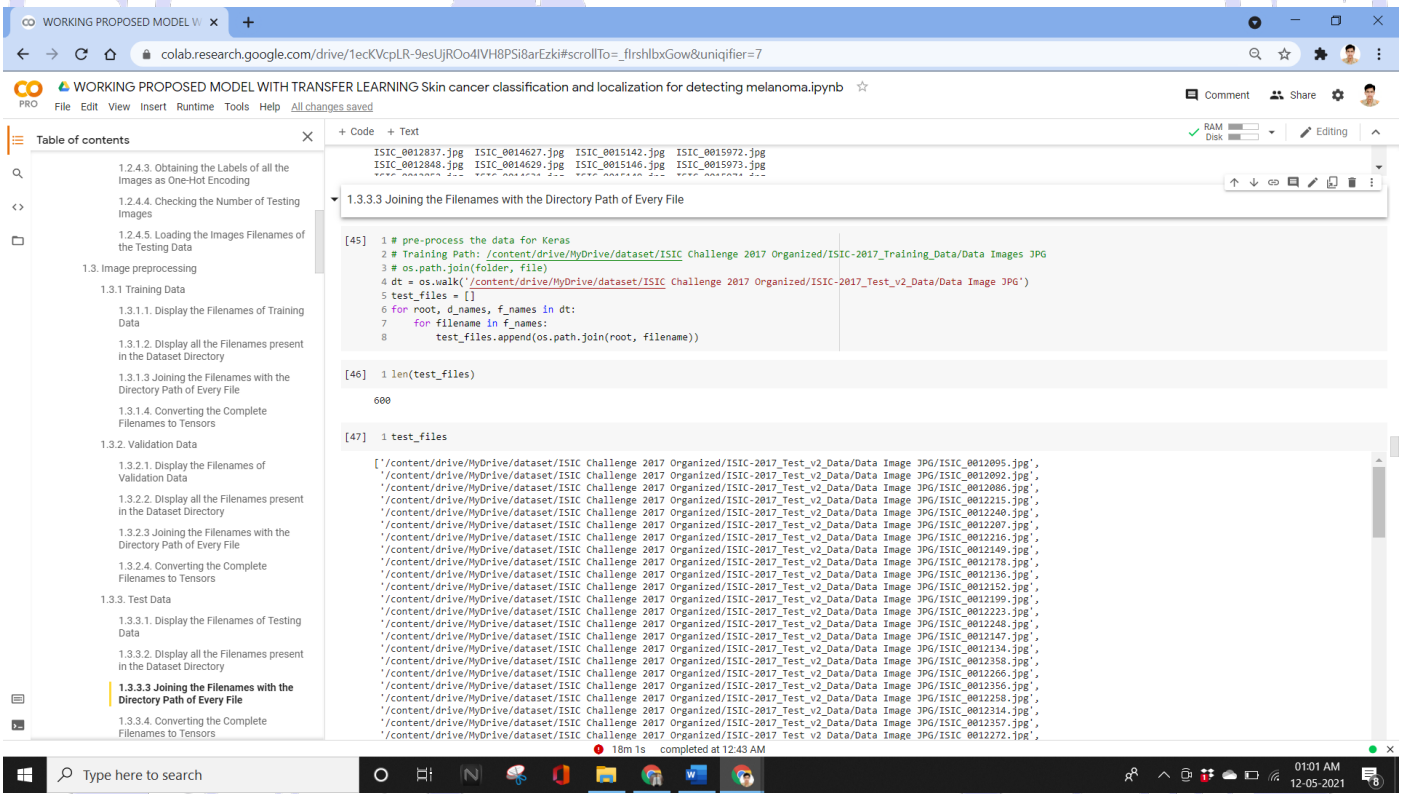

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=yczongVOvVWI&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
1.3.1.1. Display the Filenames of Training Data
[33] 1 train_files
0 ISIC_0000000
1 ISIC_0000001
2 ISIC_0000002
3 ISIC_0000003
4 ISIC_0000004
...
1995 ISIC_0015220
1996 ISIC_0015233
1997 ISIC_0015260
1998 ISIC_0015284
1999 ISIC_0015295
Name: image_id, Length: 2000, dtype: object
1.3.1.2. DIsplay all the Filenames present in the Dataset Directory
[34] 1 import os
2 os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/")
3 !ls
ISIC_0000000.jpg ISIC_0002885.jpg ISIC_0011227.jpg ISIC_0013474.jpg
ISIC_0000001.jpg ISIC_0002948.jpg ISIC_0011229.jpg ISIC_0013480.jpg
ISIC_0000002.jpg ISIC_0002975.jpg ISIC_0011230.jpg ISIC_0013486.jpg
ISIC_0000003.jpg ISIC_0002976.jpg ISIC_0011292.jpg ISIC_0013487.jpg
ISIC_0000004.jpg ISIC_0003005.jpg ISIC_0011294.jpg ISIC_0013488.jpg
ISIC_0000006.jpg ISIC_0003051.jpg ISIC_0011296.jpg ISIC_0013489.jpg
ISIC_0000007.jpg ISIC_0003056.jpg ISIC_0011297.jpg ISIC_0013490.jpg
ISIC_0000008.jpg ISIC_0003174.jpg ISIC_0011298.jpg ISIC_0013492.jpg
ISIC_0000009.jpg ISIC_0003308.jpg ISIC_0011300.jpg ISIC_0013493.jpg
ISIC_0000010.jpg ISIC_0003346.jpg ISIC_0011303.jpg ISIC_0013494.jpg
ISIC_0000011.jpg ISIC_0003728.jpg ISIC_0011304.jpg ISIC_0013495.jpg
ISIC_0000012.jpg ISIC_0004110.jpg ISIC_0011305.jpg ISIC_0013497.jpg
ISIC_0000013.jpg ISIC_0004115.jpg ISIC_0011306.jpg ISIC_0013498.jpg
ISIC_0000014.jpg ISIC_0004166.jpg ISIC_0011310.jpg ISIC_0013499.jpg
ISIC_0000015.jpg ISIC_0004168.jpg ISIC_0011315.jpg ISIC_0013500.jpg
ISIC_0000016.jpg ISIC_0004309.jpg ISIC_0011317.jpg ISIC_0013516.jpg
ISIC_0000017.jpg ISIC_0004346.jpg ISIC_0011321.jpg ISIC_0013517.jpg
ISIC_0000018.jpg ISIC_0004715.jpg ISIC_0011322.jpg ISIC_0013523.jpg
ISIC_0000019.jpg ISIC_0004985.jpg ISIC_0011324.jpg ISIC_0013525.jpg
18m 1s completed at 12:43 AM
Type here to search
01:00 AM
12-05-2021

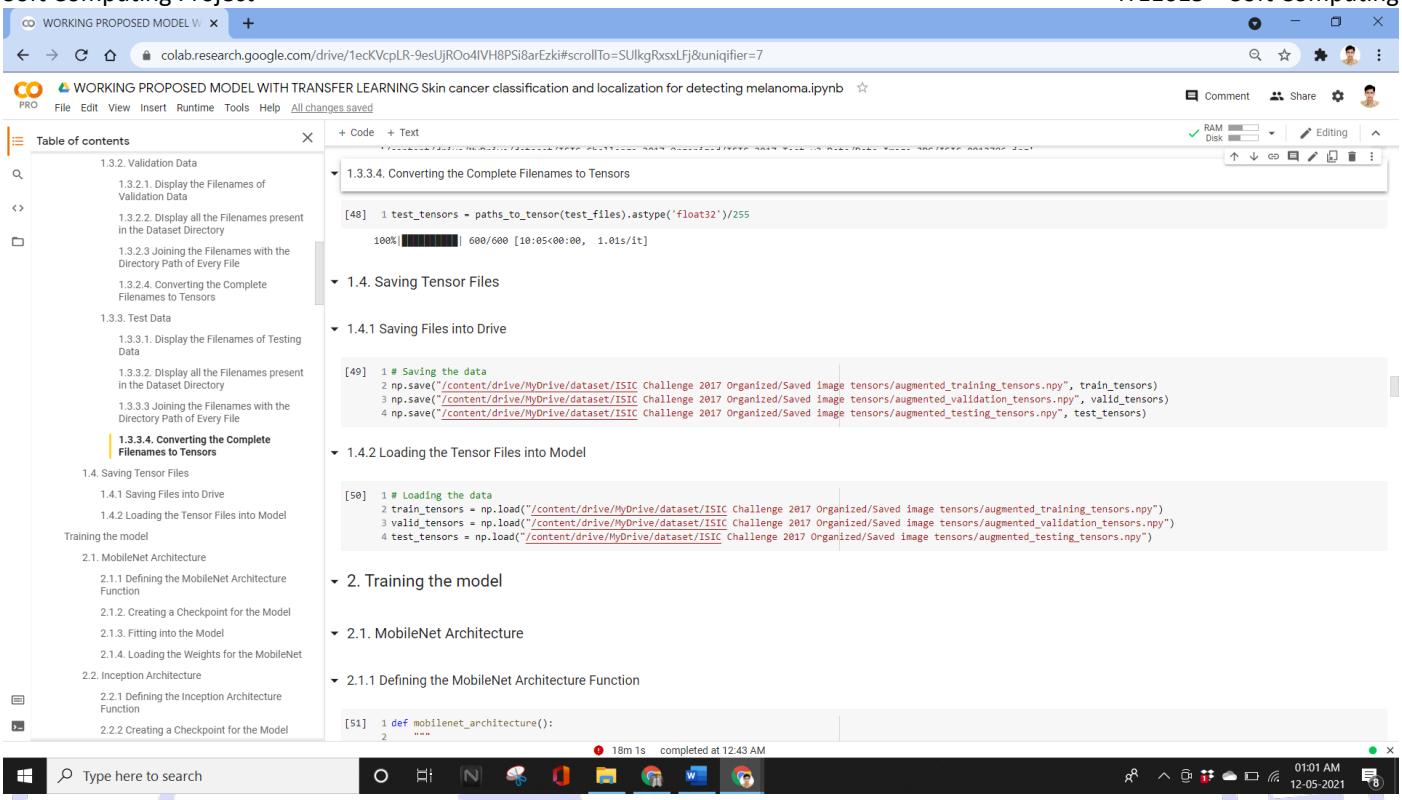

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=WBZdEpYGvk2J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
ISIC_0000062.jpg ISIC_0008507.jpg ISIC_0012107.jpg ISIC_0013670.jpg
1.3.1.3 Joining the Filenames with the Directory Path of Every File
[35] 1 # pre-process the data for Keras
2 # Training Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG
3 # os.path.join(folder, file)
4 dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG')
5 files = []
6 for root, d_names, f_names in dt:
7 for filename in f_names:
8 files.append(os.path.join(root, filename))
[36] 1 files
['/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011217.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011214.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011220.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011230.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011218.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011292.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011223.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011300.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011317.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011322.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011327.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011310.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011297.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011298.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011303.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011304.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011321.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011324.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011306.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011315.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011326.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011305.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011333.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011339.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011328.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011352.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG/ISIC_0011348.jpg',
18m 1s completed at 12:43 AM
Type here to search
01:00 AM
12-05-2021

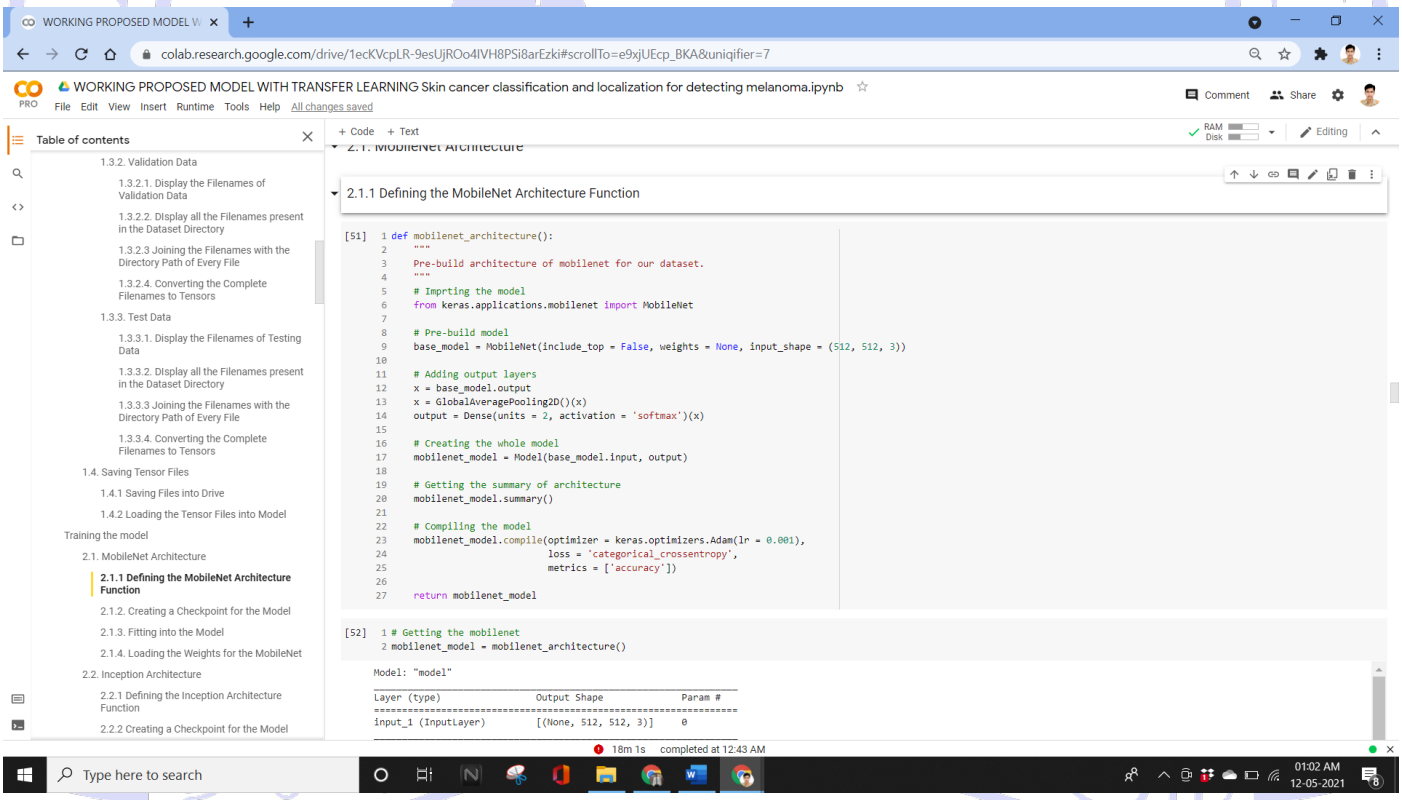

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=G2ObhhSmvsPG&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
1.3.1.2. Display all the Filenames present in the Dataset Directory
1.3.1.3 Joining the Filenames with the Directory Path of Every File
1.3.1.4. Converting the Complete Filenames to Tensors
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
+ Code + Text
RAM Disk
Editing
1.3.1.4. Converting the Complete Filenames to Tensors
[37] 1 train_tensors = paths_to_tensor(files).astype('float32')/255
100%| 2000/2000 [19:31<00:00, 1.71it/s]
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
[38] 1 valid_files
0 ISIC_0001769
1 ISIC_0001852
2 ISIC_0001871
3 ISIC_0003462
4 ISIC_0003539
...
145 ISIC_0015443
146 ISIC_0015445
147 ISIC_0015483
148 ISIC_0015496
149 ISIC_0015627
Name: image_id, Length: 150, dtype: object
1.3.2.2. Display all the Filenames present in the Dataset Directory
[39] 1 import os
2 os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/")
3 !ls
ISIC_0001769.jpg ISIC_0012221.jpg ISIC_0013501.jpg ISIC_0014611.jpg
ISIC_0001852.jpg ISIC_0012222.jpg ISIC_0013518.jpg ISIC_0014616.jpg
ISIC_0001871.jpg ISIC_0012254.jpg ISIC_0013527.jpg ISIC_0014618.jpg
ISIC_0003462.jpg ISIC_0012256.jpg ISIC_0013549.jpg ISIC_0014620.jpg
ISIC_0003539.jpg ISIC_0012288.jpg ISIC_0013561.jpg ISIC_0014623.jpg
ISIC_0003582.jpg ISIC_0012306.jpg ISIC_0013562.jpg ISIC_0014624.jpg
ISIC_0003657.jpg ISIC_0012313.jpg ISIC_0013632.jpg ISIC_0014633.jpg
18m 1s completed at 12:43 AM
Type here to search
01:00 AM
12-05-2021

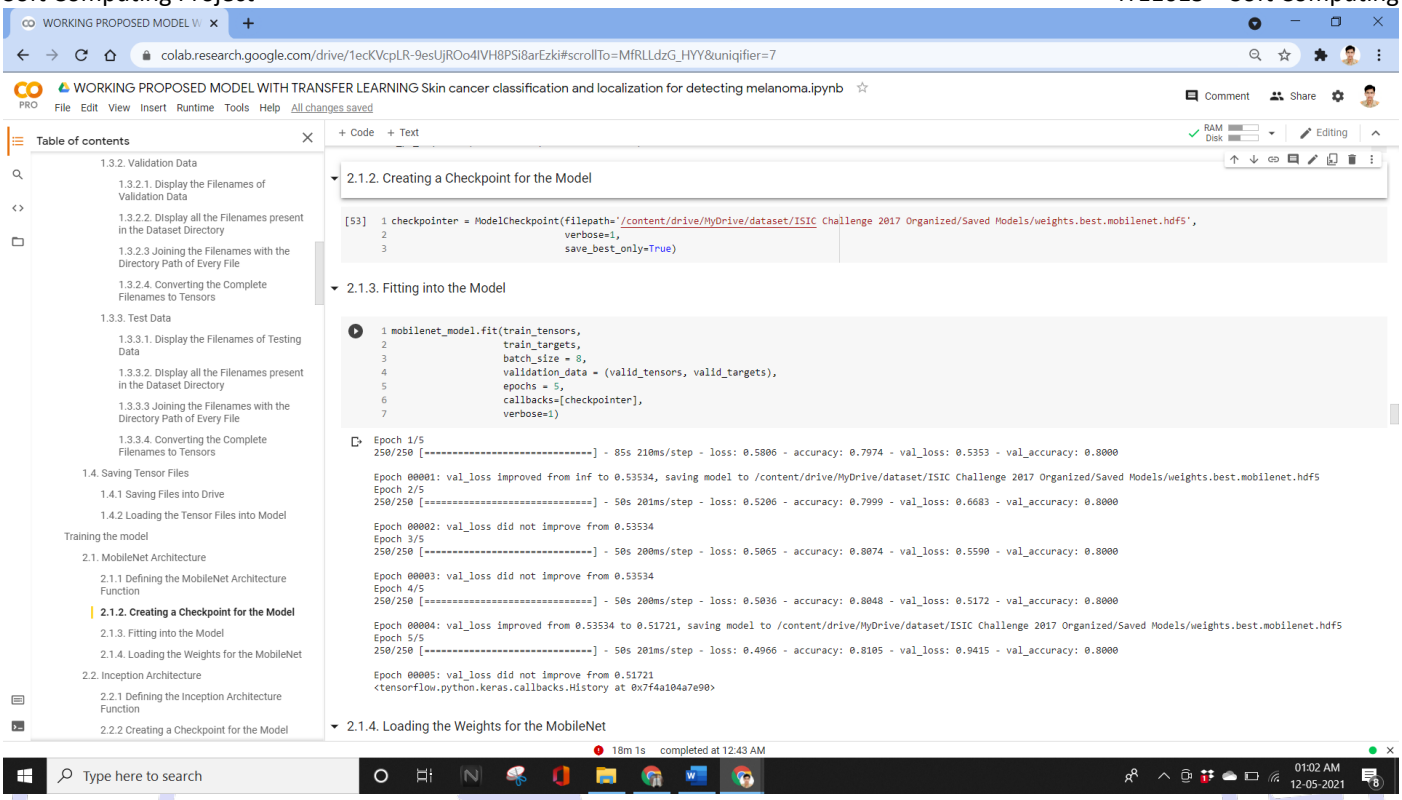

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=LIx5yXtKwZAZ&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
1.3.1.2. Display all the Filenames present in the Dataset Directory
1.3.1.3 Joining the Filenames with the Directory Path of Every File
1.3.1.4. Converting the Complete Filenames to Tensors
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
+ Code + Text
RAM Disk
Editing
1.3.2.3 Joining the Filenames with the Directory Path of Every File
[40] 1 dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG')
2 validation_files = []
3 for root, d_names, f_names in dt:
4 for filename in f_names:
5 validation_files.append(os.path.join(root, filename))
[41] 1 validation_files
['/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0004337.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003582.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003539.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003805.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0006651.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001852.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003657.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003462.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001871.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007241.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0006815.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0006914.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007141.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007344.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007235.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007796.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0008025.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0008524.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007528.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007156.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0007332.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0006671.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0010459.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012099.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012126.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012160.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0009995.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012127.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012151.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012204.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012143.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012191.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012109.jpg',
18m 1s completed at 12:43 AM
Type here to search
01:00 AM
12-05-2021

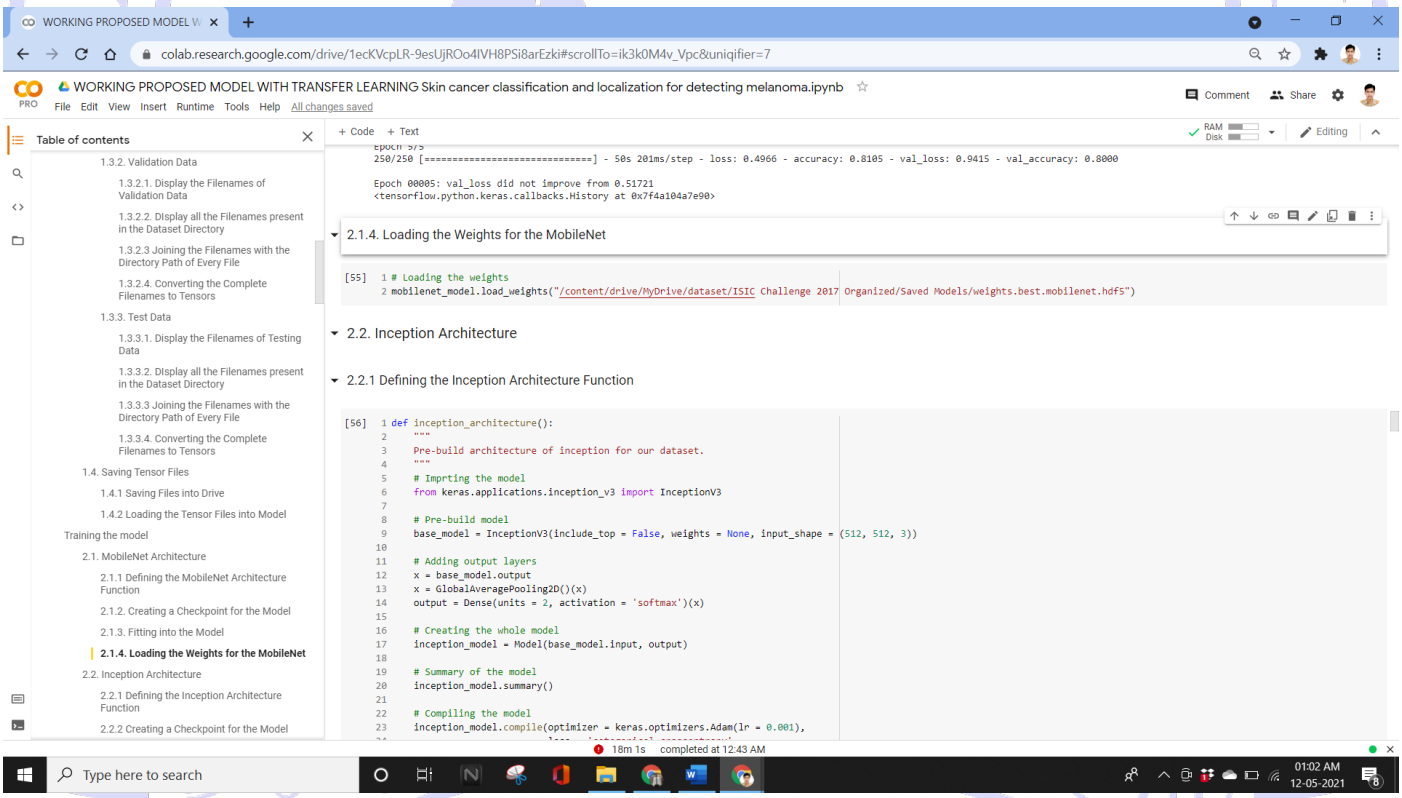

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=Te9zSTfZwpNR&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
1.3.1.2. Display all the Filenames present in the Dataset Directory
1.3.1.3 Joining the Filenames with the Directory Path of Every File
1.3.1.4. Converting the Complete Filenames to Tensors
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
+ Code + Text
RAM Disk
Editing
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012547.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0012513.jpg',
1.3.2.4. Converting the Complete Filenames to Tensors
[42] 1 valid_tensors = paths_to_tensor(validation_files).astype('float32')/255
100% | 150/150 [01:59<00:00, 1.25it/s]
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
[43] 1 test_files
0 ISIC_0012086
1 ISIC_0012092
2 ISIC_0012095
3 ISIC_0012134
4 ISIC_0012136
...
595 ISIC_0016068
596 ISIC_0016069
597 ISIC_0016070
598 ISIC_0016071
599 ISIC_0016072
Name: image_id, Length: 600, dtype: object
1.3.3.2. Display all the Filenames present in the Dataset Directory
[44] 1 import os
2 os.chdir("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/")
3 !ls
ISIC_0012086.jpg ISIC_0014027.jpg ISIC_0014964.jpg ISIC_0015482.jpg
ISIC_0012092.jpg ISIC_0014059.jpg ISIC_0014966.jpg ISIC_0015485.jpg
ISIC_0012095.jpg ISIC_0014077.jpg ISIC_0014968.jpg ISIC_0015510.jpg
ISIC_0012134.jpg ISIC_0014090.jpg ISIC_0014969.jpg ISIC_0015526.jpg
ISIC_0012136.jpg ISIC_0014103.jpg ISIC_0014973.jpg ISIC_0015537.jpg
18m 1s completed at 12:43 AM
Type here to search
01:00 AM
12-05-2021

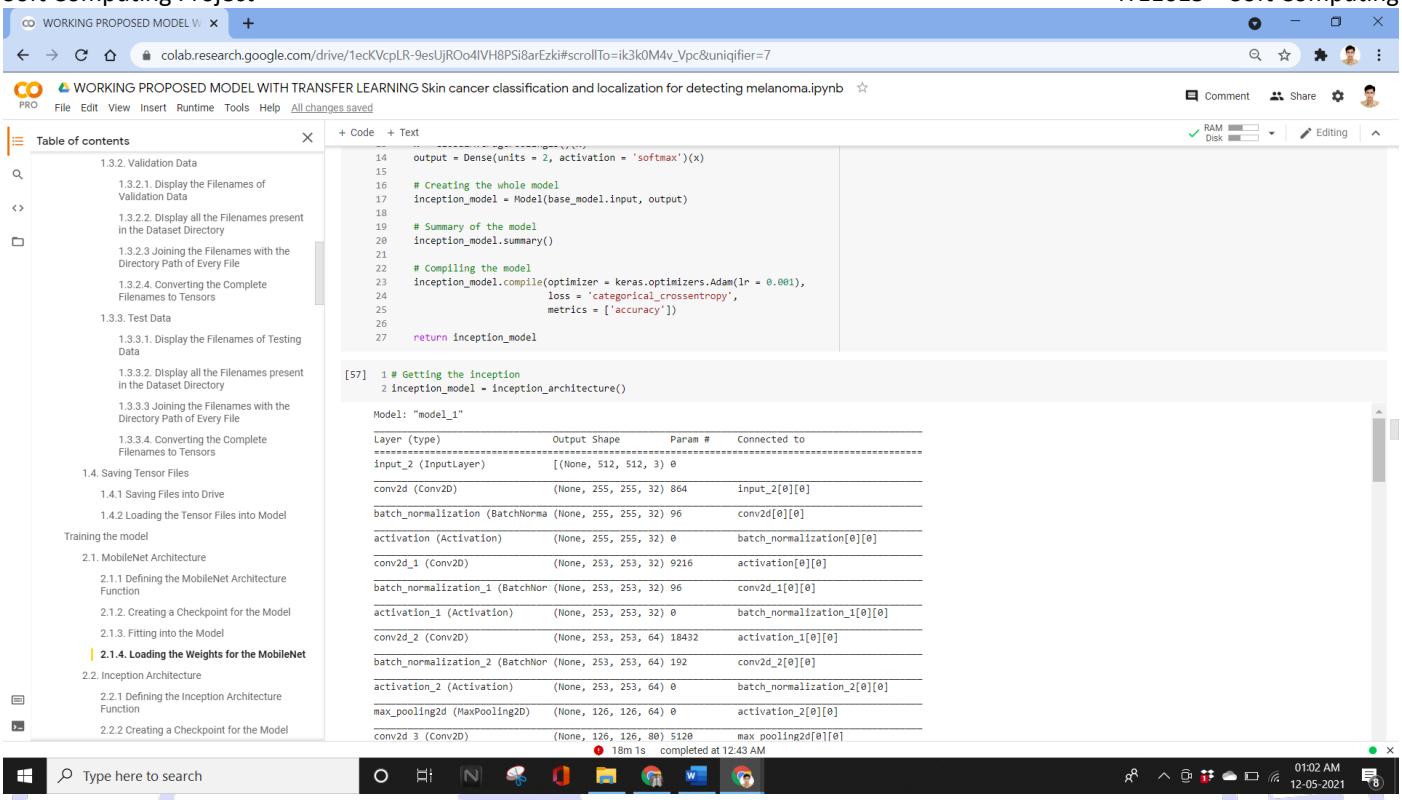

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=_flrshlbxGow&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.2.4.3. Obtaining the Labels of all the Images as One-Hot Encoding
1.2.4.4. Checking the Number of Testing Images
1.2.4.5. Loading the Images Filenames of the Testing Data
1.3. Image preprocessing
1.3.1 Training Data
1.3.1.1. Display the Filenames of Training Data
1.3.1.2. Display all the Filenames present in the Dataset Directory
1.3.1.3 Joining the Filenames with the Directory Path of Every File
1.3.1.4. Converting the Complete Filenames to Tensors
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
+ Code + Text
RAM Disk
Editing
ISIC_0012837.jpg ISIC_0014627.jpg ISIC_0015142.jpg ISIC_0015972.jpg
ISIC_0012848.jpg ISIC_0014629.jpg ISIC_0015146.jpg ISIC_0015973.jpg
1.3.3.3 Joining the Filenames with the Directory Path of Every File
[45] 1 # pre-process the data for Keras
2 # Training Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Training_Data/Data Images JPG
3 # os.path.join(folder, file)
4 dt = os.walk('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG')
5 test_files = []
6 for root, d_names, f_names in dt:
7 for filename in f_names:
8 test_files.append(os.path.join(root, filename))
[46] 1 len(test_files)
600
[47] 1 test_files
['/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012095.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012092.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012086.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012215.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012240.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012207.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012216.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012149.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012178.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012136.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012152.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012199.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012223.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012248.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012147.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012134.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012358.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012266.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012356.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012258.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012314.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012357.jpg',
'/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017 Test v2 Data/Data Image JPG/ISIC_0012272.jpg',
18m 1s completed at 12:43 AM
Type here to search
01:01 AM
12-05-2021

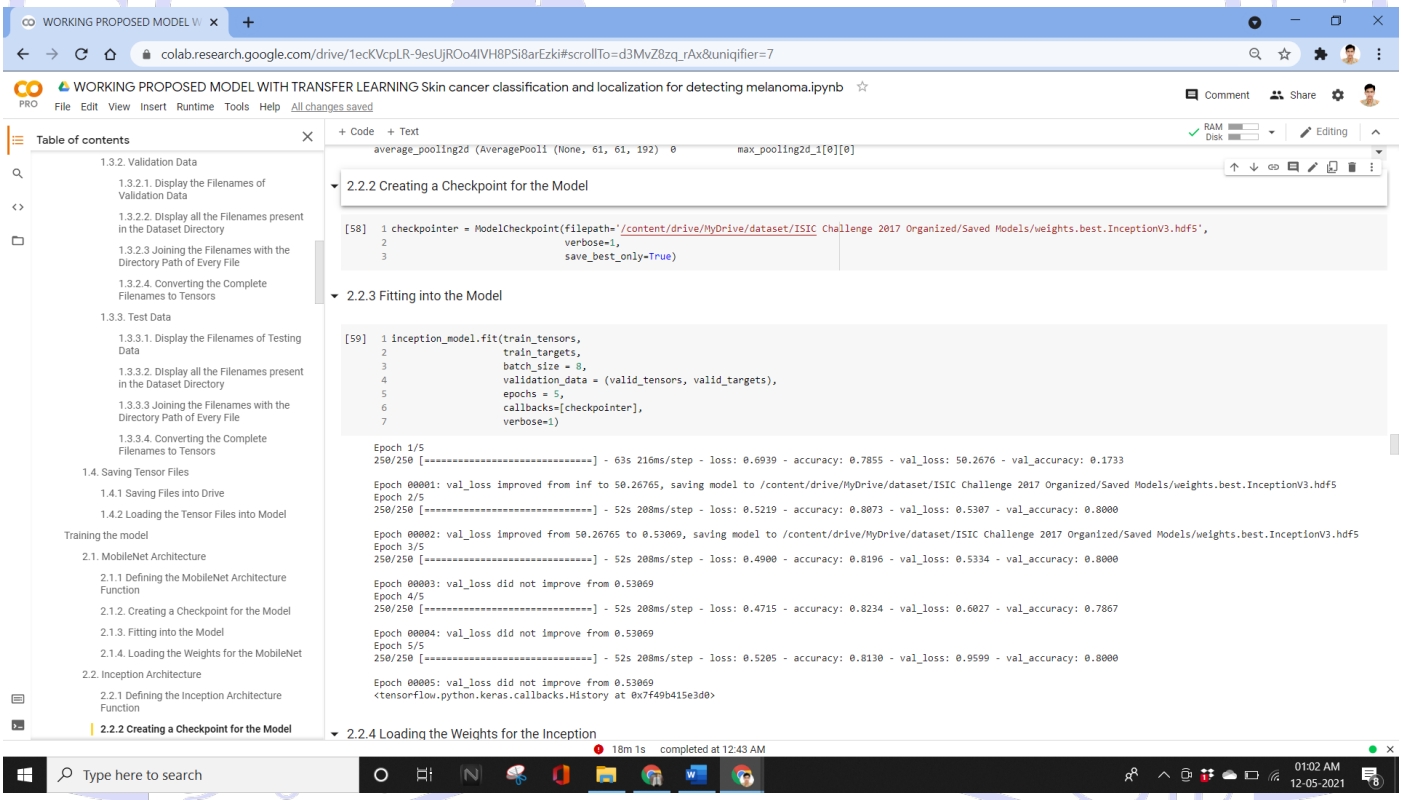

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=SUlkgRxsxLFj&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
1.3.3.4. Converting the Complete Filenames to Tensors
[48] 1 test_tensors = paths_to_tensor(test_files).astype('float32')/255
100% 600/600 [10:05<00:00, 1.01s/it]
1.4. Saving Tensor Files
1.4.1 Saving Files into Drive
[49] 1 # Saving the data
2 np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_training_tensors.npy", train_tensors)
3 np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_validation_tensors.npy", valid_tensors)
4 np.save("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_testing_tensors.npy", test_tensors)
1.4.2 Loading the Tensor Files into Model
[50] 1 # Loading the data
2 train_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_training_tensors.npy")
3 valid_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_validation_tensors.npy")
4 test_tensors = np.load("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved image tensors/augmented_testing_tensors.npy")
2. Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
[51] 1 def mobilenet_architecture():
2 """
18m 1s completed at 12:43 AM
Type here to search
01:01 AM
12-05-2021

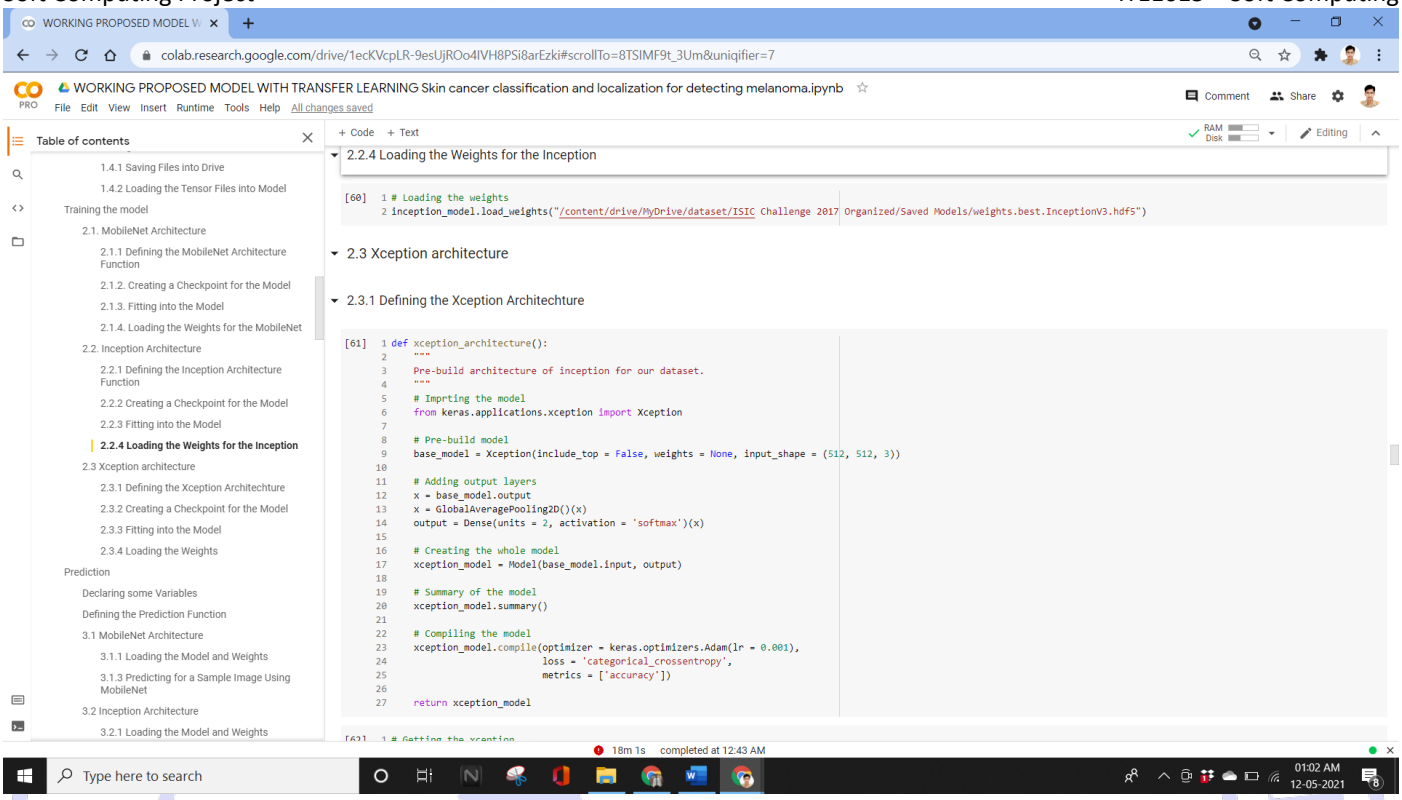

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=e9xjUEcp_BKA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
2.1.1 Defining the MobileNet Architecture Function
[51] 1 def mobilenet_architecture():
2 """
3 Pre-build architecture of mobilenet for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.mobilenet import MobileNet
7
8 # Pre-build model
9 base_model = MobileNet(include_top = False, weights = None, input_shape = (512, 512, 3))
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 mobilenet_model = Model(base_model.input, output)
18
19 # Getting the summary of architecture
20 mobilenet_model.summary()
21
22 # Compiling the model
23 mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
24 loss = 'categorical_crossentropy',
25 metrics = ['accuracy'])
26
27 return mobilenet_model
[52] 1 # Getting the mobilenet
2 mobilenet_model = mobilenet_architecture()
Model: "model"
Layer (type) Output Shape Param #
input_1 (InputLayer) [(None, 512, 512, 3)] 0
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

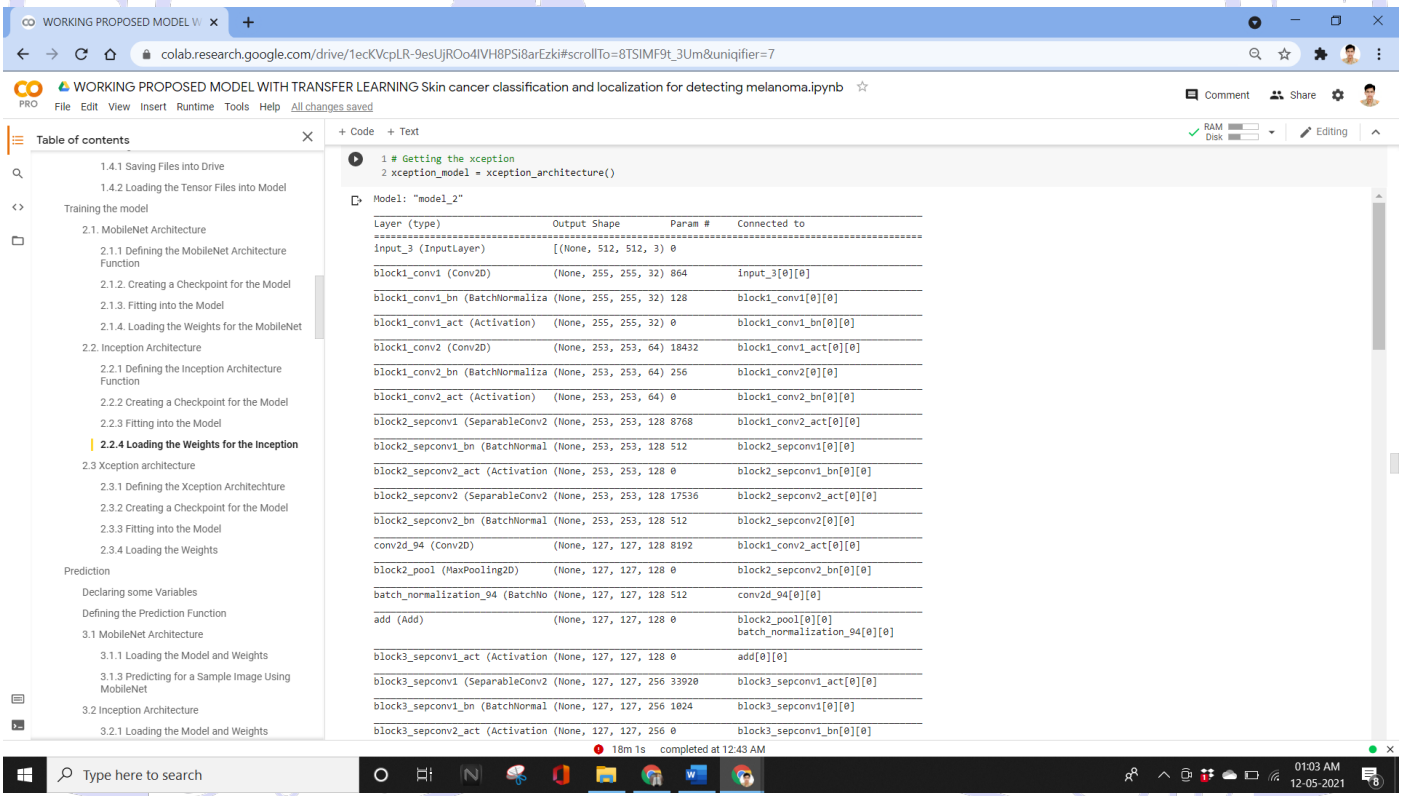

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=MfRLLdzG_HYY&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
1.4. Saving Tensor Files
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
+ Code + Text
RAM Disk
Editing
2.1.2. Creating a Checkpoint for the Model
[53] 1 checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5',
2 verbose=1,
3 save_best_only=True)
2.1.3. Fitting into the Model
1 mobilenet_model.fit(train_tensors,
2 train_targets,
3 batch_size = 8,
4 validation_data = (valid_tensors, valid_targets),
5 epochs = 5,
6 callbacks=[checkpointer],
7 verbose=1)
Epoch 1/5
250/250 [==============================] - 85s 210ms/step - loss: 0.5806 - accuracy: 0.7974 - val_loss: 0.5353 - val_accuracy: 0.8000
Epoch 00001: val_loss improved from inf to 0.53534, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Epoch 2/5
250/250 [==============================] - 50s 201ms/step - loss: 0.5206 - accuracy: 0.7999 - val_loss: 0.6683 - val_accuracy: 0.8000
Epoch 00002: val_loss did not improve from 0.53534
Epoch 3/5
250/250 [==============================] - 50s 200ms/step - loss: 0.5065 - accuracy: 0.8074 - val_loss: 0.5590 - val_accuracy: 0.8000
Epoch 00003: val_loss did not improve from 0.53534
Epoch 4/5
250/250 [==============================] - 50s 200ms/step - loss: 0.5036 - accuracy: 0.8048 - val_loss: 0.5172 - val_accuracy: 0.8000
Epoch 00004: val_loss improved from 0.53534 to 0.51721, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Epoch 5/5
250/250 [==============================] - 50s 201ms/step - loss: 0.4966 - accuracy: 0.8105 - val_loss: 0.9415 - val_accuracy: 0.8000
Epoch 00005: val_loss did not improve from 0.51721
<tensorflow.python.keras.callbacks.History at 0x7f4a104a7e90>
2.1.4. Loading the Weights for the MobileNet
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

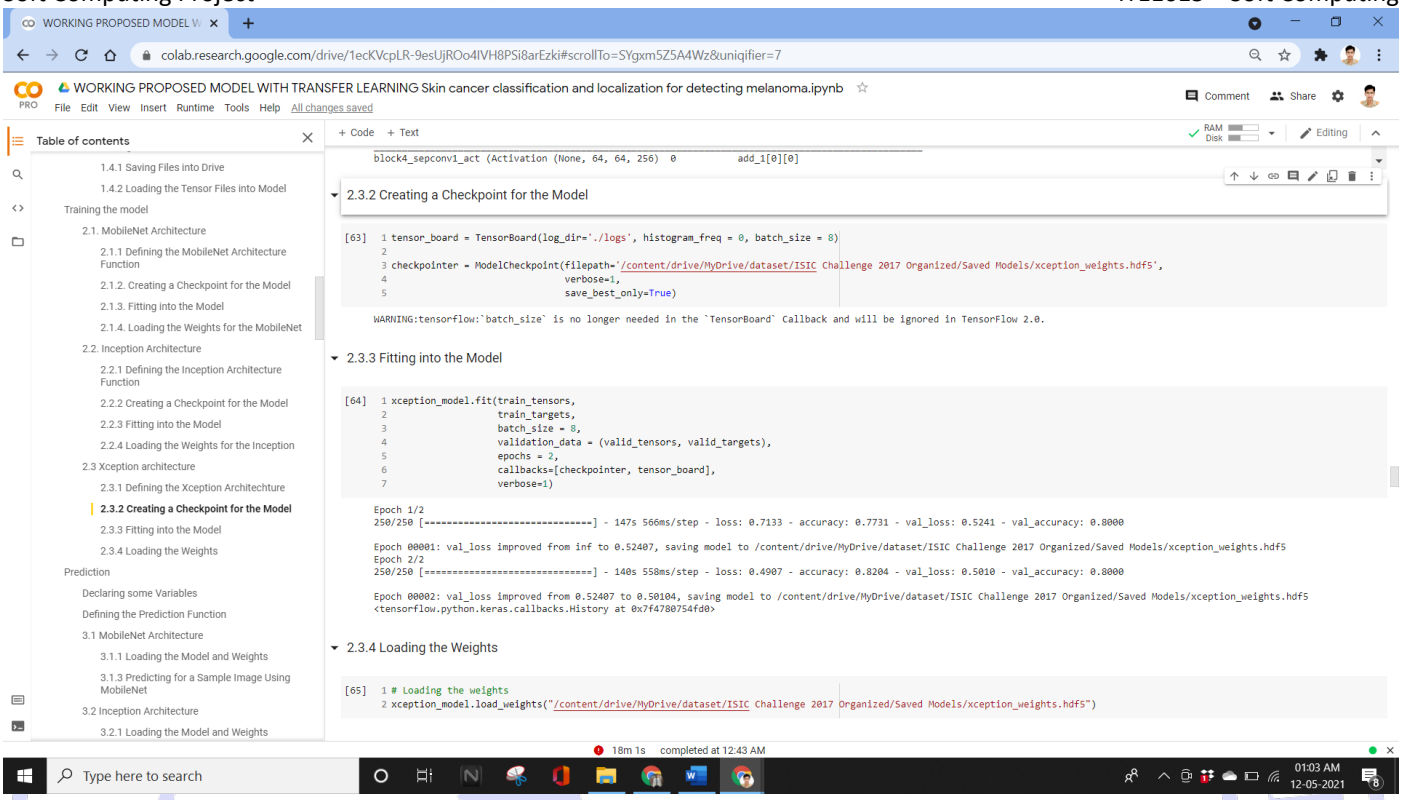

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=ik3k0M4v_Vpc&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
1.4. Saving Tensor Files
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
+ Code + Text
RAM Disk
Editing
250/250 [==============================] - 50s 201ms/step - loss: 0.4966 - accuracy: 0.8105 - val_loss: 0.9415 - val_accuracy: 0.8000
Epoch 00005: val_loss did not improve from 0.51721
<tensorflow.python.keras.callbacks.History at 0x7f4a104a7e90>
2.1.4. Loading the Weights for the MobileNet
[55] 1 # Loading the weights
2 mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5")
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
[56] 1 def inception_architecture():
2 """
3 Pre-build architecture of inception for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.inception_v3 import InceptionV3
7
8 # Pre-build model
9 base_model = InceptionV3(include_top = False, weights = None, input_shape = (512, 512, 3))
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 inception_model = Model(base_model.input, output)
18
19 # Summary of the model
20 inception_model.summary()
21
22 # Compiling the model
23 inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

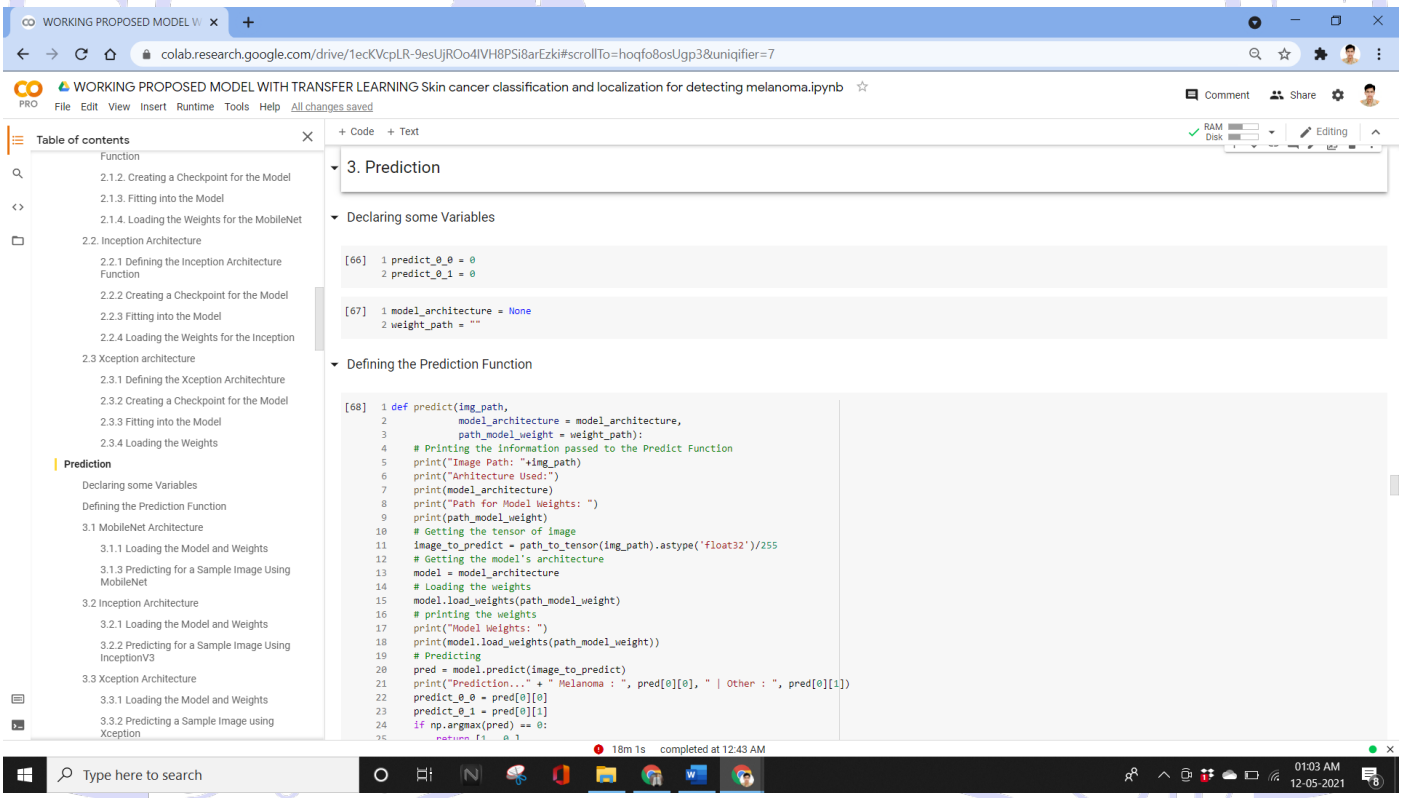

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=ik3k0M4v_Vpc&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
1.4. Saving Tensor Files
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
+ Code + Text
RAM Disk
Editing
output = Dense(units = 2, activation = 'softmax')(x)
# Creating the whole model
inception_model = Model(base_model.input, output)
# Summary of the model
inception_model.summary()
# Compiling the model
inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
loss = 'categorical_crossentropy',
metrics = ['accuracy'])
return inception_model
[57] 1 # Getting the inception
2 inception_model = inception_architecture()
Model: "model_1"
Layer (type) Output Shape Param # Connected to
input_2 (InputLayer) [(None, 512, 512, 3) 0
conv2d (Conv2D) (None, 255, 255, 32) 864 input_2[0][0]
batch_normalization (BatchNorma (None, 255, 255, 32) 96 conv2d[0][0]
activation (Activation) (None, 255, 255, 32) 0 batch_normalization[0][0]
conv2d_1 (Conv2D) (None, 253, 253, 32) 9216 activation[0][0]
batch_normalization_1 (BatchNor (None, 253, 253, 32) 96 conv2d_1[0][0]
activation_1 (Activation) (None, 253, 253, 32) 0 batch_normalization_1[0][0]
conv2d_2 (Conv2D) (None, 253, 253, 64) 18432 activation_1[0][0]
batch_normalization_2 (BatchNor (None, 253, 253, 64) 192 conv2d_2[0][0]
activation_2 (Activation) (None, 253, 253, 64) 0 batch_normalization_2[0][0]
max_pooling2d (MaxPooling2D) (None, 126, 126, 64) 0 activation_2[0][0]
conv2d_3 (Conv2D) (None, 126, 126, 80) 5120 max_pooling2d[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

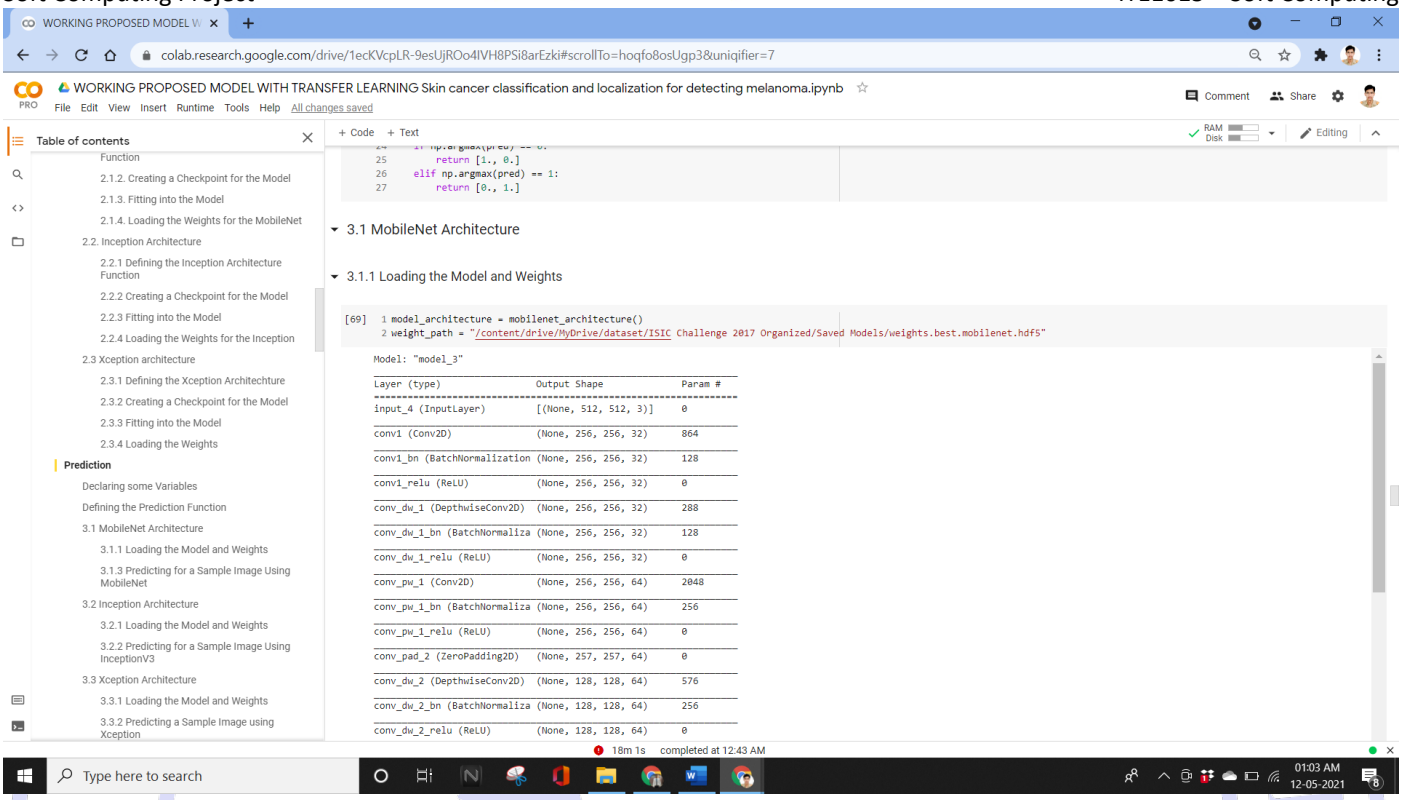

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=d3MvZ8zq_rAx&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
1.3.2. Validation Data
1.3.2.1. Display the Filenames of Validation Data
1.3.2.2. Display all the Filenames present in the Dataset Directory
1.3.2.3 Joining the Filenames with the Directory Path of Every File
1.3.2.4. Converting the Complete Filenames to Tensors
1.3.3. Test Data
1.3.3.1. Display the Filenames of Testing Data
1.3.3.2. Display all the Filenames present in the Dataset Directory
1.3.3.3 Joining the Filenames with the Directory Path of Every File
1.3.3.4. Converting the Complete Filenames to Tensors
1.4. Saving Tensor Files
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
+ Code + Text
RAM Disk
Editing
average_pooling2d (AveragePool1 (None, 61, 61, 192) 0 max_pooling2d_1[0][0]
2.2.2 Creating a Checkpoint for the Model
[58] 1 checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5',
2 verbose=1,
3 save_best_only=True)
2.2.3 Fitting into the Model
[59] 1 inception_model.fit(train_tensors,
2 train_targets,
3 batch_size = 8,
4 validation_data = (valid_tensors, valid_targets),
5 epochs = 5,
6 callbacks=[checkpointer],
7 verbose=1)
Epoch 1/5
250/250 [==============================] - 63s 216ms/step - loss: 0.6939 - accuracy: 0.7855 - val_loss: 50.2676 - val_accuracy: 0.1733
Epoch 00001: val_loss improved from inf to 50.26765, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Epoch 2/5
250/250 [==============================] - 52s 208ms/step - loss: 0.5219 - accuracy: 0.8073 - val_loss: 0.5307 - val_accuracy: 0.8000
Epoch 00002: val_loss improved from 50.26765 to 0.53069, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Epoch 3/5
250/250 [==============================] - 52s 208ms/step - loss: 0.4900 - accuracy: 0.8196 - val_loss: 0.5334 - val_accuracy: 0.8000
Epoch 00003: val_loss did not improve from 0.53069
Epoch 4/5
250/250 [==============================] - 52s 208ms/step - loss: 0.4715 - accuracy: 0.8234 - val_loss: 0.6027 - val_accuracy: 0.7867
Epoch 00004: val_loss did not improve from 0.53069
Epoch 5/5
250/250 [==============================] - 52s 208ms/step - loss: 0.5205 - accuracy: 0.8130 - val_loss: 0.9599 - val_accuracy: 0.8000
Epoch 00005: val_loss did not improve from 0.53069
<tensorflow.python.keras.callbacks.History at 0x7f49b415e3d0>
2.2.4 Loading the Weights for the Inception
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

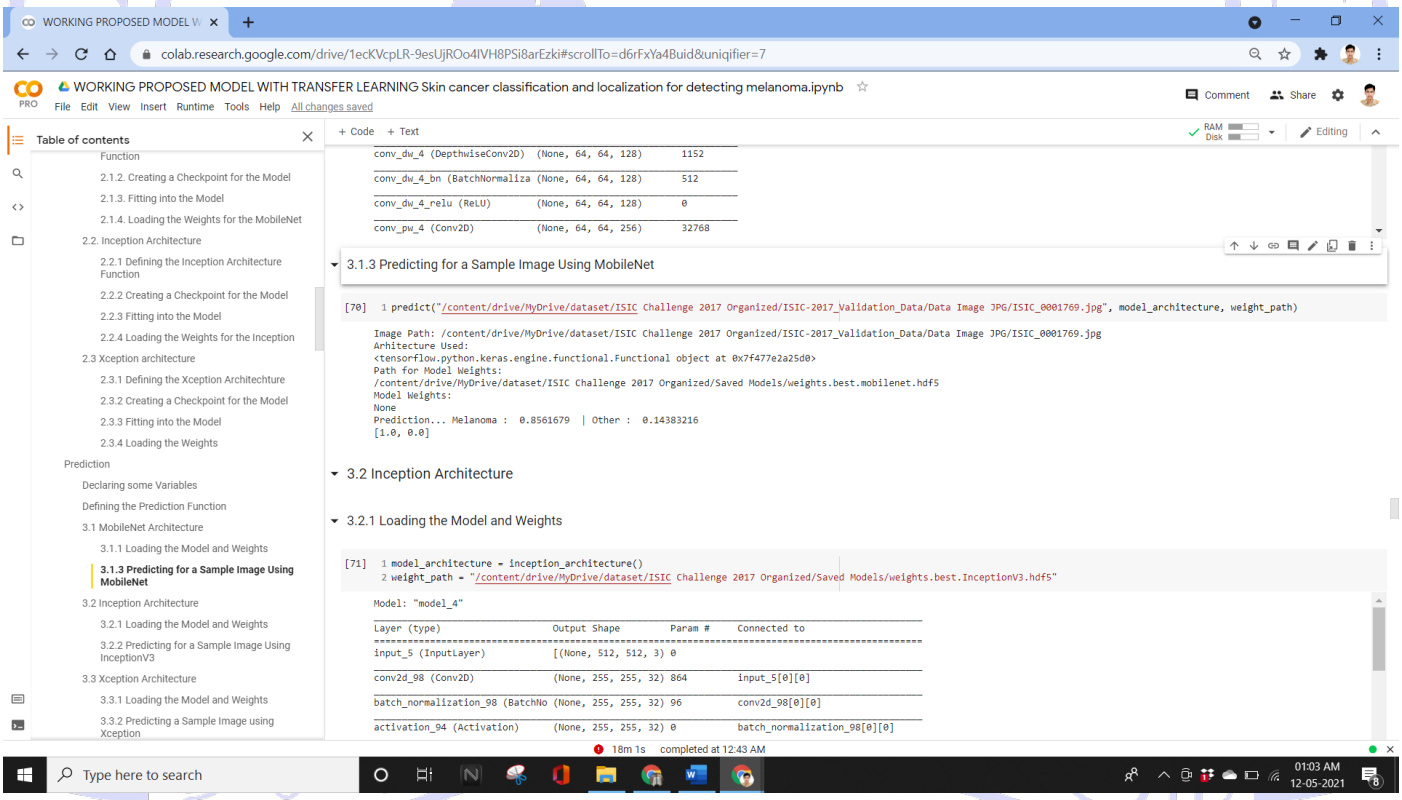

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=8TSIMF9t_3Um&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
2.2.3 Fitting into the Model
2.2.4 Loading the Weights for the Inception
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
2.3.2 Creating a Checkpoint for the Model
2.3.3 Fitting into the Model
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
+ Code + Text
RAM Disk
Editing
2.2.4 Loading the Weights for the Inception
[60] 1 # Loading the weights
2 inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5")
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
[61] 1 def xception_architecture():
2 """
3 Pre-build architecture of inception for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.xception import Xception
7
8 # Pre-build model
9 base_model = Xception(include_top = False, weights = None, input_shape = (512, 512, 3))
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 xception_model = Model(base_model.input, output)
18
19 # Summary of the model
20 xception_model.summary()
21
22 # Compiling the model
23 xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
24 loss = 'categorical_crossentropy',
25 metrics = ['accuracy'])
26
27 return xception_model
18m 1s completed at 12:43 AM
Type here to search
01:02 AM
12-05-2021

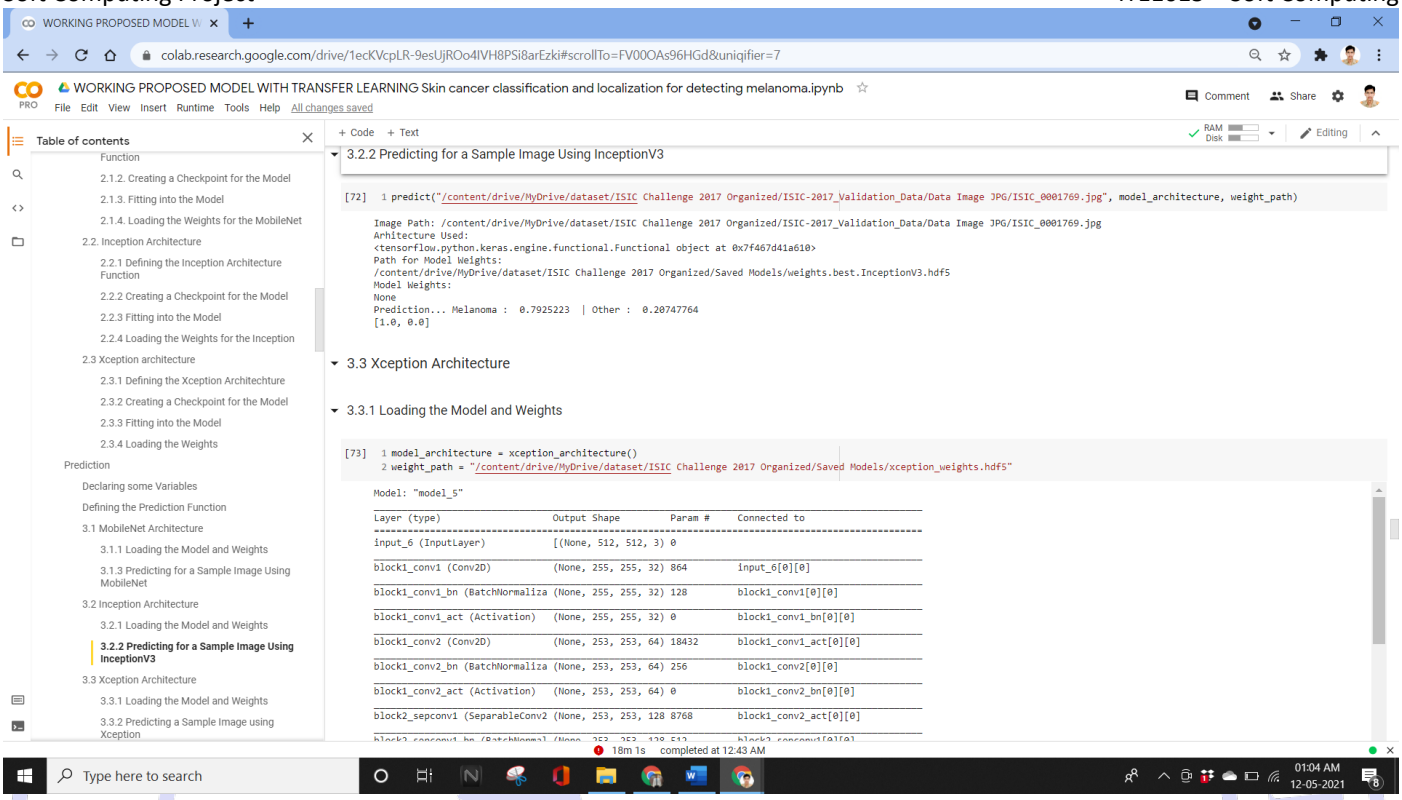

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=8TSIMF9t_3Um&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
1 # Getting the xception
2 xception_model = xception_architecture()
Model: "model_2"
Layer (type) Output Shape Param # Connected to
input_3 (InputLayer) [(None, 512, 512, 3) 0
block1_conv1 (Conv2D) (None, 255, 255, 32) 864 input_3[0][0]
block1_conv1_bn (BatchNormaliza (None, 255, 255, 32) 128 block1_conv1[0][0]
block1_conv1_act (Activation) (None, 255, 255, 32) 0 block1_conv1_bn[0][0]
block1_conv2 (Conv2D) (None, 253, 253, 64) 18432 block1_conv1_act[0][0]
block1_conv2_bn (BatchNormaliza (None, 253, 253, 64) 256 block1_conv2[0][0]
block1_conv2_act (Activation) (None, 253, 253, 64) 0 block1_conv2_bn[0][0]
block2_sepconv1 (SeparableConv2 (None, 253, 253, 128 8768 block1_conv2_act[0][0]
block2_sepconv1_bn (BatchNormal (None, 253, 253, 128 512 block2_sepconv1[0][0]
block2_sepconv2_act (Activation (None, 253, 253, 128 0 block2_sepconv1_bn[0][0]
block2_sepconv2 (SeparableConv2 (None, 253, 253, 128 17536 block2_sepconv2_act[0][0]
block2_sepconv2_bn (BatchNormal (None, 253, 253, 128 512 block2_sepconv2[0][0]
conv2d_94 (Conv2D) (None, 127, 127, 128 8192 block1_conv2_act[0][0]
block2_pool (MaxPooling2D) (None, 127, 127, 128 0 block2_sepconv2_bn[0][0]
batch_normalization_94 (BatchNo (None, 127, 127, 128 512 conv2d_94[0][0]
add (Add) (None, 127, 127, 128 0 block2_pool[0][0] batch_normalization_94[0][0]
block3_sepconv1_act (Activation (None, 127, 127, 128 0 add[0][0]
block3_sepconv1 (SeparableConv2 (None, 127, 127, 256 33920 block3_sepconv1_act[0][0]
block3_sepconv1_bn (BatchNormal (None, 127, 127, 256 1024 block3_sepconv1[0][0]
block3_sepconv2_act (Activation (None, 127, 127, 256 0 block3_sepconv1_bn[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:03 AM
12-05-2021

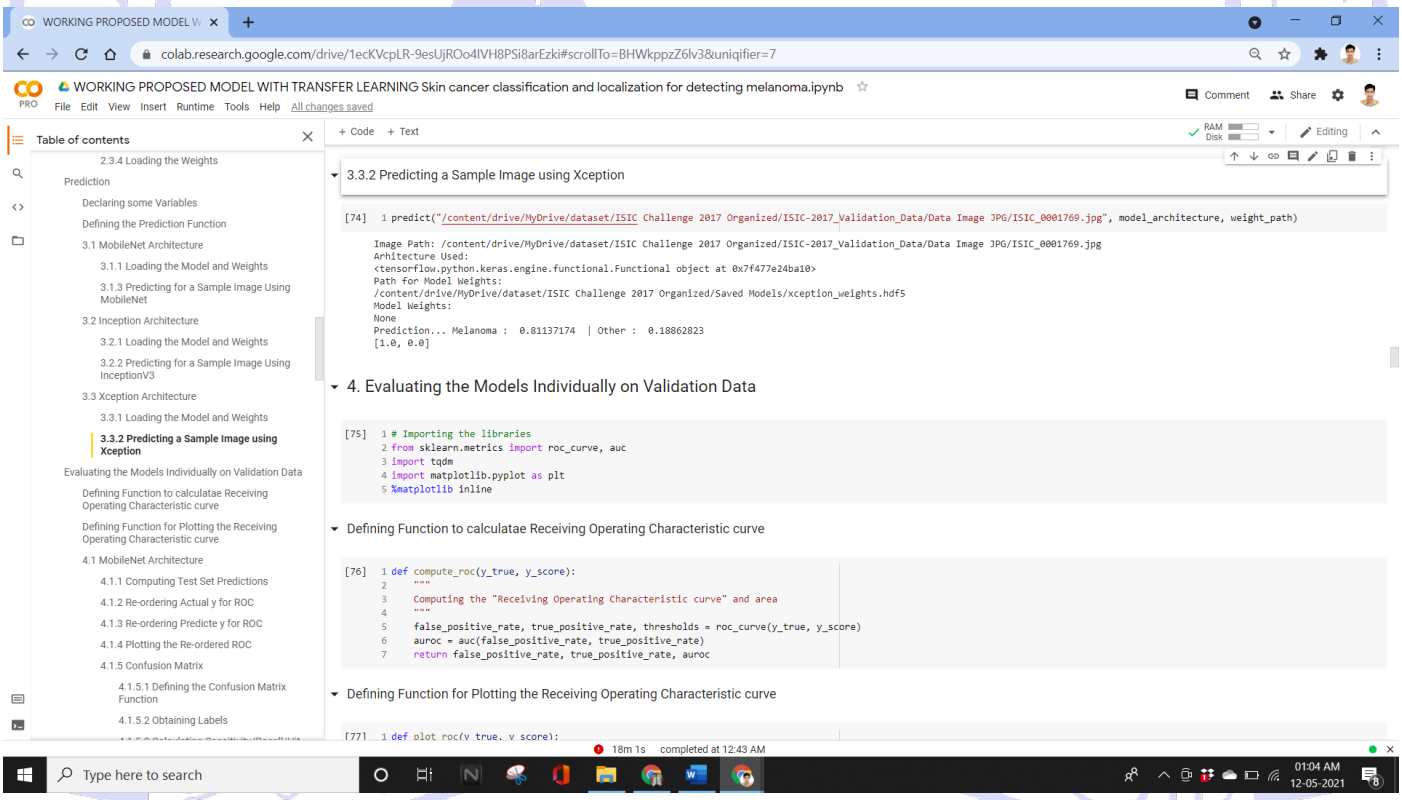

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=SYgxm5Z5A4Wz&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
1.4.1 Saving Files into Drive
1.4.2 Loading the Tensor Files into Model
Training the model
2.1. MobileNet Architecture
2.1.1 Defining the MobileNet Architecture Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
2.2.3 Fitting into the Model
2.2.4 Loading the Weights for the Inception
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
2.3.2 Creating a Checkpoint for the Model
2.3.3 Fitting into the Model
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
+ Code + Text
RAM Disk
Editing
block4_sepconv1_act (Activation (None, 64, 64, 256) 0 add_1[0][0]
2.3.2 Creating a Checkpoint for the Model
[63] 1 tensor_board = TensorBoard(log_dir='./logs', histogram_freq = 0, batch_size = 8)
2
3 checkpointer = ModelCheckpoint(filepath='/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5',
4 verbose=1,
5 save_best_only=True)
WARNING:tensorflow:`batch_size` is no longer needed in the `TensorBoard` Callback and will be ignored in TensorFlow 2.0.
2.3.3 Fitting into the Model
[64] 1 xception_model.fit(train_tensors,
2 train_targets,
3 batch_size = 8,
4 validation_data = (valid_tensors, valid_targets),
5 epochs = 2,
6 callbacks=[checkpointer, tensor_board],
7 verbose=1)
Epoch 1/2
250/250 [==============================] - 147s 566ms/step - loss: 0.7133 - accuracy: 0.7731 - val_loss: 0.5241 - val_accuracy: 0.8000
Epoch 00001: val_loss improved from inf to 0.52407, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Epoch 2/2
250/250 [==============================] - 140s 558ms/step - loss: 0.4907 - accuracy: 0.8204 - val_loss: 0.5010 - val_accuracy: 0.8000
Epoch 00002: val_loss improved from 0.52407 to 0.50104, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
<tensorflow.python.keras.callbacks.History at 0x7f4780754fd0>
2.3.4 Loading the Weights
[65] 1 # Loading the weights
2 xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5")
18m 1s completed at 12:43 AM
Type here to search
01:03 AM 12-05-2021

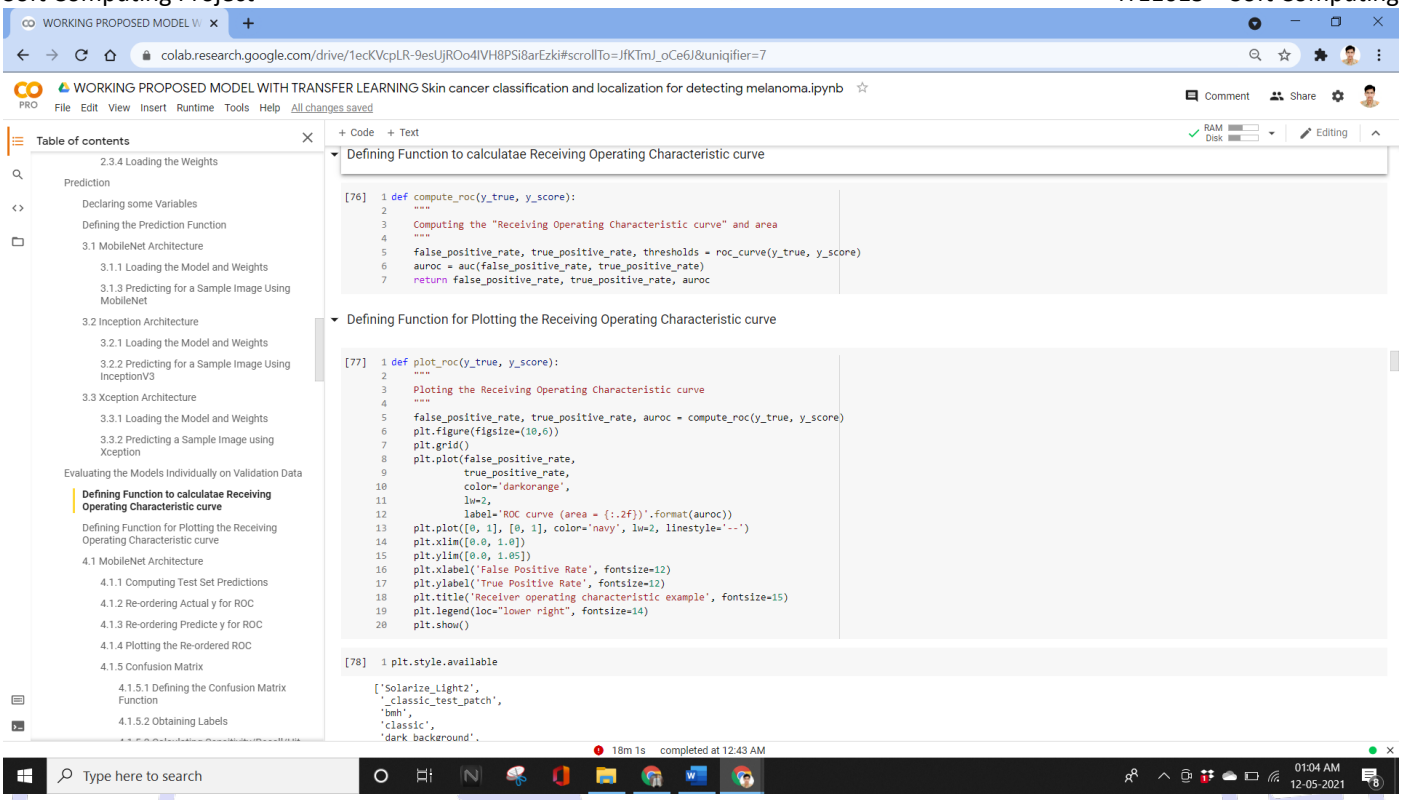

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=hoqfo8osUgp3&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
2.2.3 Fitting into the Model
2.2.4 Loading the Weights for the Inception
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
2.3.2 Creating a Checkpoint for the Model
2.3.3 Fitting into the Model
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
+ Code + Text
RAM Disk
Editing
3. Prediction
Declaring some Variables
[66] 1 predict_0_0 = 0
2 predict_0_1 = 0
[67] 1 model_architecture = None
2 weight_path = ""
Defining the Prediction Function
[68] 1 def predict(img_path,
2 model_architecture = model_architecture,
3 path_model_weight = weight_path):
4 # Printing the information passed to the Predict Function
5 print("Image Path: "+img_path)
6 print("Arhitecture Used:")
7 print(model_architecture)
8 print("Path for Model Weights: ")
9 print(path_model_weight)
10 # Getting the tensor of image
11 image_to_predict = path_to_tensor(img_path).astype('float32')/255
12 # Getting the model's architecture
13 model = model_architecture
14 # Loading the weights
15 model.load_weights(path_model_weight)
16 # printing the weights
17 print("Model Weights: ")
18 print(model.load_weights(path_model_weight))
19 # Predicting
20 pred = model.predict(image_to_predict)
21 print("Prediction..." + " Melanoma : ", pred[0][0], " | Other : ", pred[0][1])
22 predict_0_0 = pred[0][0]
23 predict_0_1 = pred[0][1]
24 if np.argmax(pred) == 0:
18m 1s completed at 12:43 AM
Type here to search
01:03 AM 12-05-2021

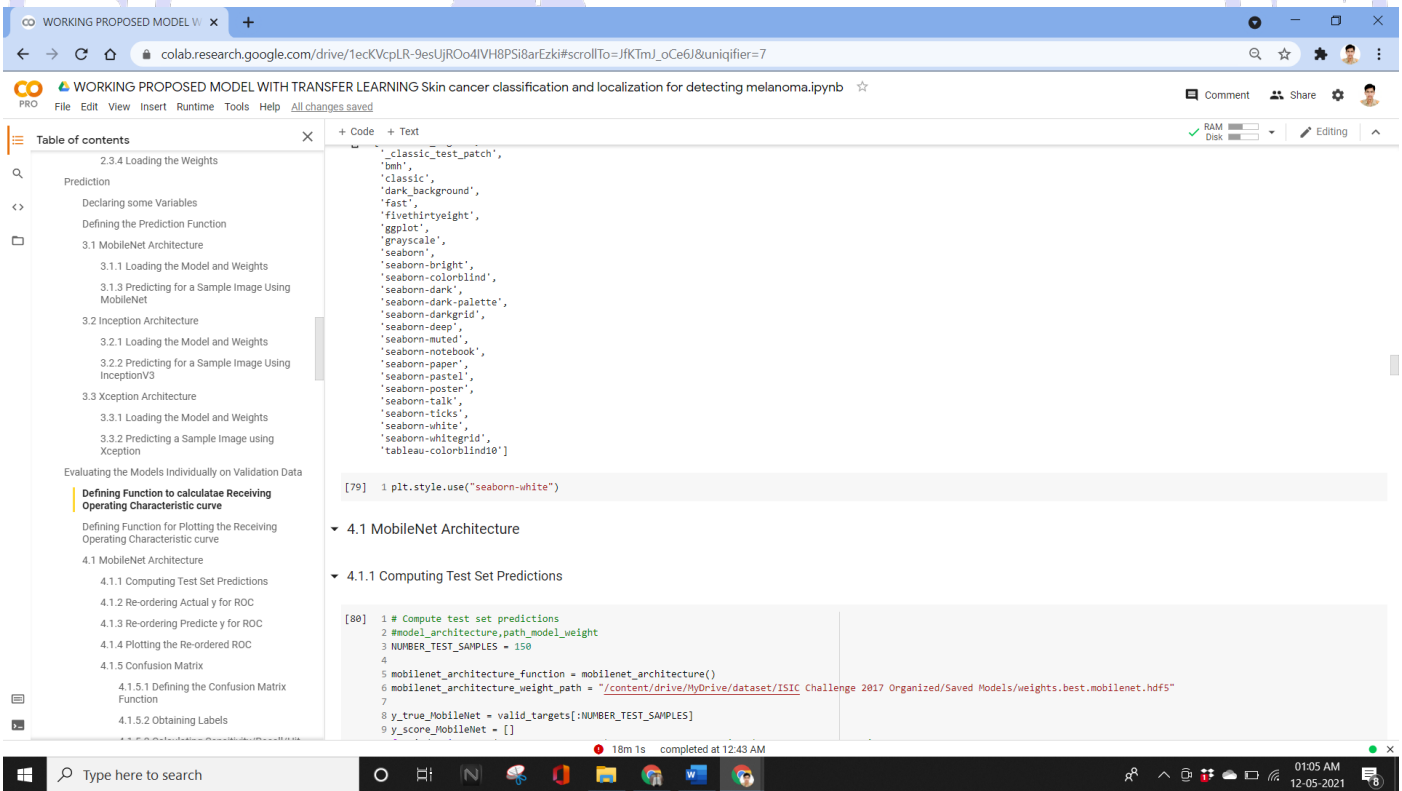

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=hoqfo8osUgp3&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
2.2.3 Fitting into the Model
2.2.4 Loading the Weights for the Inception
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
2.3.2 Creating a Checkpoint for the Model
2.3.3 Fitting into the Model
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
+ Code + Text
RAM Disk
Editing
return [1., 0.]
elif np.argmax(pred) == 1:
return [0., 1.]
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
[69] 1 model_architecture = mobilenet_architecture()
2 weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
Model: "model_3"
Layer (type) Output Shape Param #
input_4 (InputLayer) [(None, 512, 512, 3)] 0
conv1 (Conv2D) (None, 256, 256, 32) 864
conv1_bn (BatchNormalization (None, 256, 256, 32) 128
conv1_relu (ReLU) (None, 256, 256, 32) 0
conv_dw_1 (DepthwiseConv2D) (None, 256, 256, 32) 288
conv_dw_1_bn (BatchNormaliza (None, 256, 256, 32) 128
conv_dw_1_relu (ReLU) (None, 256, 256, 32) 0
conv_pw_1 (Conv2D) (None, 256, 256, 64) 2048
conv_pw_1_bn (BatchNormaliza (None, 256, 256, 64) 256
conv_pw_1_relu (ReLU) (None, 256, 256, 64) 0
conv_pad_2 (ZeroPadding2D) (None, 257, 257, 64) 0
conv_dw_2 (DepthwiseConv2D) (None, 128, 128, 64) 576
conv_dw_2_bn (BatchNormaliza (None, 128, 128, 64) 256
conv_dw_2_relu (ReLU) (None, 128, 128, 64) 0
18m 1s completed at 12:43 AM
Type here to search
01:03 AM 12-05-2021

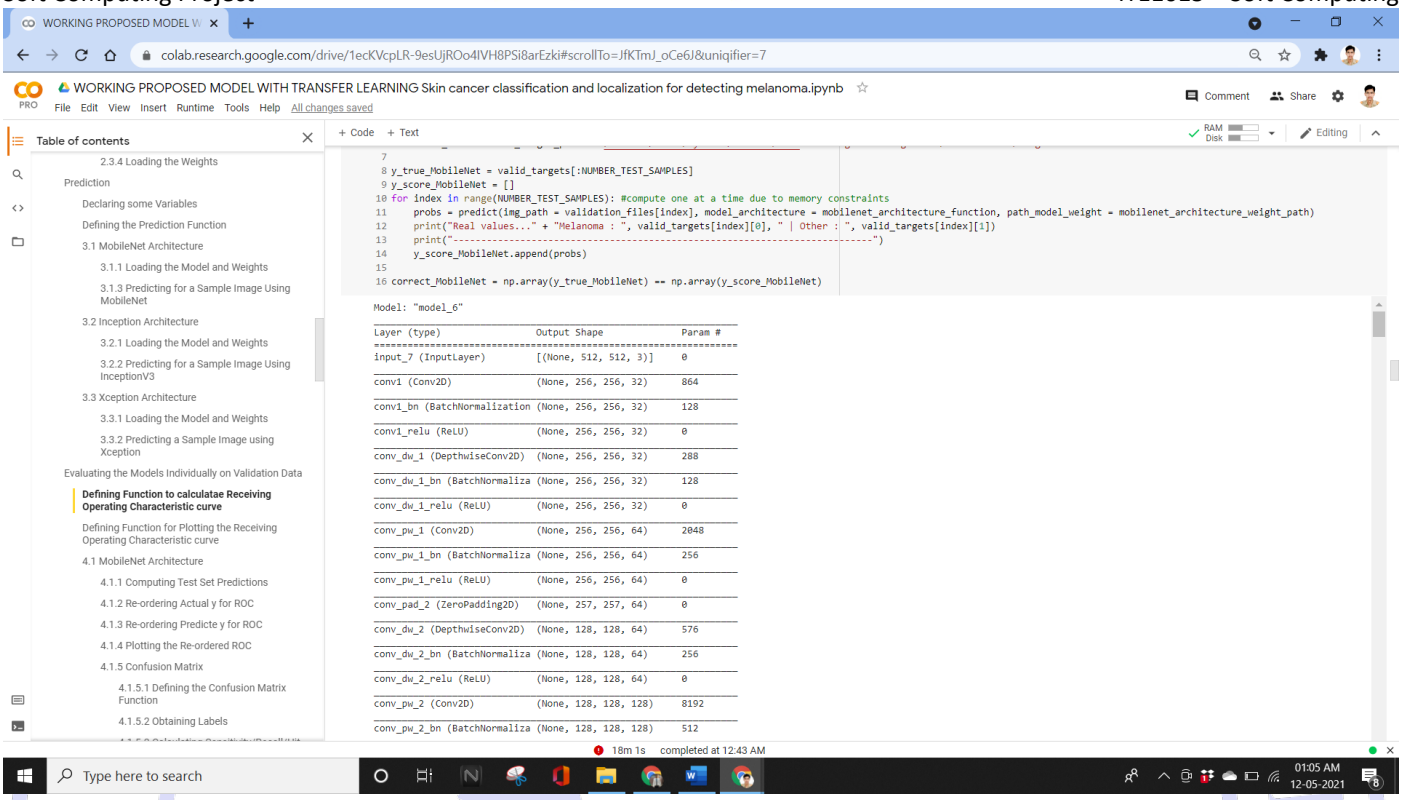

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=d6rFxYa4Buid&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
conv_dw_4 (DepthwiseConv2D) (None, 64, 64, 128) 1152
conv_dw_4_bn (BatchNormaliza (None, 64, 64, 128) 512
conv_dw_4_relu (ReLU) (None, 64, 64, 128) 0
conv_pw_4 (Conv2D) (None, 64, 64, 256) 32768
3.1.3 Predicting for a Sample Image Using MobileNet
[70] 1 predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e2a25d0>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8561679 | Other : 0.14383216
[1.0, 0.0]
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
[71] 1 model_architecture = inception_architecture()
2 weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5"
Model: "model_4"
Layer (type) Output Shape Param # Connected to
input_5 (InputLayer) [(None, 512, 512, 3) 0
conv2d_98 (Conv2D) (None, 255, 255, 32) 864 input_5[0][0]
batch_normalization_98 (BatchNo (None, 255, 255, 32) 96 conv2d_98[0][0]
activation_94 (Activation) (None, 255, 255, 32) 0 batch_normalization_98[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:03 AM 12-05-2021

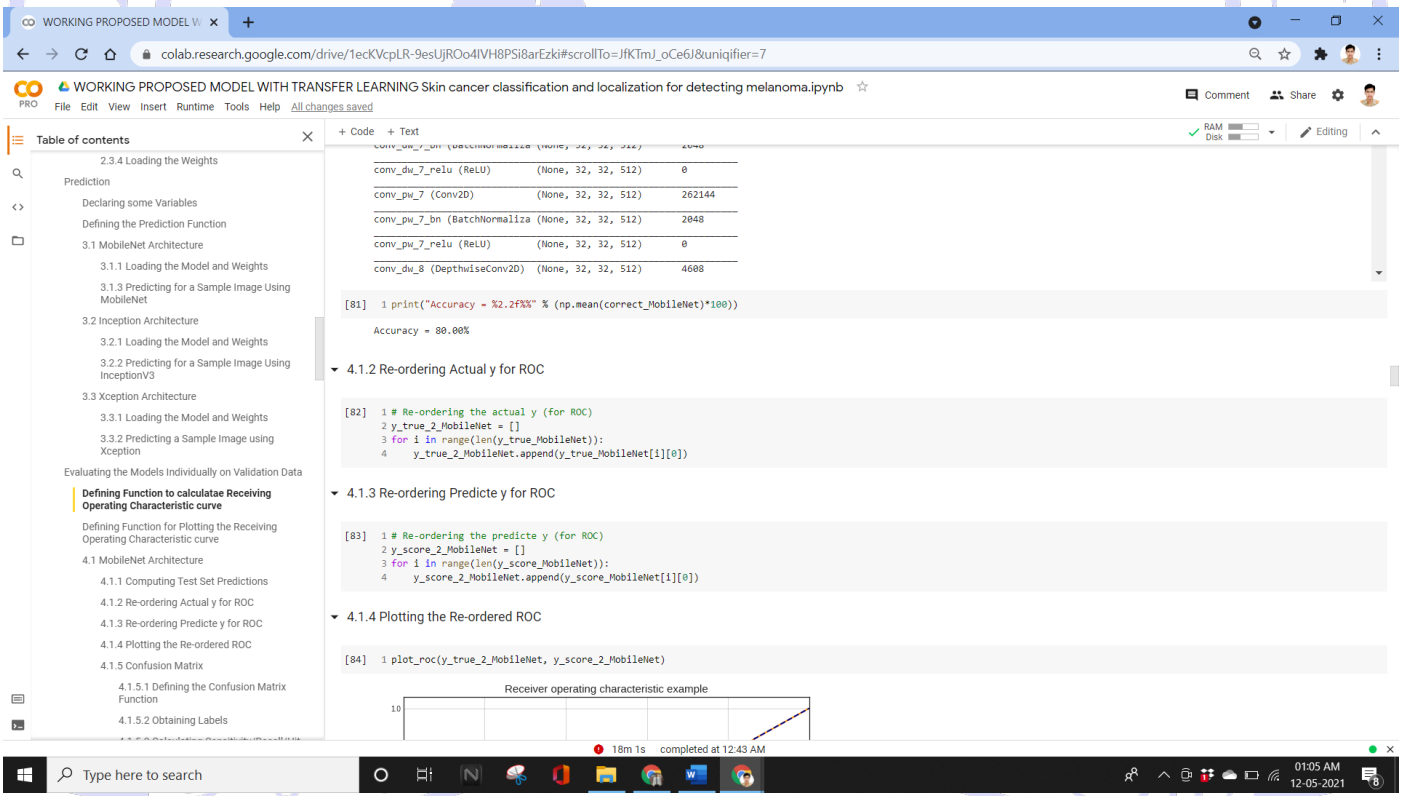

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=FV00OAs96HGd&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
Function
2.1.2. Creating a Checkpoint for the Model
2.1.3. Fitting into the Model
2.1.4. Loading the Weights for the MobileNet
2.2. Inception Architecture
2.2.1 Defining the Inception Architecture Function
2.2.2 Creating a Checkpoint for the Model
2.2.3 Fitting into the Model
2.2.4 Loading the Weights for the Inception
2.3 Xception architecture
2.3.1 Defining the Xception Architechture
2.3.2 Creating a Checkpoint for the Model
2.3.3 Fitting into the Model
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
3.2.2 Predicting for a Sample Image Using InceptionV3
[72] 1 predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41a610>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.7925223 | Other : 0.20747764
[1.0, 0.0]
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
[73] 1 model_architecture = xception_architecture()
2 weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5"
Model: "model_5"
Layer (type) Output Shape Param # Connected to
input_6 (InputLayer) [(None, 512, 512, 3) 0
block1_conv1 (Conv2D) (None, 255, 255, 32) 864 input_6[0][0]
block1_conv1_bn (BatchNormaliza (None, 255, 255, 32) 128 block1_conv1[0][0]
block1_conv1_act (Activation) (None, 255, 255, 32) 0 block1_conv1_bn[0][0]
block1_conv2 (Conv2D) (None, 253, 253, 64) 18432 block1_conv1_act[0][0]
block1_conv2_bn (BatchNormaliza (None, 253, 253, 64) 256 block1_conv2[0][0]
block1_conv2_act (Activation) (None, 253, 253, 64) 0 block1_conv2_bn[0][0]
block2_sepconv1 (SeparableConv2 (None, 253, 253, 128 8768 block1_conv2_act[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:04 AM
12-05-2021

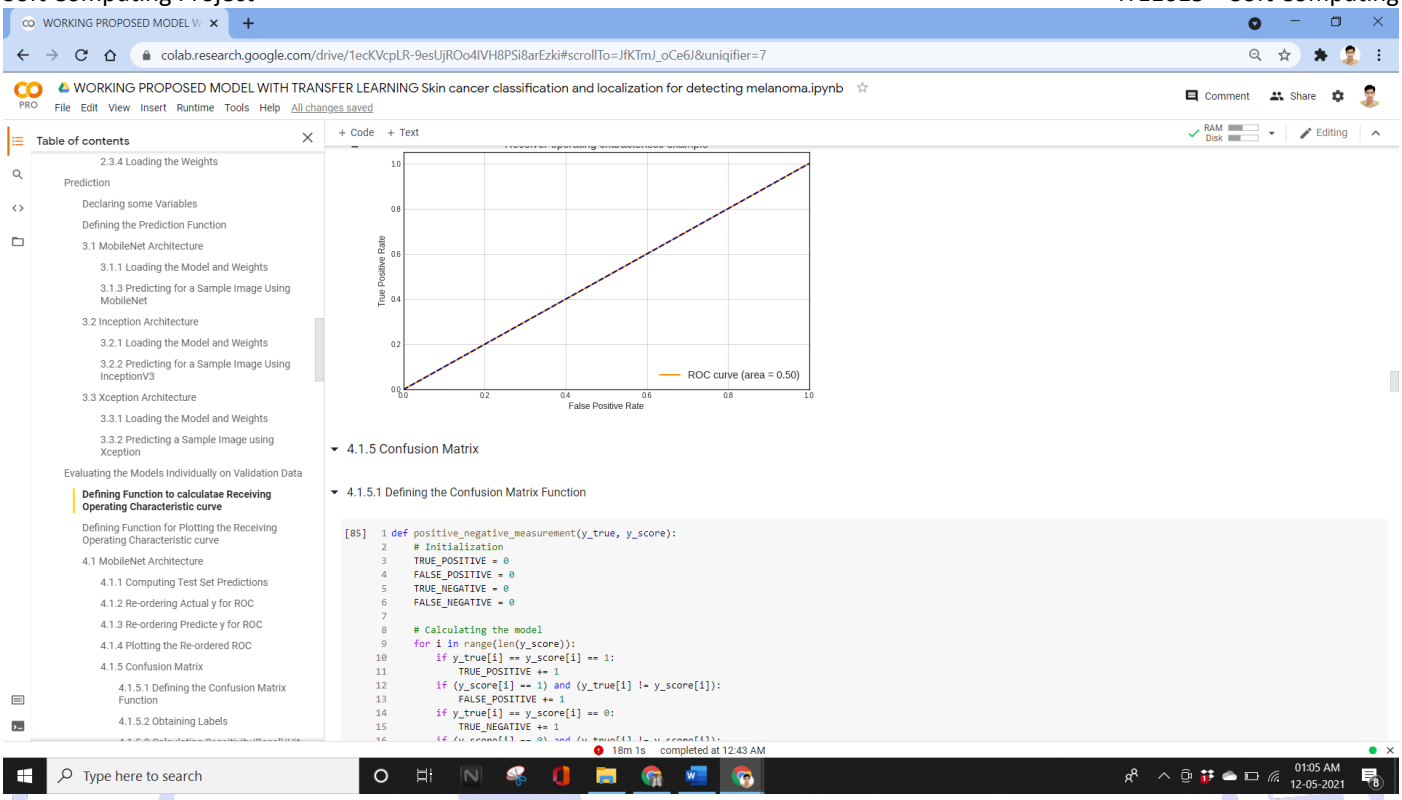

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=BHWkppzZ6lv3&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
3.3.2 Predicting a Sample Image using Xception
[74] 1 predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg", model_architecture, weight_path)
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e24ba10>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.81137174 | Other : 0.18862823
[1.0, 0.0]
4. Evaluating the Models Individually on Validation Data
[75] 1 # Importing the libraries
2 from sklearn.metrics import roc_curve, auc
3 import tqdm
4 import matplotlib.pyplot as plt
5 %matplotlib inline
Defining Function to calculatae Receiving Operating Characteristic curve
[76] 1 def compute_roc(y_true, y_score):
2 """
3 Computing the "Receiving Operating Characteristic curve" and area
4 """
5 false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)
6 auroc = auc(false_positive_rate, true_positive_rate)
7 return false_positive_rate, true_positive_rate, auroc
Defining Function for Plotting the Receiving Operating Characteristic curve
[77] 1 def plot_roc(y_true, y_score):
18m 1s completed at 12:43 AM
Type here to search
01:04 AM
12-05-2021

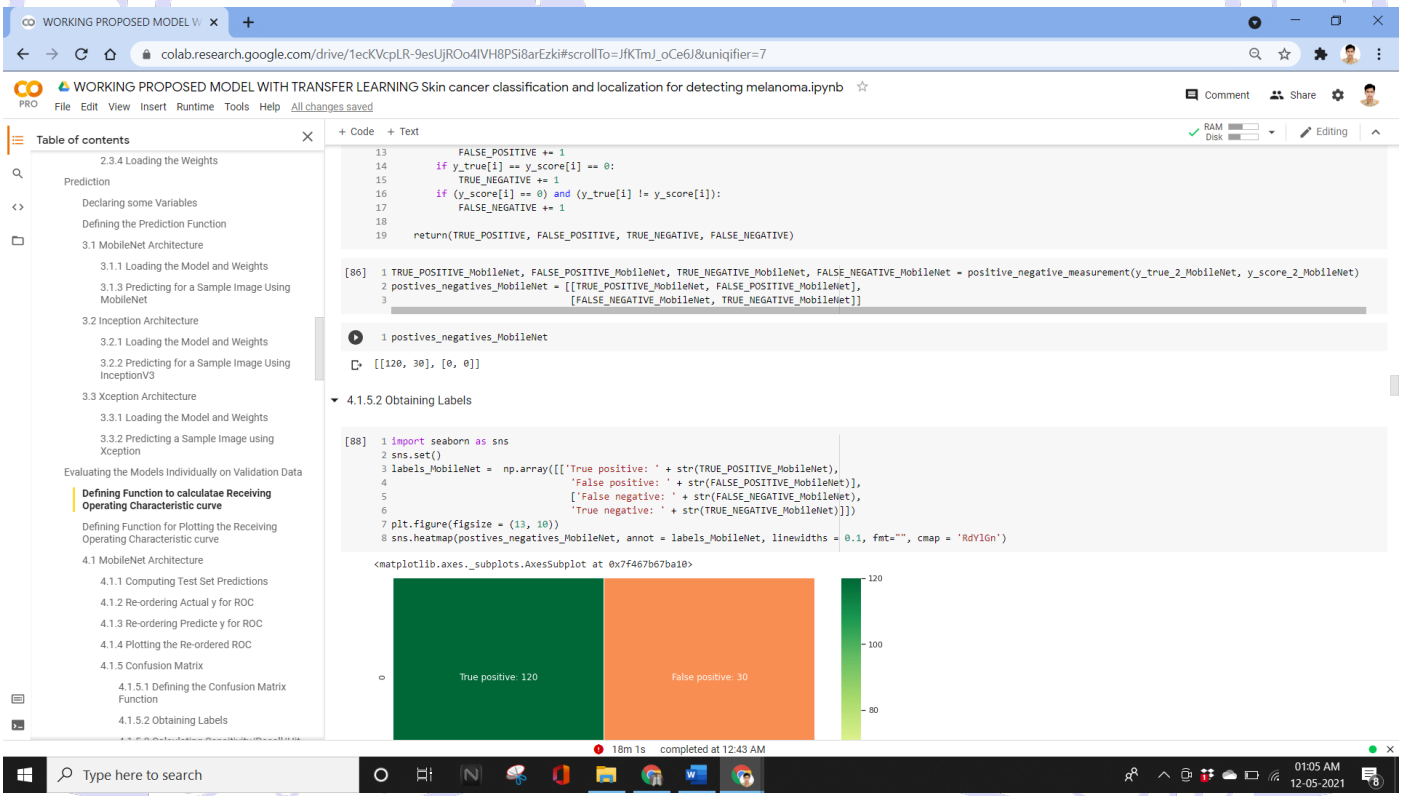

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
+ Code + Text
RAM Disk
Editing
Defining Function to calculatae Receiving Operating Characteristic curve
[76] 1 def compute_roc(y_true, y_score):
2 """
3 Computing the "Receiving Operating Characteristic curve" and area
4 """
5 false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)
6 auroc = auc(false_positive_rate, true_positive_rate)
7 return false_positive_rate, true_positive_rate, auroc
Defining Function for Plotting the Receiving Operating Characteristic curve
[77] 1 def plot_roc(y_true, y_score):
2 """
3 Ploting the Receiving Operating Characteristic curve
4 """
5 false_positive_rate, true_positive_rate, auroc = compute_roc(y_true, y_score)
6 plt.figure(figsize=(10,6))
7 plt.grid()
8 plt.plot(false_positive_rate,
9 true_positive_rate,
10 color='darkorange',
11 lw=2,
12 label='ROC curve (area = {:.2f})'.format(auroc))
13 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
14 plt.xlim([0.0, 1.0])
15 plt.ylim([0.0, 1.05])
16 plt.xlabel('False Positive Rate', fontsize=12)
17 plt.ylabel('True Positive Rate', fontsize=12)
18 plt.title('Receiver operating characteristic example', fontsize=15)
19 plt.legend(loc="lower right", fontsize=14)
20 plt.show()
[78] 1 plt.style.available
['Solarize_Light2',
'_classic_test_patch',
'bmh',
'classic',
'dark_background',
18m 1s completed at 12:43 AM
Type here to search
01:04 AM
12-05-2021

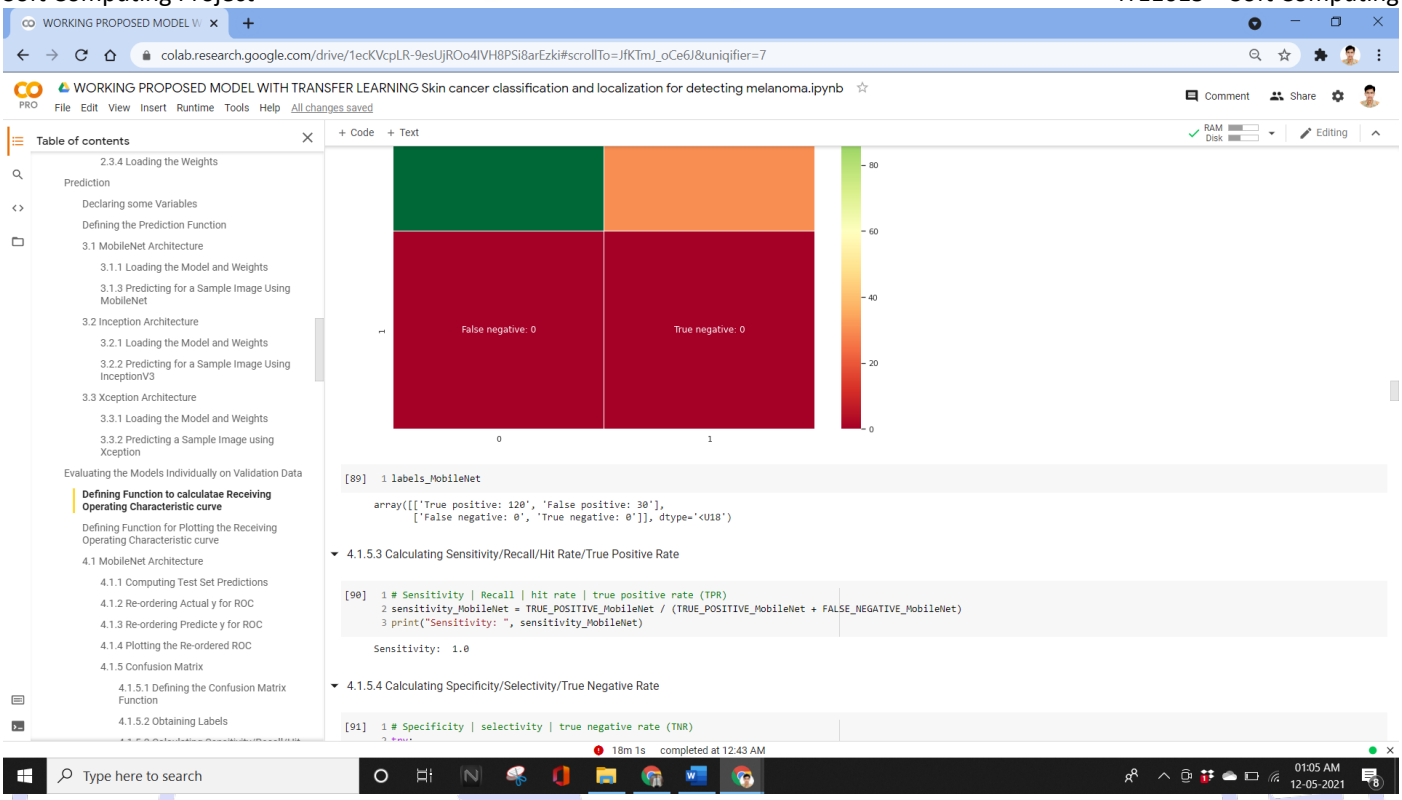

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
+ Code + Text
RAM Disk
Editing
'_classic_test_patch',
'bmh',
'classic',
'dark_background',
'fast',
'fivethirtyeight',
'ggplot',
'grayscale',
'seaborn',
'seaborn-bright',
'seaborn-colorblind',
'seaborn-dark',
'seaborn-dark-palette',
'seaborn-darkgrid',
'seaborn-deep',
'seaborn-muted',
'seaborn-notebook',
'seaborn-paper',
'seaborn-pastel',
'seaborn-poster',
'seaborn-talk',
'seaborn-ticks',
'seaborn-white',
'seaborn-whitegrid',
'tableau-colorblind10']
[79] 1 plt.style.use("seaborn-white")
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
[80] 1 # Compute test set predictions
2 #model_architecture,path_model_weight
3 NUMBER_TEST_SAMPLES = 150
4
5 mobilenet_architecture_function = mobilenet_architecture()
6 mobilenet_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
7
8 y_true_MobileNet = valid_targets[:NUMBER_TEST_SAMPLES]
9 y_score_MobileNet = []
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

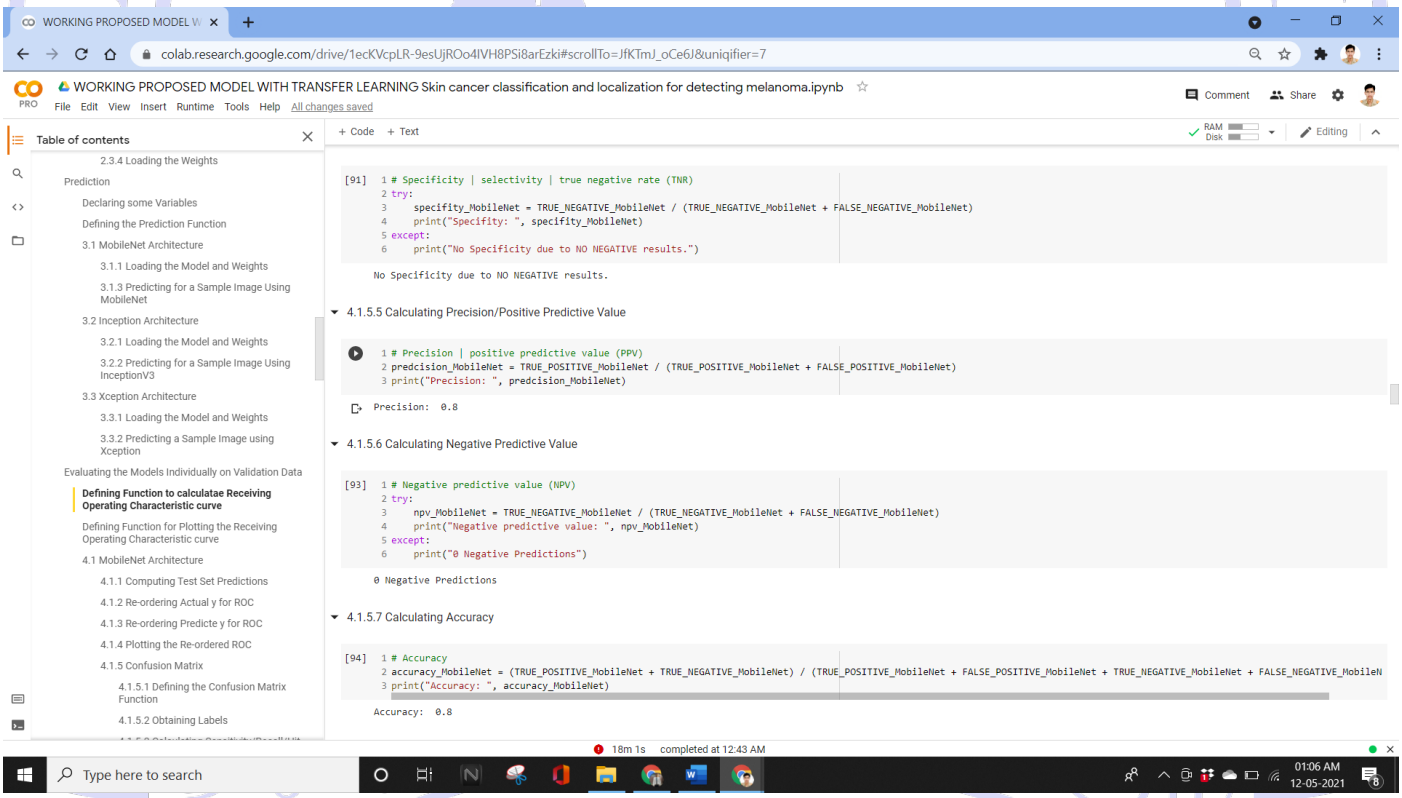

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
7
8 y_true_MobileNet = valid_targets[:NUMBER_TEST_SAMPLES]
9 y_score_MobileNet = []
10 for index in range(NUMBER_TEST_SAMPLES): #compute one at a time due to memory constraints
11 probs = predict(img_path = validation_files[index], model_architecture = mobilenet_architecture_function, path_model_weight = mobilenet_architecture_weight_path)
12 print("Real values..." + "Melanoma : ", valid_targets[index][0], " | Other : ", valid_targets[index][1])
13 print("---")
14 y_score_MobileNet.append(probs)
15
16 correct_MobileNet = np.array(y_true_MobileNet) == np.array(y_score_MobileNet)
Model: "model_6"
Layer (type) Output Shape Param #
input_7 (InputLayer) [(None, 512, 512, 3)] 0
conv1 (Conv2D) (None, 256, 256, 32) 864
conv1_bn (BatchNormalization (None, 256, 256, 32) 128
conv1_relu (ReLU) (None, 256, 256, 32) 0
conv_dw_1 (DepthwiseConv2D) (None, 256, 256, 32) 288
conv_dw_1_bn (BatchNormaliza (None, 256, 256, 32) 128
conv_dw_1_relu (ReLU) (None, 256, 256, 32) 0
conv_pw_1 (Conv2D) (None, 256, 256, 64) 2048
conv_pw_1_bn (BatchNormaliza (None, 256, 256, 64) 256
conv_pw_1_relu (ReLU) (None, 256, 256, 64) 0
conv_pad_2 (ZeroPadding2D) (None, 257, 257, 64) 0
conv_dw_2 (DepthwiseConv2D) (None, 128, 128, 64) 576
conv_dw_2_bn (BatchNormaliza (None, 128, 128, 64) 256
conv_dw_2_relu (ReLU) (None, 128, 128, 64) 0
conv_pw_2 (Conv2D) (None, 128, 128, 128) 8192
conv_pw_2_bn (BatchNormaliza (None, 128, 128, 128) 512
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

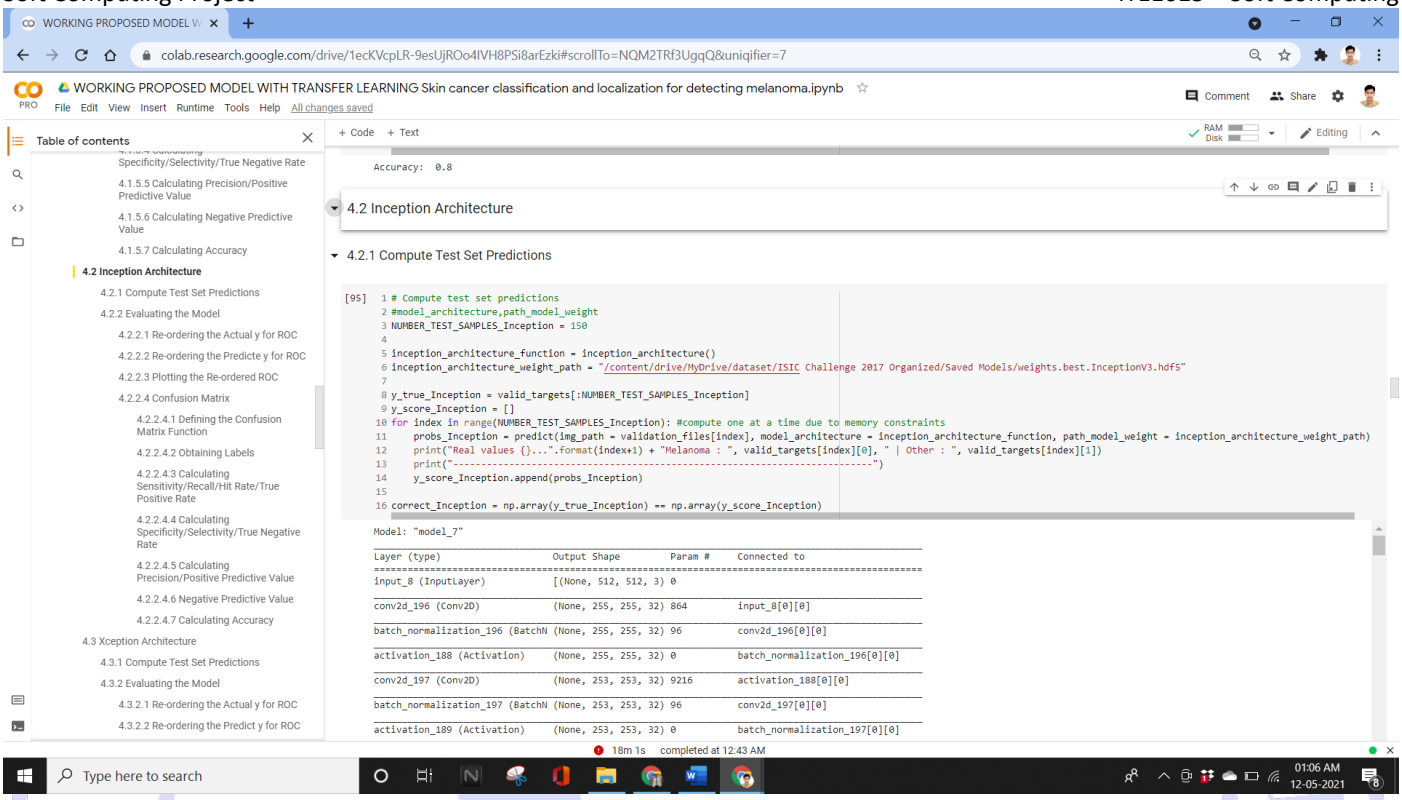

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
conv_dw_7_relu (ReLU) (None, 32, 32, 512) 0
conv_pw_7 (Conv2D) (None, 32, 32, 512) 262144
conv_pw_7_bn (BatchNormaliza (None, 32, 32, 512) 2048
conv_pw_7_relu (ReLU) (None, 32, 32, 512) 0
conv_dw_8 (DepthwiseConv2D) (None, 32, 32, 512) 4608
[81] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_MobileNet)*100))
Accuracy = 80.00%
4.1.2 Re-ordering Actual y for ROC
[82] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_MobileNet = []
3 for i in range(len(y_true_MobileNet)):
4 y_true_2_MobileNet.append(y_true_MobileNet[i][0])
4.1.3 Re-ordering Predicte y for ROC
[83] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_MobileNet = []
3 for i in range(len(y_score_MobileNet)):
4 y_score_2_MobileNet.append(y_score_MobileNet[i][0])
4.1.4 Plotting the Re-ordered ROC
[84] 1 plot_roc(y_true_2_MobileNet, y_score_2_MobileNet)
Receiver operating characteristic example
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

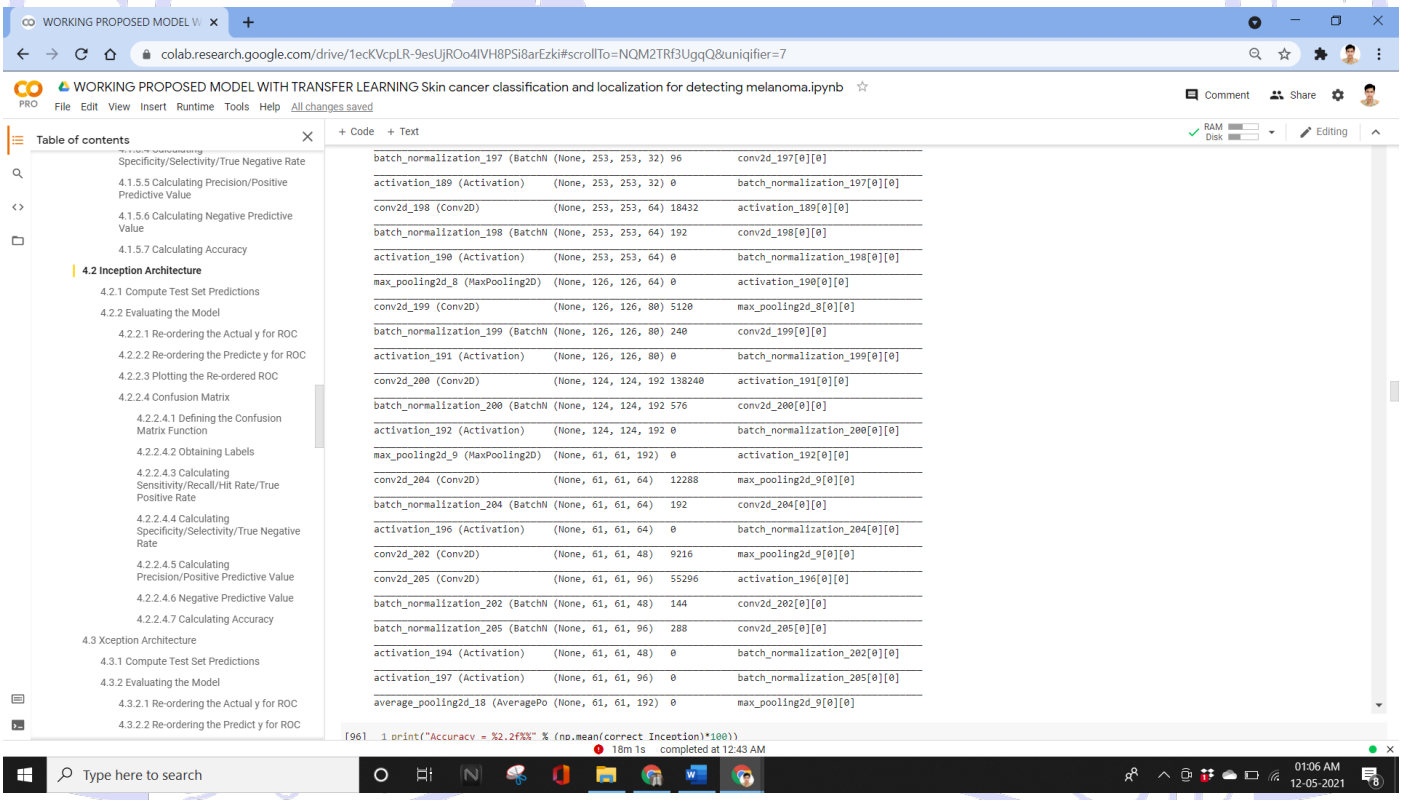

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
+ Code + Text
RAM Disk
Editing
True Positive Rate
False Positive Rate
ROC curve (area = 0.50)
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
[85]
def positive_negative_measurement(y_true, y_score):
# Initialization
TRUE_POSITIVE = 0
FALSE_POSITIVE = 0
TRUE_NEGATIVE = 0
FALSE_NEGATIVE = 0
# Calculating the model
for i in range(len(y_score)):
if y_true[i] == y_score[i] == 1:
TRUE_POSITIVE += 1
if (y_score[i] == 1) and (y_true[i] != y_score[i]):
FALSE_POSITIVE += 1
if y_true[i] == y_score[i] == 0:
TRUE_NEGATIVE += 1
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

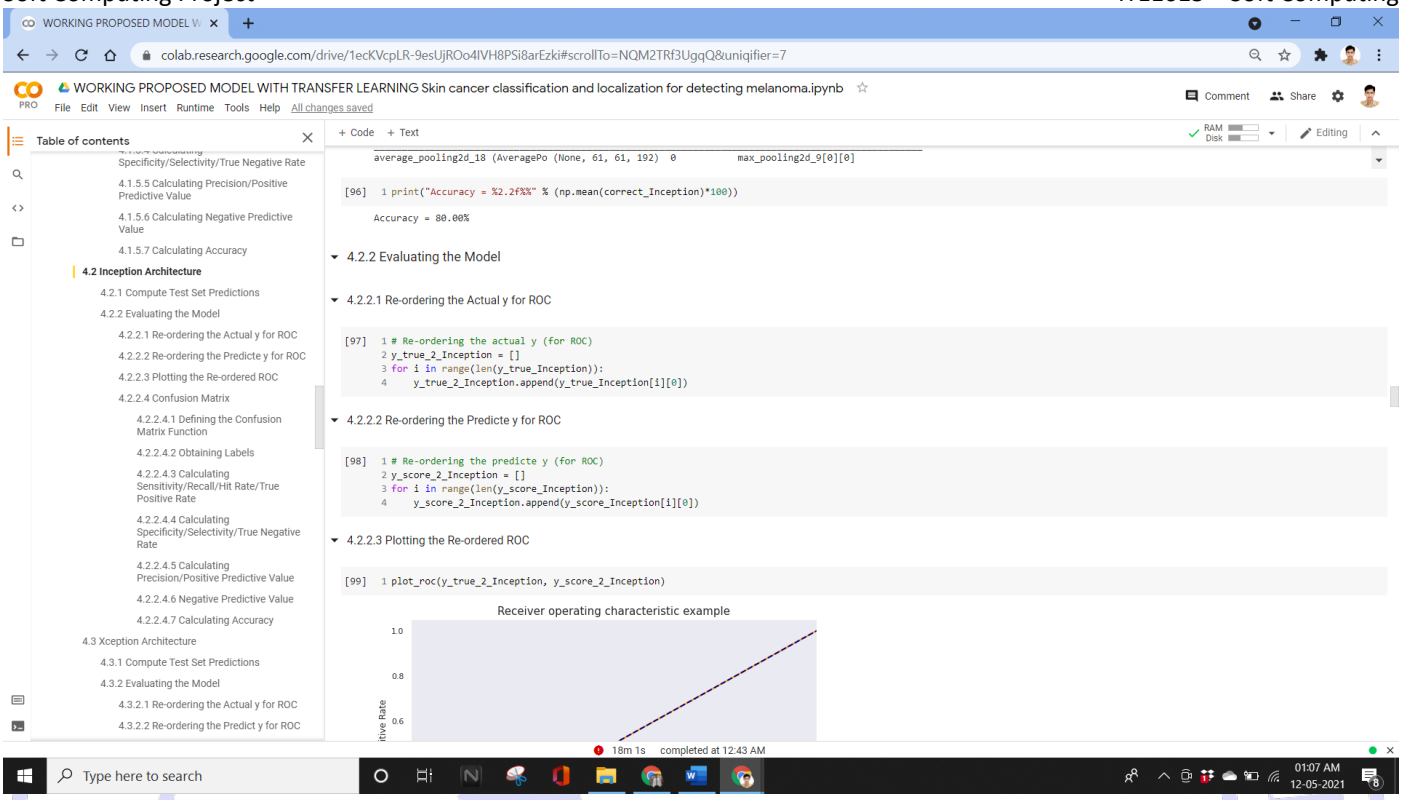

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
FALSE_POSITIVE += 1
if y_true[i] == y_score[i] == 0:
TRUE_NEGATIVE += 1
if (y_score[i] == 0) and (y_true[i] != y_score[i]):
FALSE_NEGATIVE += 1
return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[86]
TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet, TRUE_NEGATIVE_MobileNet, FALSE_NEGATIVE_MobileNet = positive_negative_measurement(y_true_2_MobileNet, y_score_2_MobileNet)
postives_negatives_MobileNet = [[TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet],
[FALSE_NEGATIVE_MobileNet, TRUE_NEGATIVE_MobileNet]]
postives_negatives_MobileNet
[[120, 30], [0, 0]]
4.1.5.2 Obtaining Labels
[88]
import seaborn as sns
sns.set()
labels_MobileNet = np.array([['True positive: ' + str(TRUE_POSITIVE_MobileNet),
'False positive: ' + str(FALSE_POSITIVE_MobileNet)],
['False negative: ' + str(FALSE_NEGATIVE_MobileNet),
'True negative: ' + str(TRUE_NEGATIVE_MobileNet)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_MobileNet, annot = labels_MobileNet, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f467b67ba10>
True positive: 120
False positive: 30
120
100
80
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

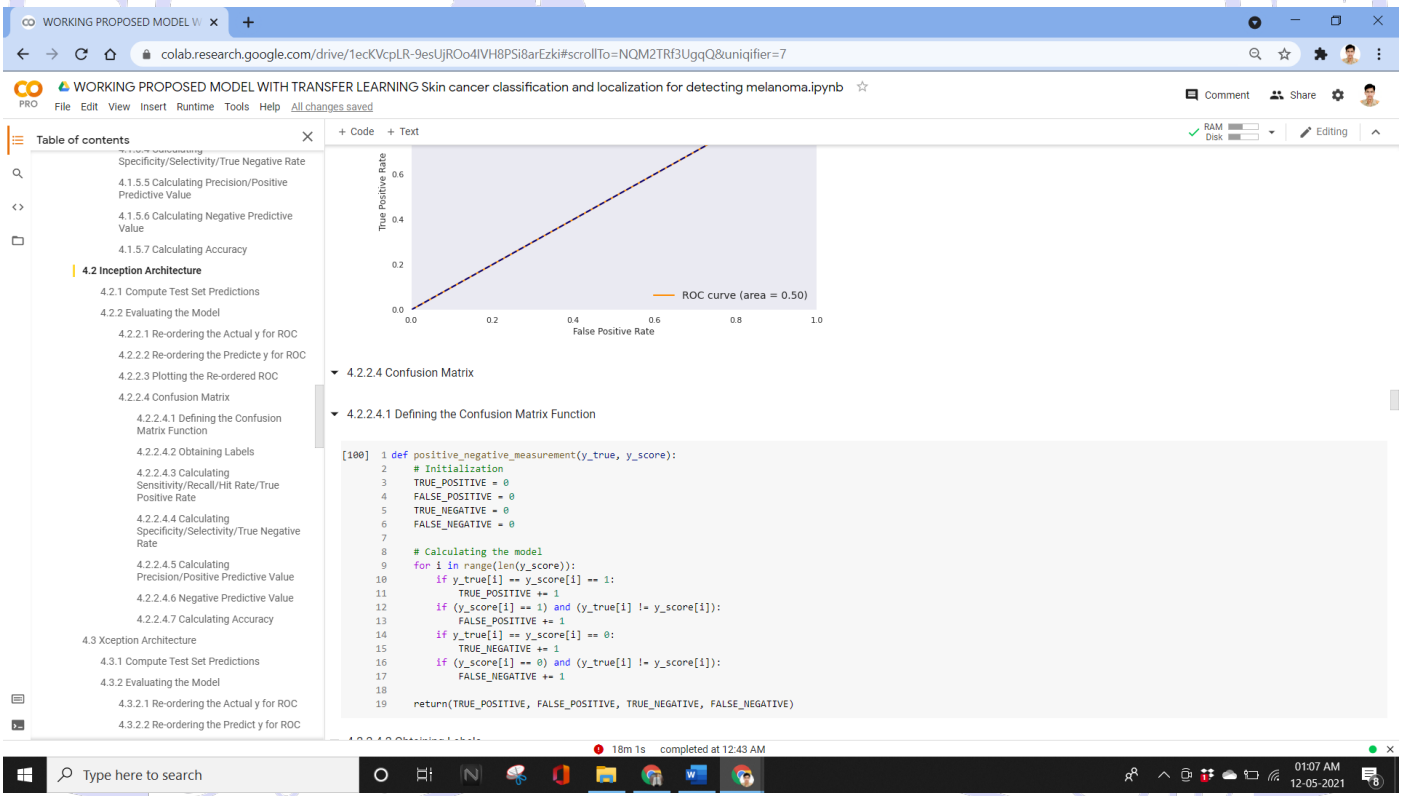

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
2.3.4 Loading the Weights
Prediction
Declaring some Variables
Defining the Prediction Function
3.1 MobileNet Architecture
3.1.1 Loading the Model and Weights
3.1.3 Predicting for a Sample Image Using MobileNet
3.2 Inception Architecture
3.2.1 Loading the Model and Weights
3.2.2 Predicting for a Sample Image Using InceptionV3
3.3 Xception Architecture
3.3.1 Loading the Model and Weights
3.3.2 Predicting a Sample Image using Xception
Evaluating the Models Individually on Validation Data
Defining Function to calculatae Receiving Operating Characteristic curve
Defining Function for Plotting the Receiving Operating Characteristic curve
4.1 MobileNet Architecture
4.1.1 Computing Test Set Predictions
4.1.2 Re-ordering Actual y for ROC
4.1.3 Re-ordering Predicte y for ROC
4.1.4 Plotting the Re-ordered ROC
4.1.5 Confusion Matrix
4.1.5.1 Defining the Confusion Matrix Function
4.1.5.2 Obtaining Labels
+ Code
+ Text
RAM
Disk
Editing
False negative: 0
True negative: 0
[89] 1 labels_MobileNet
array([['True positive: 120', 'False positive: 30'],
['False negative: 0', 'True negative: 0']], dtype='<U18')
4.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[90] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
3 print("Sensitivity: ", sensitivity_MobileNet)
Sensitivity: 1.0
4.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
[91] 1 # Specificity | selectivity | true negative rate (TNR)
18m 1s completed at 12:43 AM
Type here to search
01:05 AM
12-05-2021

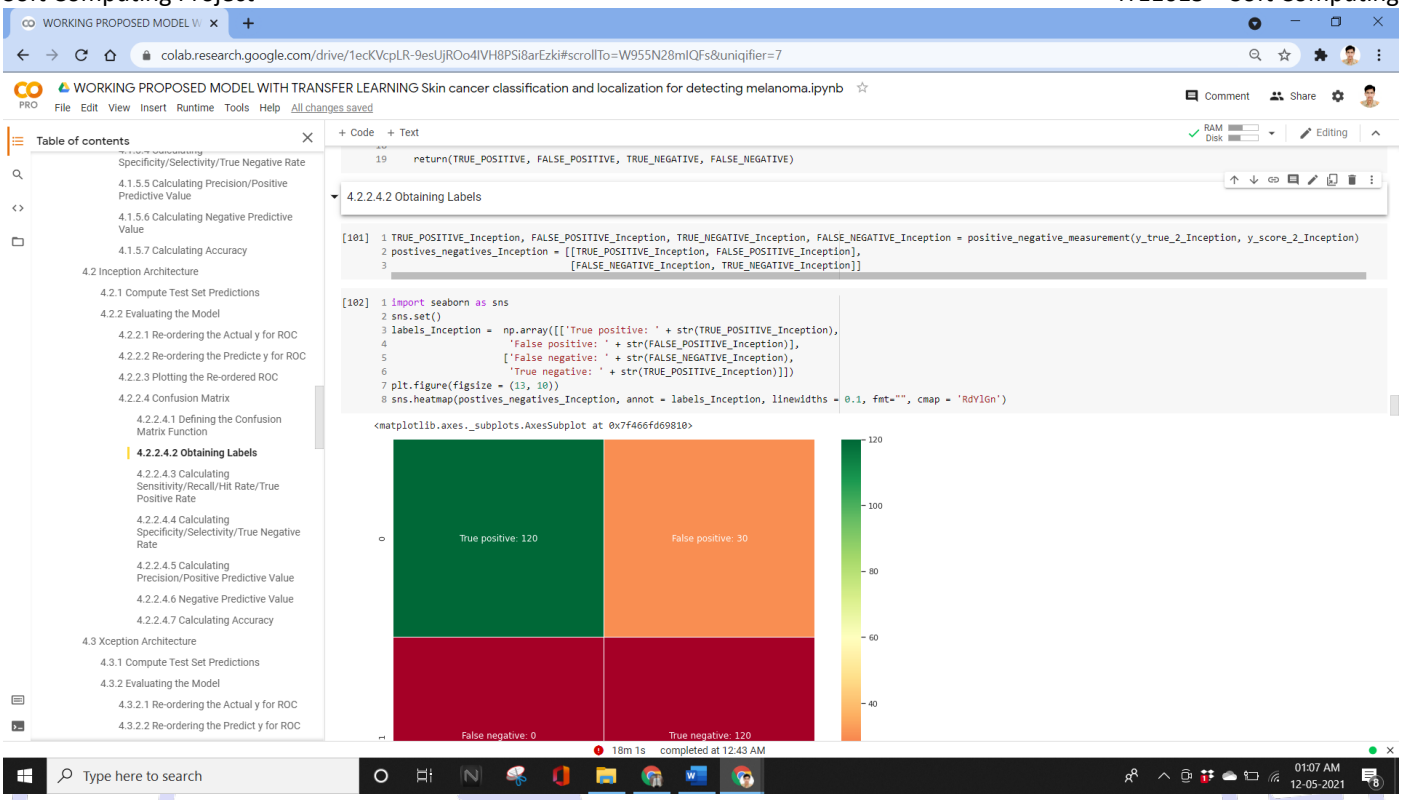

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=JfKTmJ_oCe6J&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code
+ Text
RAM
Disk
Editing
[91] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
4 print("Specifity: ", specifity_MobileNet)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
4.1.5.5 Calculating Precision/Positive Predictive Value
1 # Precision | positive predictive value (PPV)
2 predcision_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet + FALSE_POSITIVE_MobileNet)
3 print("Precision: ", predcision_MobileNet)
Precision: 0.8
4.1.5.6 Calculating Negative Predictive Value
[93] 1 # Negative predictive value (NPV)
2 try:
3 npv_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
4 print("Negative predictive value: ", npv_MobileNet)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
4.1.5.7 Calculating Accuracy
[94] 1 # Accuracy
2 accuracy_MobileNet = (TRUE_POSITIVE_MobileNet + TRUE_NEGATIVE_MobileNet) / (TRUE_POSITIVE_MobileNet + FALSE_POSITIVE_MobileNet + TRUE_NEGATIVE_MobileNet + FALSE_NEGATIVE_MobileN
3 print("Accuracy: ", accuracy_MobileNet)
Accuracy: 0.8
18m 1s completed at 12:43 AM
Type here to search
01:06 AM
12-05-2021

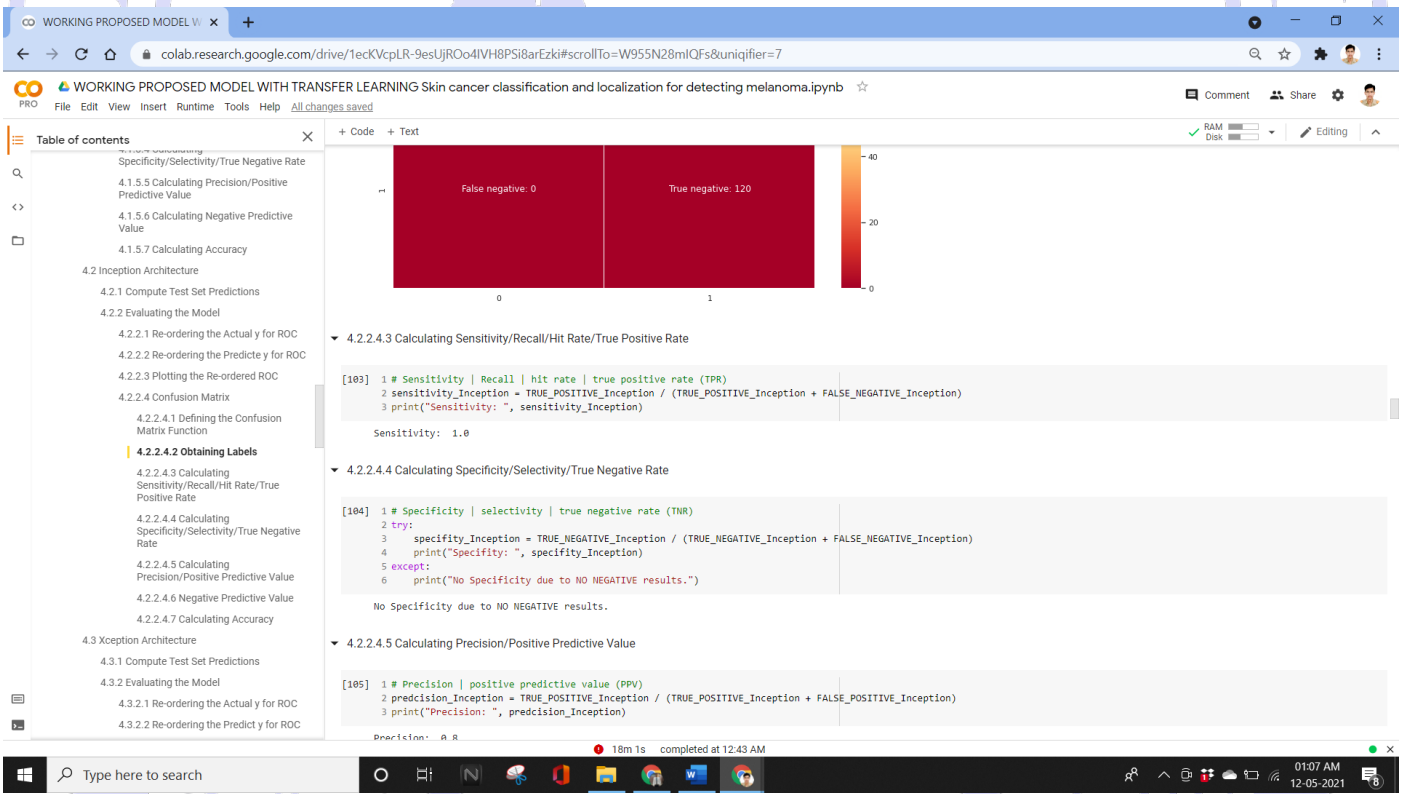

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=NQM2TRf3UgqQ&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Specificity/Selectivity/True Negative Rate
4.1.5.5 Calculating Precision/Positive Predictive Value
4.1.5.6 Calculating Negative Predictive Value
4.1.5.7 Calculating Accuracy
4.2 Inception Architecture
4.2.1 Compute Test Set Predictions
4.2.2 Evaluating the Model
4.2.2.1 Re-ordering the Actual y for ROC
4.2.2.2 Re-ordering the Predicte y for ROC
4.2.2.3 Plotting the Re-ordered ROC
4.2.2.4 Confusion Matrix
4.2.2.4.1 Defining the Confusion Matrix Function
4.2.2.4.2 Obtaining Labels
4.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.2.2.4.5 Calculating Precision/Positive Predictive Value
4.2.2.4.6 Negative Predictive Value
4.2.2.4.7 Calculating Accuracy
4.3 Xception Architecture
4.3.1 Compute Test Set Predictions
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
Accuracy: 0.8
4.2 Inception Architecture
4.2.1 Compute Test Set Predictions
[95] 1 # Compute test set predictions
2 #model_architecture,path_model_weight
3 NUMBER_TEST_SAMPLES_Inception = 150
4
5 inception_architecture_function = inception_architecture()
6 inception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5"
7
8 y_true_Inception = valid_targets[:NUMBER_TEST_SAMPLES_Inception]
9 y_score_Inception = []
10 for index in range(NUMBER_TEST_SAMPLES_Inception): #compute one at a time due to memory constraints
11 probs_Inception = predict(img_path = validation_files[index], model_architecture = inception_architecture_function, path_model_weight = inception_architecture_weight_path)
12 print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0], " | Other : ", valid_targets[index][1])
13 print("----------")
14 y_score_Inception.append(probs_Inception)
15
16 correct_Inception = np.array(y_true_Inception) == np.array(y_score_Inception)
Model: "model_7"
Layer (type) Output Shape Param # Connected to
input_8 (InputLayer) [(None, 512, 512, 3) 0
conv2d_196 (Conv2D) (None, 255, 255, 32) 864 input_8[0][0]
batch_normalization_196 (BatchN (None, 255, 255, 32) 96 conv2d_196[0][0]
activation_188 (Activation) (None, 255, 255, 32) 0 batch_normalization_196[0][0]
conv2d_197 (Conv2D) (None, 253, 253, 32) 9216 activation_188[0][0]
batch_normalization_197 (BatchN (None, 253, 253, 32) 96 conv2d_197[0][0]
activation_189 (Activation) (None, 253, 253, 32) 0 batch_normalization_197[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:06 AM
12-05-2021

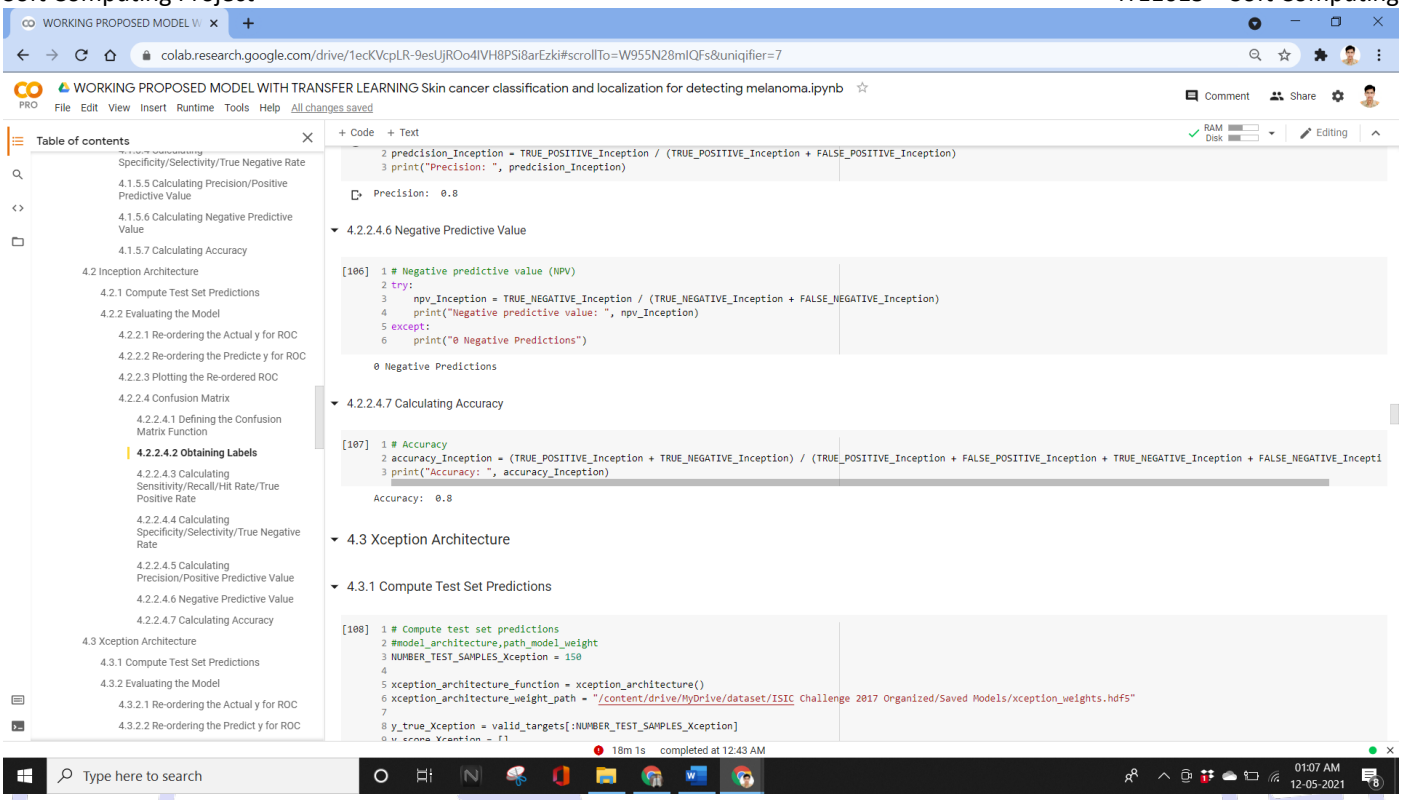

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=NQM2TRf3UgqQ&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Specificity/Selectivity/True Negative Rate
4.1.5.5 Calculating Precision/Positive Predictive Value
4.1.5.6 Calculating Negative Predictive Value
4.1.5.7 Calculating Accuracy
4.2 Inception Architecture
4.2.1 Compute Test Set Predictions
4.2.2 Evaluating the Model
4.2.2.1 Re-ordering the Actual y for ROC
4.2.2.2 Re-ordering the Predicte y for ROC
4.2.2.3 Plotting the Re-ordered ROC
4.2.2.4 Confusion Matrix
4.2.2.4.1 Defining the Confusion Matrix Function
4.2.2.4.2 Obtaining Labels
4.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.2.2.4.5 Calculating Precision/Positive Predictive Value
4.2.2.4.6 Negative Predictive Value
4.2.2.4.7 Calculating Accuracy
4.3 Xception Architecture
4.3.1 Compute Test Set Predictions
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
batch_normalization_197 (BatchN (None, 253, 253, 32) 96 conv2d_197[0][0]
activation_189 (Activation) (None, 253, 253, 32) 0 batch_normalization_197[0][0]
conv2d_198 (Conv2D) (None, 253, 253, 64) 18432 activation_189[0][0]
batch_normalization_198 (BatchN (None, 253, 253, 64) 192 conv2d_198[0][0]
activation_190 (Activation) (None, 253, 253, 64) 0 batch_normalization_198[0][0]
max_pooling2d_8 (MaxPooling2D) (None, 126, 126, 64) 0 activation_190[0][0]
conv2d_199 (Conv2D) (None, 126, 126, 80) 5120 max_pooling2d_8[0][0]
batch_normalization_199 (BatchN (None, 126, 126, 80) 240 conv2d_199[0][0]
activation_191 (Activation) (None, 126, 126, 80) 0 batch_normalization_199[0][0]
conv2d_200 (Conv2D) (None, 124, 124, 192 138240 activation_191[0][0]
batch_normalization_200 (BatchN (None, 124, 124, 192 576 conv2d_200[0][0]
activation_192 (Activation) (None, 124, 124, 192 0 batch_normalization_200[0][0]
max_pooling2d_9 (MaxPooling2D) (None, 61, 61, 192) 0 activation_192[0][0]
conv2d_204 (Conv2D) (None, 61, 61, 64) 12288 max_pooling2d_9[0][0]
batch_normalization_204 (BatchN (None, 61, 61, 64) 192 conv2d_204[0][0]
activation_196 (Activation) (None, 61, 61, 64) 0 batch_normalization_204[0][0]
conv2d_202 (Conv2D) (None, 61, 61, 48) 9216 max_pooling2d_9[0][0]
conv2d_205 (Conv2D) (None, 61, 61, 96) 55296 activation_196[0][0]
batch_normalization_202 (BatchN (None, 61, 61, 48) 144 conv2d_202[0][0]
batch_normalization_205 (BatchN (None, 61, 61, 96) 288 conv2d_205[0][0]
activation_194 (Activation) (None, 61, 61, 48) 0 batch_normalization_202[0][0]
activation_197 (Activation) (None, 61, 61, 96) 0 batch_normalization_205[0][0]
average_pooling2d_18 (AveragePo (None, 61, 61, 192) 0 max_pooling2d_9[0][0]
[96] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Inception)*100))
18m 1s completed at 12:43 AM
Type here to search
01:06 AM
12-05-2021

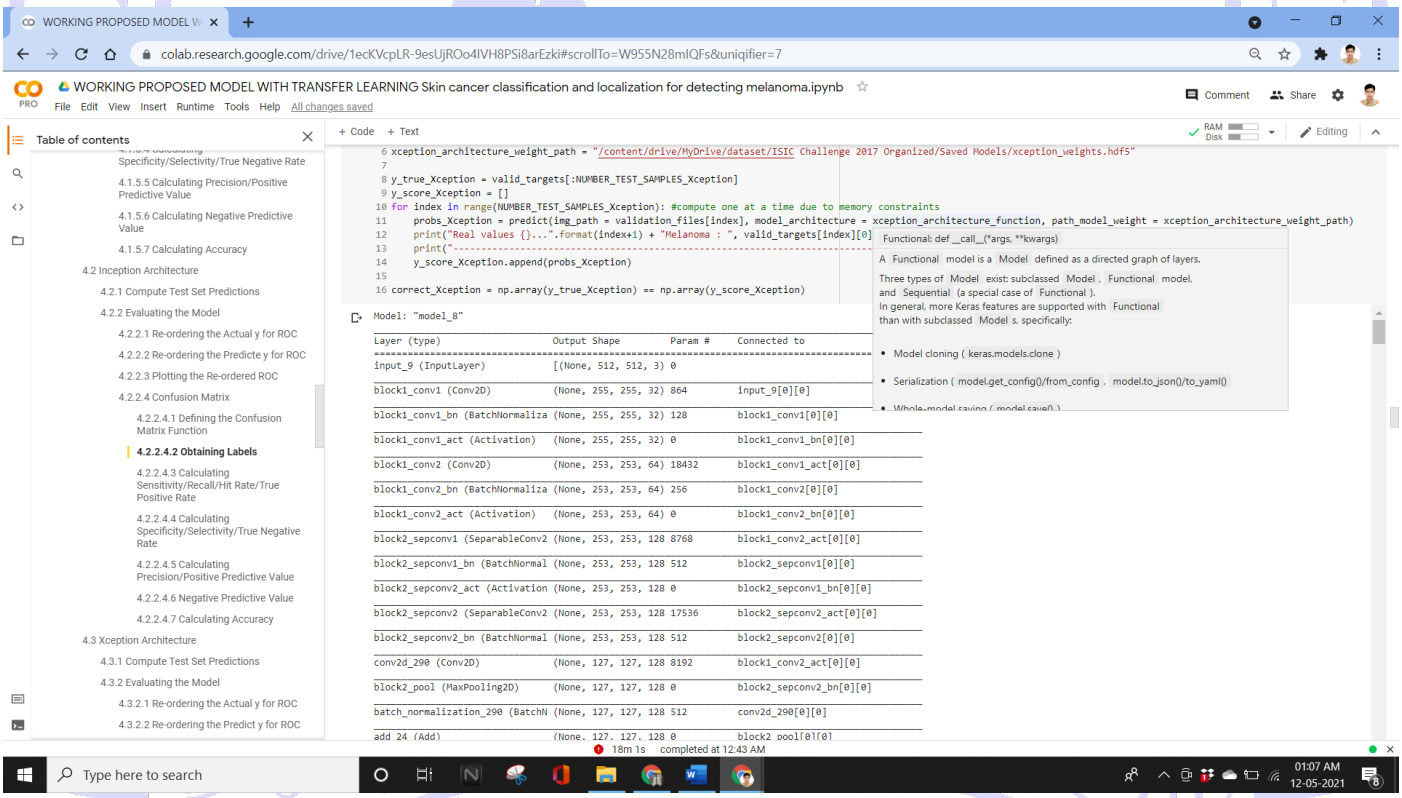

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=NQM2TRf3UgqQ&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
average_pooling2d_18 (AveragePo (None, 61, 61, 192) 0 max_pooling2d_9[0][0]
[96] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Inception)*100))
Accuracy = 80.00%
4.2.2 Evaluating the Model
4.2.2.1 Re-ordering the Actual y for ROC
[97] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Inception = []
3 for i in range(len(y_true_Inception)):
4 y_true_2_Inception.append(y_true_Inception[i][0])
4.2.2.2 Re-ordering the Predicte y for ROC
[98] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Inception = []
3 for i in range(len(y_score_Inception)):
4 y_score_2_Inception.append(y_score_Inception[i][0])
4.2.2.3 Plotting the Re-ordered ROC
[99] 1 plot_roc(y_true_2_Inception, y_score_2_Inception)
Receiver operating characteristic example
1.0
0.8
0.6
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

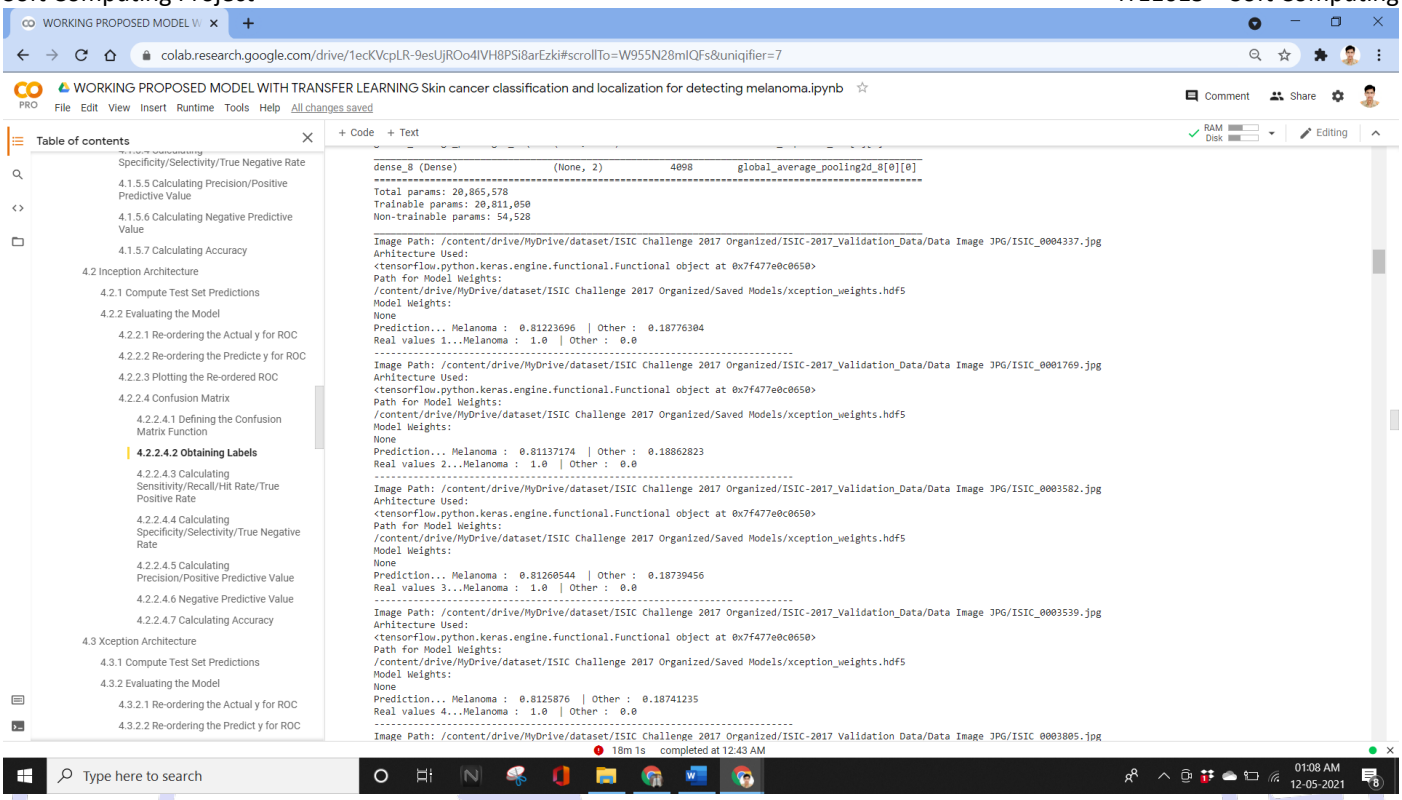

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=NQM2TRf3UgqQ&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
True Positive Rate
0.6
0.4
0.2
0.0
ROC curve (area = 0.50)
0.0 0.2 0.4 0.6 0.8 1.0
False Positive Rate
4.2.2.4 Confusion Matrix
4.2.2.4.1 Defining the Confusion Matrix Function
[100] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

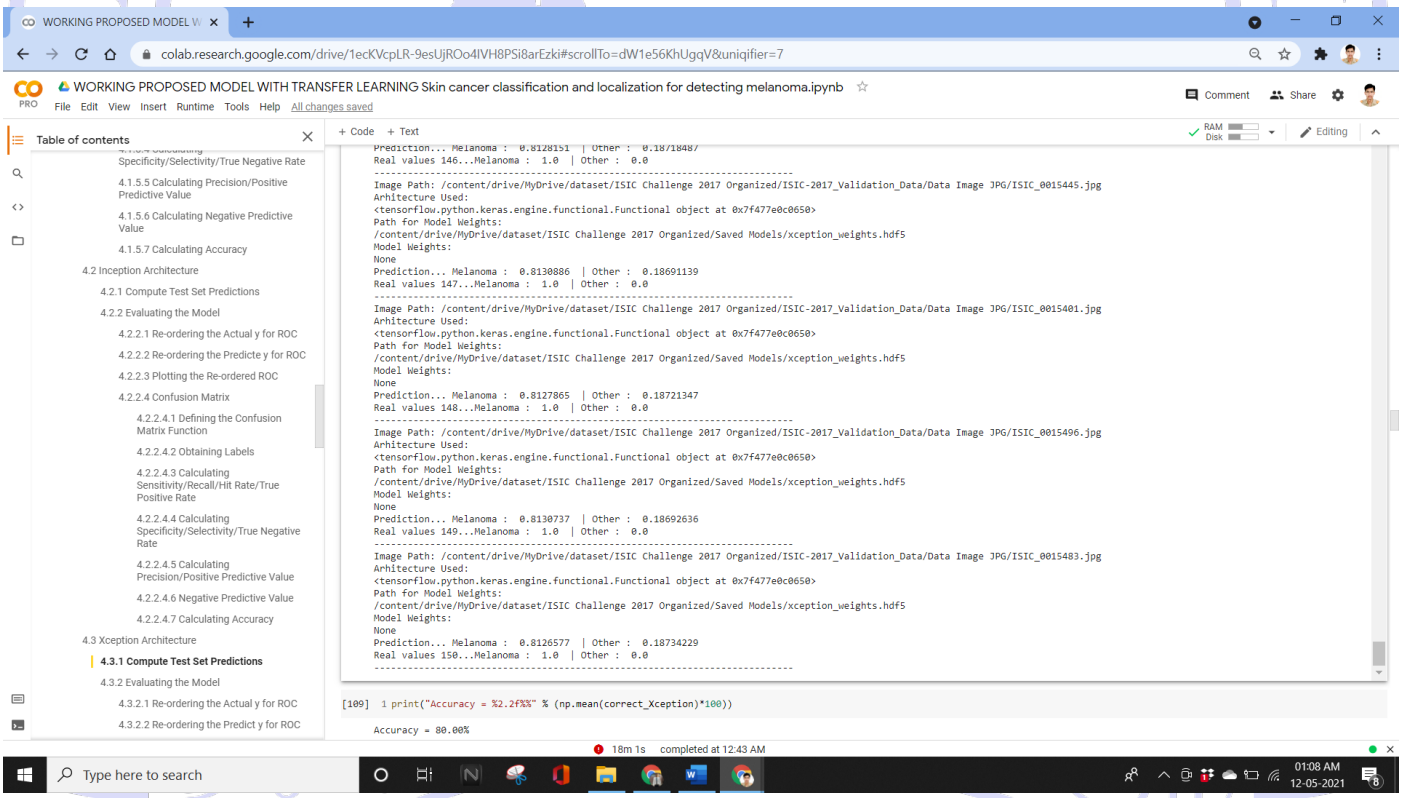

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=W955N28mIQFs&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
4.2.2.4.2 Obtaining Labels
[101] 1 TRUE_POSITIVE_Inception, FALSE_POSITIVE_Inception, TRUE_NEGATIVE_Inception, FALSE_NEGATIVE_Inception = positive_negative_measurement(y_true_2_Inception, y_score_2_Inception)
2 postives_negatives_Inception = [[TRUE_POSITIVE_Inception, FALSE_POSITIVE_Inception],
3 [FALSE_NEGATIVE_Inception, TRUE_NEGATIVE_Inception]]
[102] 1 import seaborn as sns
2 sns.set()
3 labels_Inception = np.array([['True positive: ' + str(TRUE_POSITIVE_Inception),
4 'False positive: ' + str(FALSE_POSITIVE_Inception)],
5 ['False negative: ' + str(FALSE_NEGATIVE_Inception),
6 'True negative: ' + str(TRUE_POSITIVE_Inception)]])
7 plt.figure(figsize = (13, 10))
8 sns.heatmap(postives_negatives_Inception, annot = labels_Inception, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f466fd69810>
True positive: 120
False positive: 30
False negative: 0
True negative: 120
120
100
80
60
40
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

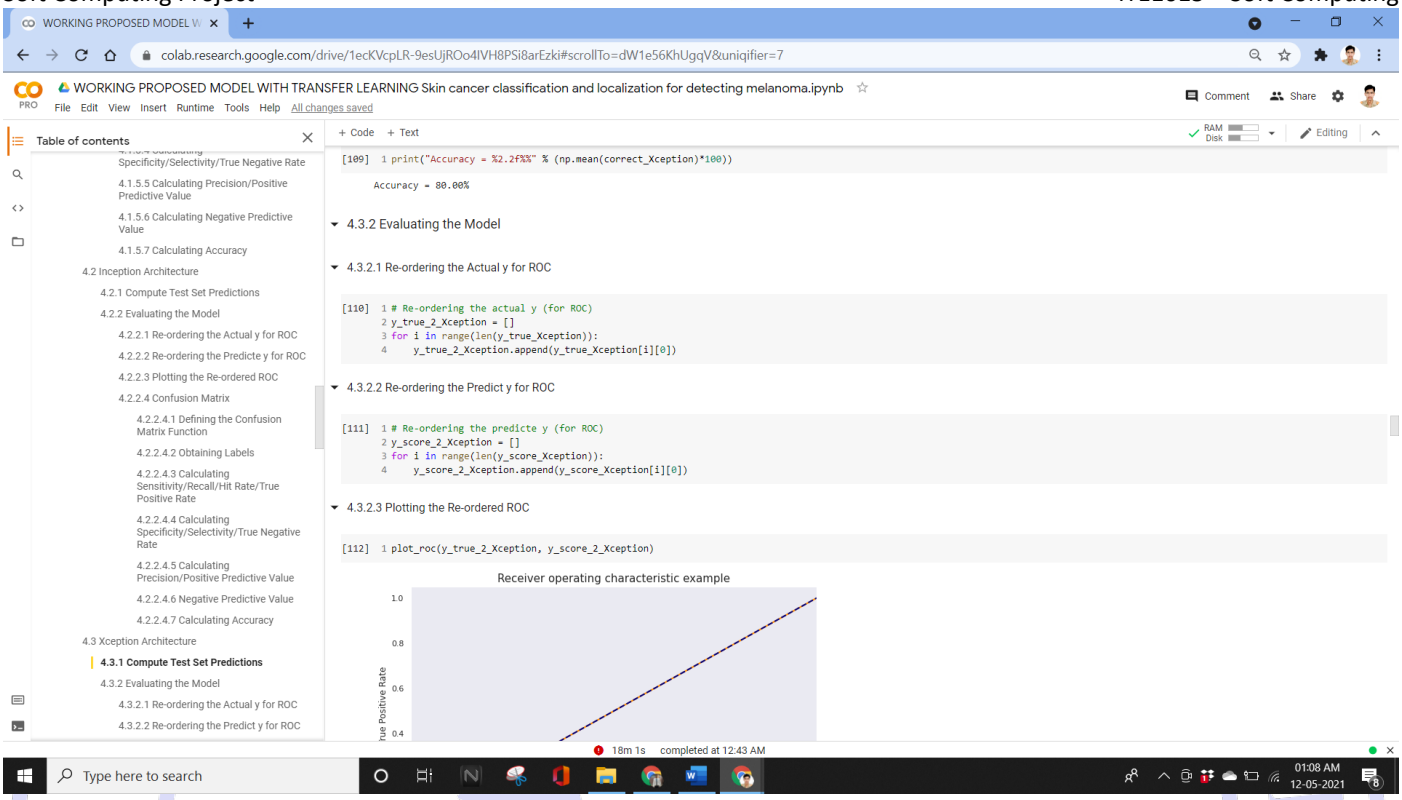

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=W955N28mIQFs&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
False negative: 0
True negative: 120
40
20
0
0
1
4.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[103] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception + FALSE_NEGATIVE_Inception)
3 print("Sensitivity: ", sensitivity_Inception)
Sensitivity: 1.0
4.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[104] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception + FALSE_NEGATIVE_Inception)
4 print("Specifity: ", specifity_Inception)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
4.2.2.4.5 Calculating Precision/Positive Predictive Value
[105] 1 # Precision | positive predictive value (PPV)
2 predcision_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception)
3 print("Precision: ", predcision_Inception)
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

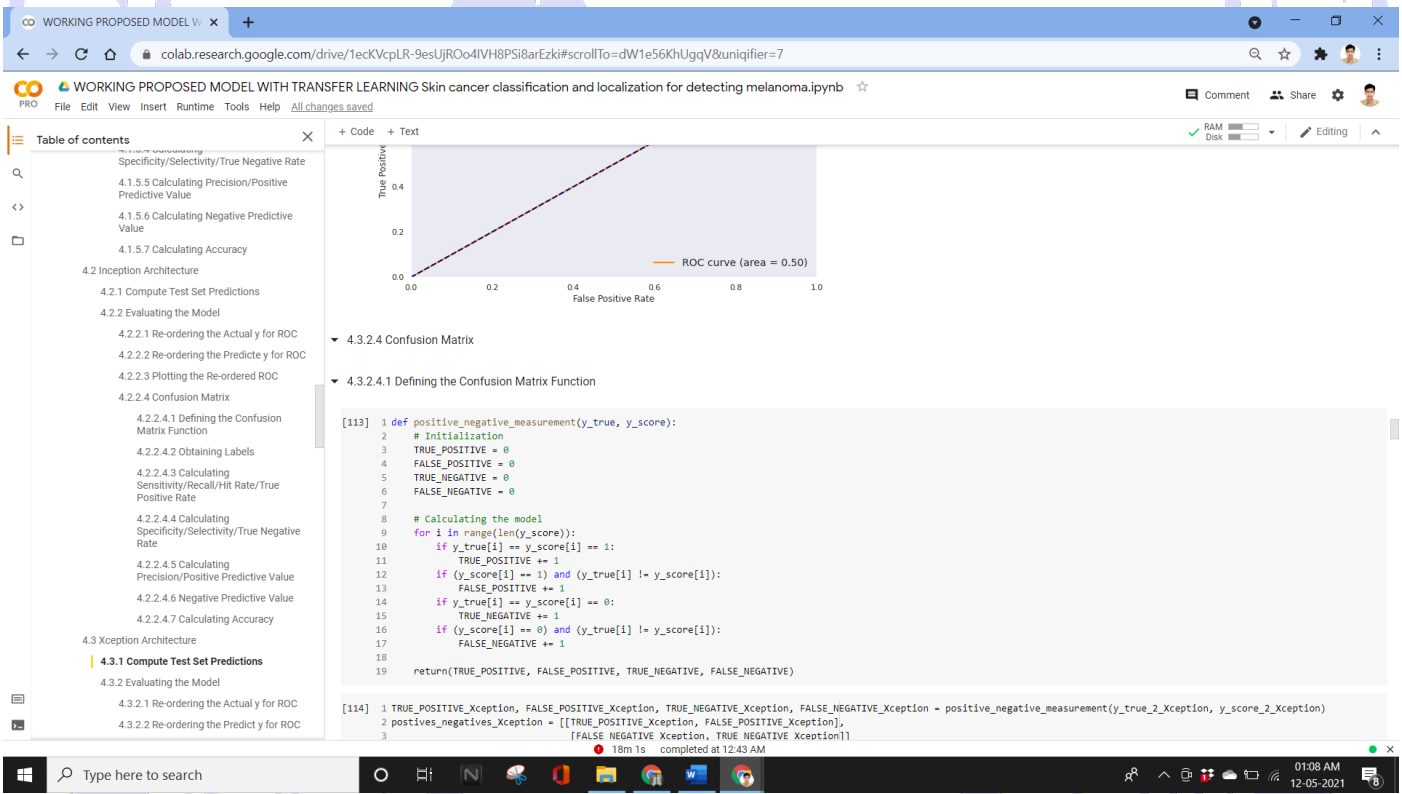

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=W955N28mIQFs&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
predcision_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception)
print("Precision: ", predcision_Inception)
Precision: 0.8
4.2.2.4.6 Negative Predictive Value
[106]
# Negative predictive value (NPV)
try:
npv_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception + FALSE_NEGATIVE_Inception)
print("Negative predictive value: ", npv_Inception)
except:
print("0 Negative Predictions")
0 Negative Predictions
4.2.2.4.7 Calculating Accuracy
[107]
# Accuracy
accuracy_Inception = (TRUE_POSITIVE_Inception + TRUE_NEGATIVE_Inception) / (TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception + TRUE_NEGATIVE_Inception + FALSE_NEGATIVE_Inceptio
print("Accuracy: ", accuracy_Inception)
Accuracy: 0.8
4.3 Xception Architecture
4.3.1 Compute Test Set Predictions
[108]
Compute test set predictions
#model_architecture,path_model_weight
NUMBER_TEST_SAMPLES_Xception = 150
xception_architecture_function = xception_architecture()
xception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5"
y_true_Xception = valid_targets[:NUMBER_TEST_SAMPLES_Xception]
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

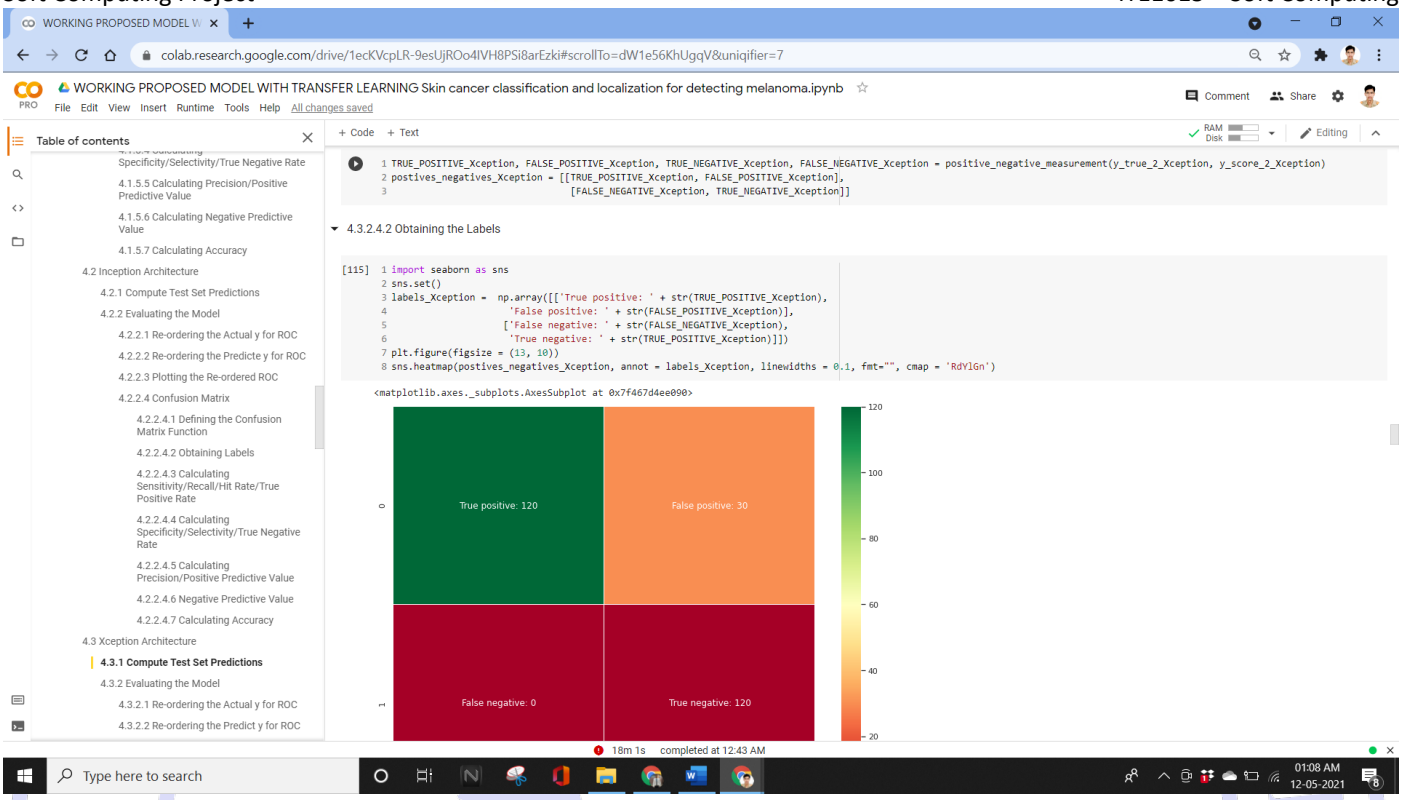

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=W955N28mIQFs&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
xception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5"
y_true_Xception = valid_targets[:NUMBER_TEST_SAMPLES_Xception]
y_score_Xception = []
for index in range(NUMBER_TEST_SAMPLES_Xception): #compute one at a time due to memory constraints
probs_Xception = predict(img_path = validation_files[index], model_architecture = xception_architecture_function, path_model_weight = xception_architecture_weight_path)
print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0])
print("----------")
y_score_Xception.append(probs_Xception)
correct_Xception = np.array(y_true_Xception) == np.array(y_score_Xception)
Functional: def __call__(*args, **kwargs)
A Functional model is a Model defined as a directed graph of layers.
Three types of Model exist: subclassed Model , Functional model,
and Sequential (a special case of Functional).
In general, more Keras features are supported with Functional
than with subclassed Model s. specifically:
Model cloning (keras.models.clone)
Serialization (model.get_config()/from_config , model.to_json()/to_yaml()
Model: "model_8"
Layer (type) Output Shape Param # Connected to
input_9 (InputLayer) [(None, 512, 512, 3) 0
block1_conv1 (Conv2D) (None, 255, 255, 32) 864 input_9[0][0]
block1_conv1_bn (BatchNormaliza (None, 255, 255, 32) 128 block1_conv1[0][0]
block1_conv1_act (Activation) (None, 255, 255, 32) 0 block1_conv1_bn[0][0]
block1_conv2 (Conv2D) (None, 253, 253, 64) 18432 block1_conv1_act[0][0]
block1_conv2_bn (BatchNormaliza (None, 253, 253, 64) 256 block1_conv2[0][0]
block1_conv2_act (Activation) (None, 253, 253, 64) 0 block1_conv2_bn[0][0]
block2_sepconv1 (SeparableConv2 (None, 253, 253, 128 8768 block1_conv2_act[0][0]
block2_sepconv1_bn (BatchNormal (None, 253, 253, 128 512 block2_sepconv1[0][0]
block2_sepconv2_act (Activation (None, 253, 253, 128 0 block2_sepconv1_bn[0][0]
block2_sepconv2 (SeparableConv2 (None, 253, 253, 128 17536 block2_sepconv2_act[0][0]
block2_sepconv2_bn (BatchNormal (None, 253, 253, 128 512 block2_sepconv2[0][0]
conv2d_290 (Conv2D) (None, 127, 127, 128 8192 block1_conv2_act[0][0]
block2_pool (MaxPooling2D) (None, 127, 127, 128 0 block2_sepconv2_bn[0][0]
batch_normalization_290 (BatchN (None, 127, 127, 128 512 conv2d_290[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:07 AM
12-05-2021

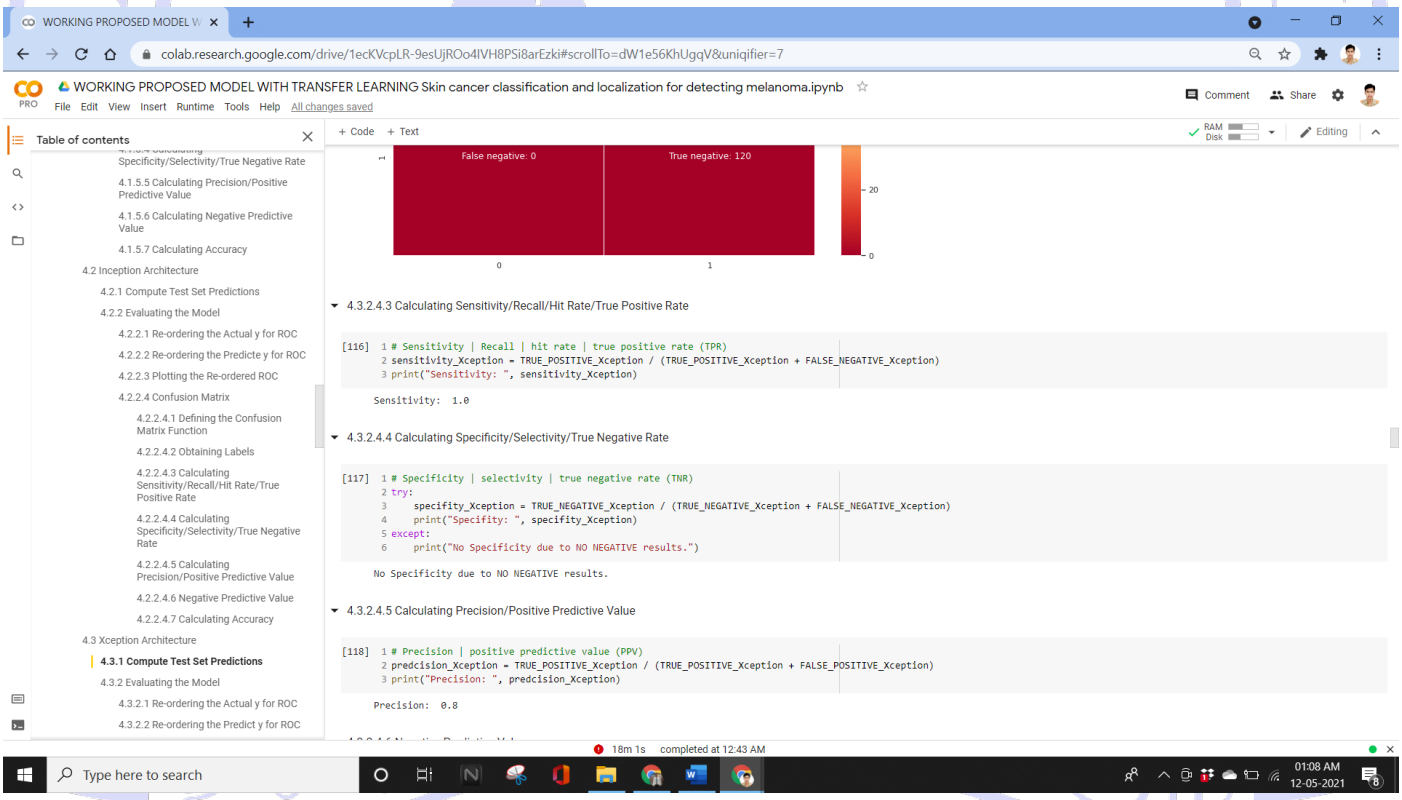

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=W955N28mlQFs&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
dense_8 (Dense) (None, 2) 4098 global_average_pooling2d_8[0][0]
Total params: 20,865,578
Trainable params: 20,811,050
Non-trainable params: 54,528
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0004337.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.81223696 | Other : 0.18776304
Real values 1...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.81137174 | Other : 0.18862823
Real values 2...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003582.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.81260544 | Other : 0.18739456
Real values 3...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003539.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8125876 | Other : 0.18741235
Real values 4...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0003805.jpg
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

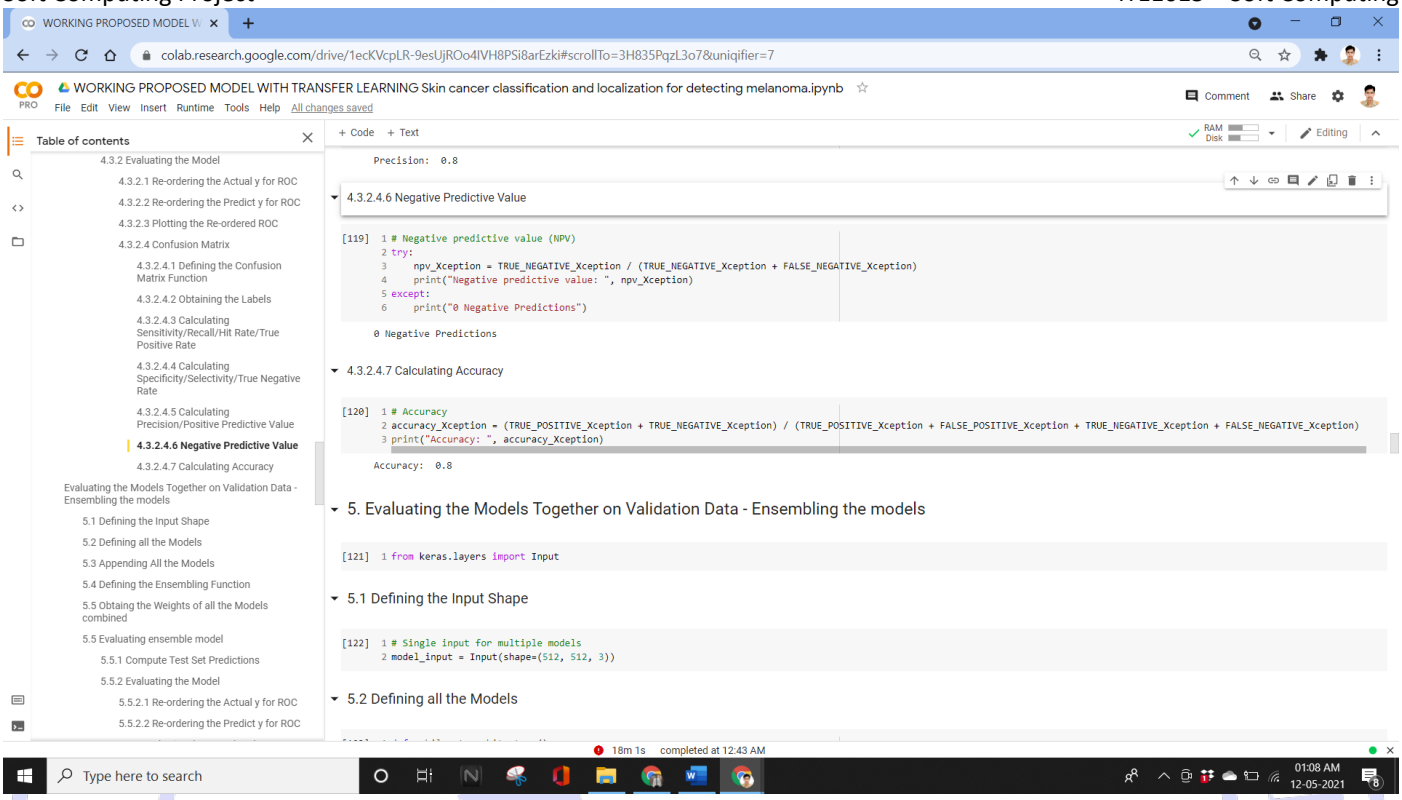

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=dW1e56KhUgqV&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Real values 146...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015445.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8130886 | Other : 0.18691139
Real values 147...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015401.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8127865 | Other : 0.18721347
Real values 148...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015496.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8130737 | Other : 0.18692636
Real values 149...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015483.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f477e0c0650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8126577 | Other : 0.18734229
Real values 150...Melanoma : 1.0 | Other : 0.0
[109] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Xception)*100))
Accuracy = 80.00%
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

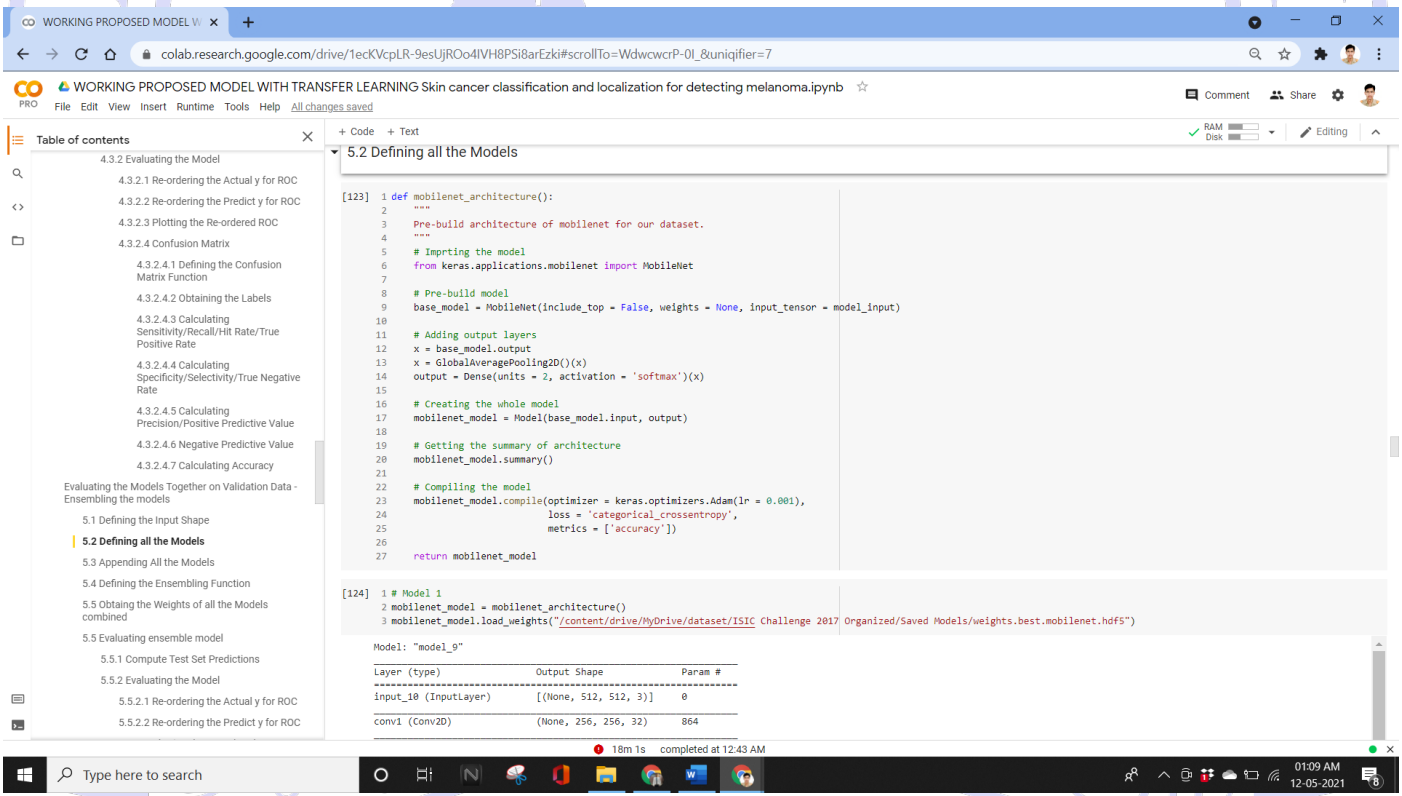

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=dW1e56KhUgqV&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
[109] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Xception)*100))
Accuracy = 80.00%
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
[110] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Xception = []
3 for i in range(len(y_true_Xception)):
4 y_true_2_Xception.append(y_true_Xception[i][0])
4.3.2.2 Re-ordering the Predict y for ROC
[111] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Xception = []
3 for i in range(len(y_score_Xception)):
4 y_score_2_Xception.append(y_score_Xception[i][0])
4.3.2.3 Plotting the Re-ordered ROC
[112] 1 plot_roc(y_true_2_Xception, y_score_2_Xception)
Receiver operating characteristic example
True Positive Rate
1.0
0.8
0.6
0.4
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

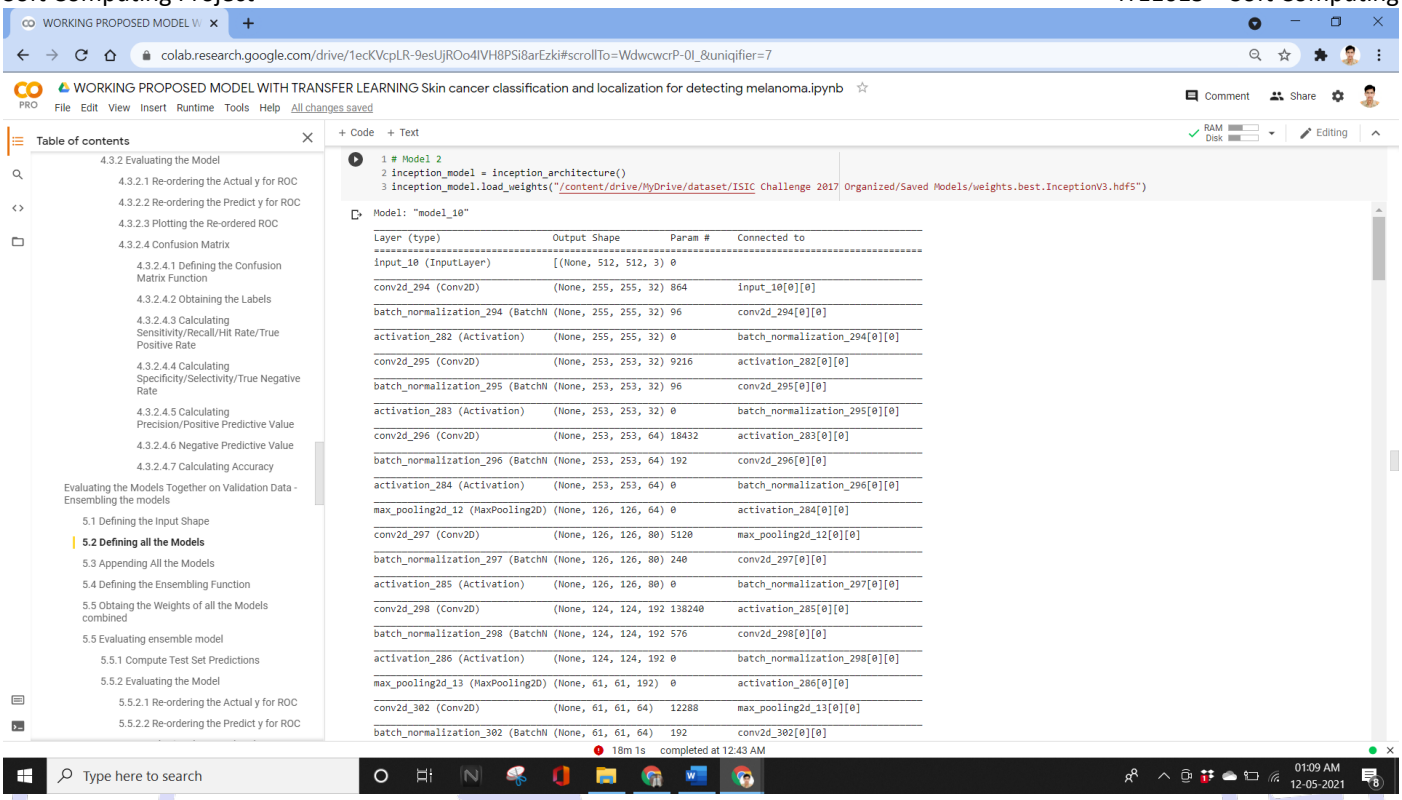

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=dW1e56KhUgqV&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
True Positive
0.4
0.2
0.0
ROC curve (area = 0.50)
0.0 0.2 0.4 0.6 0.8 1.0
False Positive Rate
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
[113] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[114] 1 TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception, TRUE_NEGATIVE_Xception, FALSE_NEGATIVE_Xception = positive_negative_measurement(y_true_2_Xception, y_score_2_Xception)
2 postives_negatives_Xception = [[TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception],
3 [FALSE_NEGATIVE_Xception, TRUE_NEGATIVE_Xception]]
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

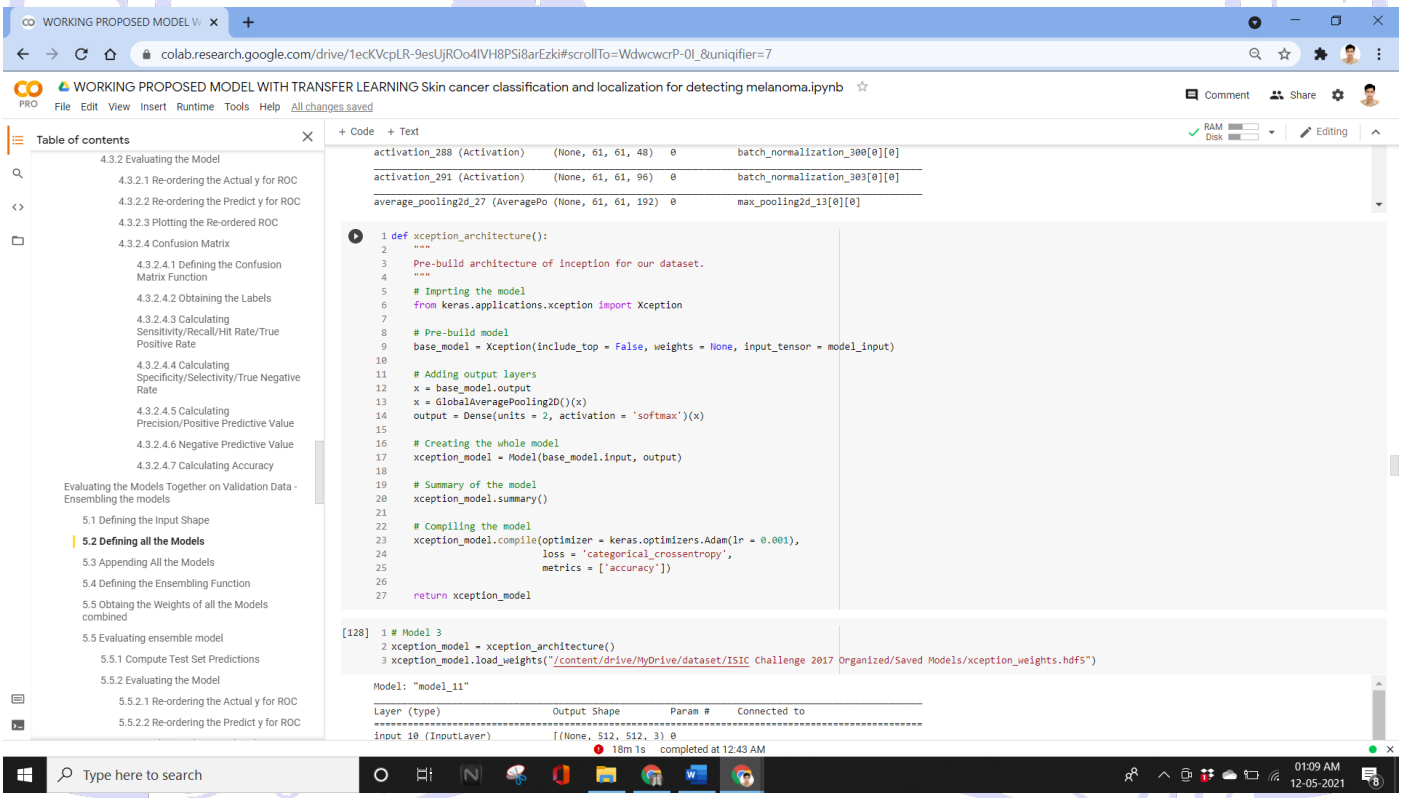

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=dW1e56KhUgqV&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
1 TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception, TRUE_NEGATIVE_Xception, FALSE_NEGATIVE_Xception = positive_negative_measurement(y_true_2_Xception, y_score_2_Xception)
2 postives_negatives_Xception = [[TRUE_POSITIVE_Xception, FALSE_POSITIVE_Xception],
3 [FALSE_NEGATIVE_Xception, TRUE_NEGATIVE_Xception]]
4.3.2.4.2 Obtaining the Labels
[115] 1 import seaborn as sns
2 sns.set()
3 labels_Xception = np.array([['True positive: ' + str(TRUE_POSITIVE_Xception),
4 'False positive: ' + str(FALSE_POSITIVE_Xception)],
5 ['False negative: ' + str(FALSE_NEGATIVE_Xception),
6 'True negative: ' + str(TRUE_POSITIVE_Xception)]])
7 plt.figure(figsize = (13, 10))
8 sns.heatmap(postives_negatives_Xception, annot = labels_Xception, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f467d4ee090>
True positive: 120
False positive: 30
False negative: 0
True negative: 120
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

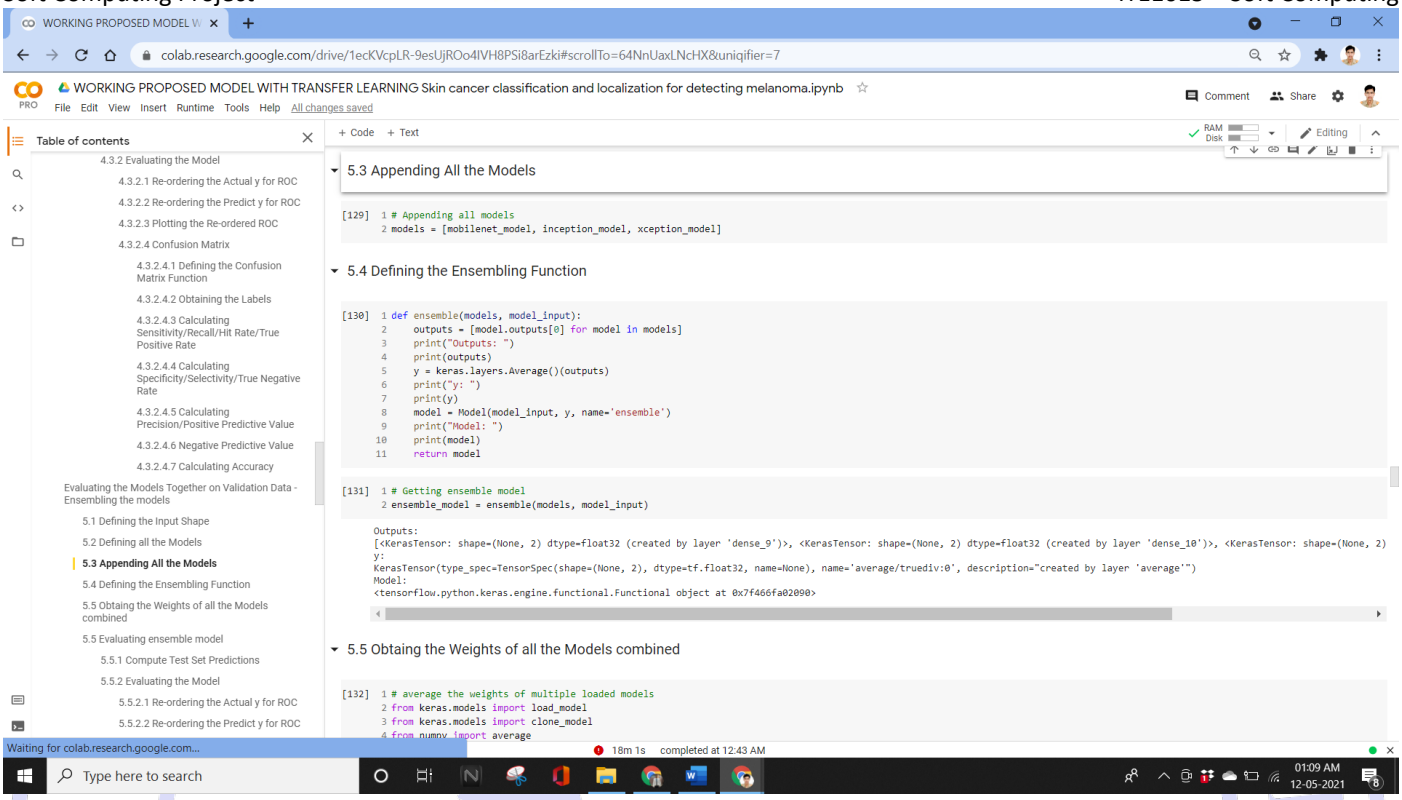

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=dW1e56KhUgqV&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
False negative: 0
True negative: 120
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[116] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception + FALSE_NEGATIVE_Xception)
3 print("Sensitivity: ", sensitivity_Xception)
Sensitivity: 1.0
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[117] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception + FALSE_NEGATIVE_Xception)
4 print("Specifity: ", specifity_Xception)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
4.3.2.4.5 Calculating Precision/Positive Predictive Value
[118] 1 # Precision | positive predictive value (PPV)
2 predcision_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception + FALSE_POSITIVE_Xception)
3 print("Precision: ", predcision_Xception)
Precision: 0.8
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

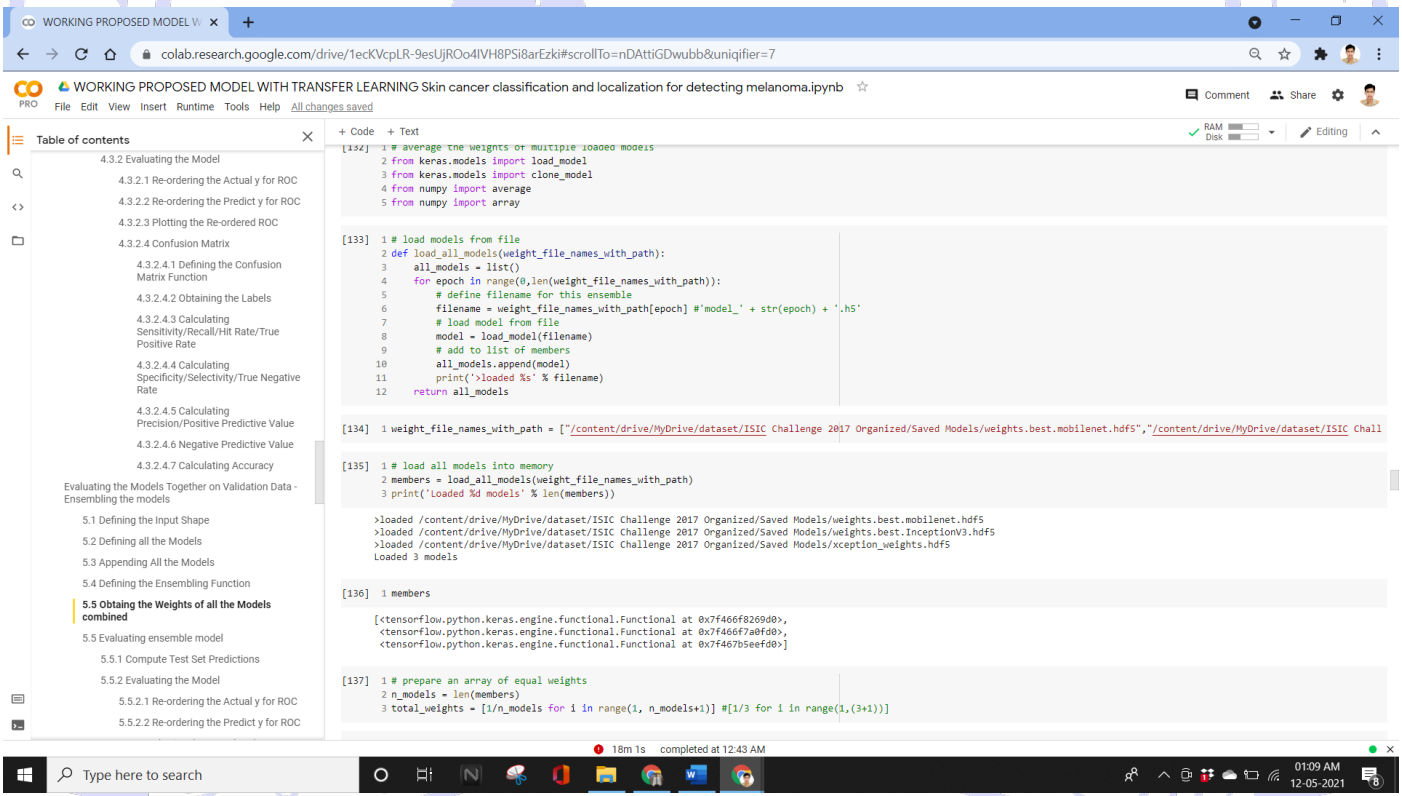

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=3H835PqzL3o7&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
Editing
Precision: 0.8
4.3.2.4.6 Negative Predictive Value
[119] 1 # Negative predictive value (NPV)
2 try:
3 npv_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception + FALSE_NEGATIVE_Xception)
4 print("Negative predictive value: ", npv_Xception)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
4.3.2.4.7 Calculating Accuracy
[120] 1 # Accuracy
2 accuracy_Xception = (TRUE_POSITIVE_Xception + TRUE_NEGATIVE_Xception) / (TRUE_POSITIVE_Xception + FALSE_POSITIVE_Xception + TRUE_NEGATIVE_Xception + FALSE_NEGATIVE_Xception)
3 print("Accuracy: ", accuracy_Xception)
Accuracy: 0.8
5. Evaluating the Models Together on Validation Data - Ensembling the models
[121] 1 from keras.layers import Input
5.1 Defining the Input Shape
[122] 1 # Single input for multiple models
2 model_input = Input(shape=(512, 512, 3))
5.2 Defining all the Models
18m 1s completed at 12:43 AM
Type here to search
01:08 AM
12-05-2021

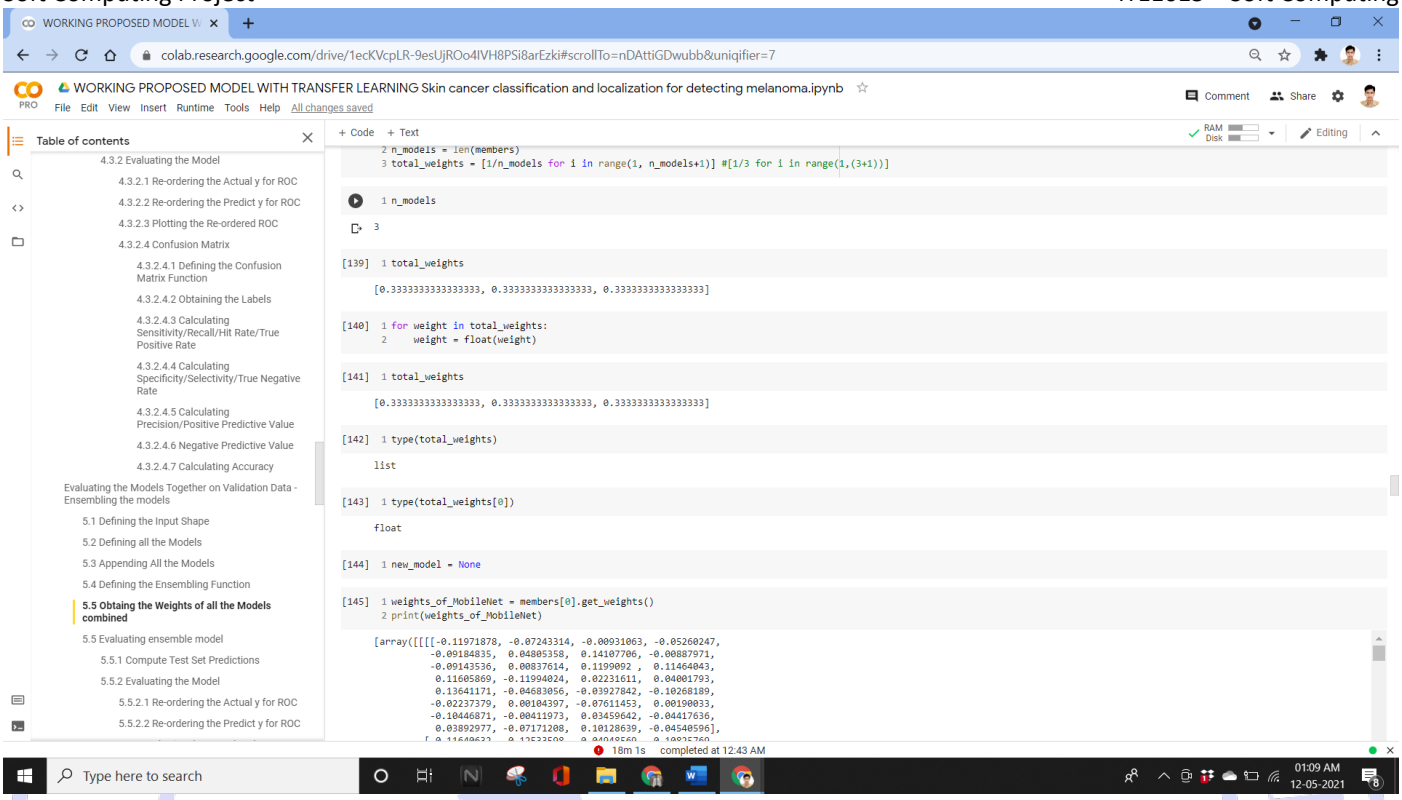

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=WdwcwcrP-0I_&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
Editing
5.2 Defining all the Models
[123] 1 def mobilenet_architecture():
2 """
3 Pre-build architecture of mobilenet for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.mobilenet import MobileNet
7
8 # Pre-build model
9 base_model = MobileNet(include_top = False, weights = None, input_tensor = model_input)
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 mobilenet_model = Model(base_model.input, output)
18
19 # Getting the summary of architecture
20 mobilenet_model.summary()
21
22 # Compiling the model
23 mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
24 loss = 'categorical_crossentropy',
25 metrics = ['accuracy'])
26
27 return mobilenet_model
[124] 1 # Model 1
2 mobilenet_model = mobilenet_architecture()
3 mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5")
Model: "model_9"
Layer (type) Output Shape Param #
input_10 (InputLayer) [(None, 512, 512, 3)] 0
conv1 (Conv2D) (None, 256, 256, 32) 864
18m 1s completed at 12:43 AM
Type here to search
01:09 AM
12-05-2021

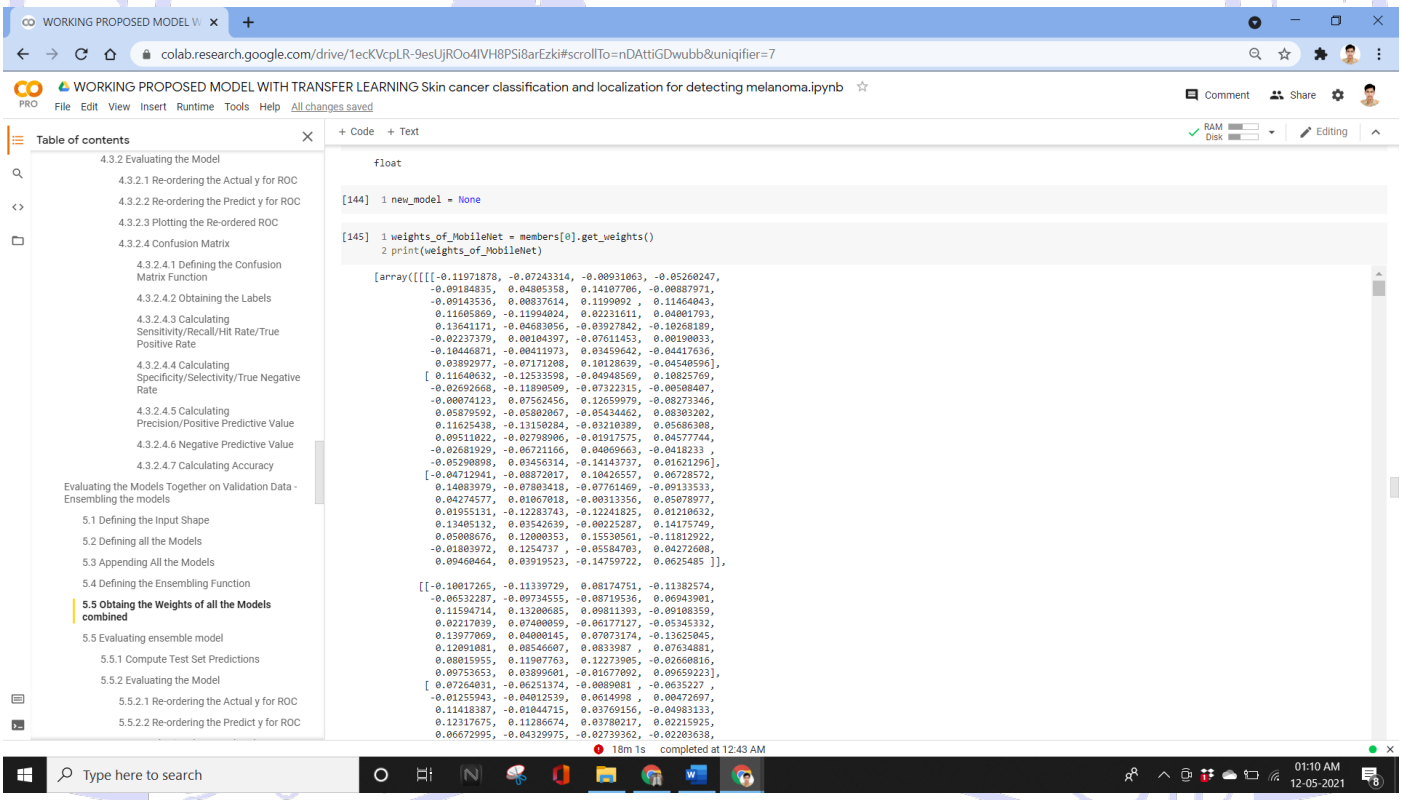

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=WdwcwcrP-0I_&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
1 # Model 2
2 inception_model = inception_architecture()
3 inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5")
Model: "model_10"
Layer (type) Output Shape Param # Connected to
input_10 (InputLayer) [(None, 512, 512, 3) 0
conv2d_294 (Conv2D) (None, 255, 255, 32) 864 input_10[0][0]
batch_normalization_294 (BatchN (None, 255, 255, 32) 96 conv2d_294[0][0]
activation_282 (Activation) (None, 255, 255, 32) 0 batch_normalization_294[0][0]
conv2d_295 (Conv2D) (None, 253, 253, 32) 9216 activation_282[0][0]
batch_normalization_295 (BatchN (None, 253, 253, 32) 96 conv2d_295[0][0]
activation_283 (Activation) (None, 253, 253, 32) 0 batch_normalization_295[0][0]
conv2d_296 (Conv2D) (None, 253, 253, 64) 18432 activation_283[0][0]
batch_normalization_296 (BatchN (None, 253, 253, 64) 192 conv2d_296[0][0]
activation_284 (Activation) (None, 253, 253, 64) 0 batch_normalization_296[0][0]
max_pooling2d_12 (MaxPooling2D) (None, 126, 126, 64) 0 activation_284[0][0]
conv2d_297 (Conv2D) (None, 126, 126, 80) 5120 max_pooling2d_12[0][0]
batch_normalization_297 (BatchN (None, 126, 126, 80) 240 conv2d_297[0][0]
activation_285 (Activation) (None, 126, 126, 80) 0 batch_normalization_297[0][0]
conv2d_298 (Conv2D) (None, 124, 124, 192 138240 activation_285[0][0]
batch_normalization_298 (BatchN (None, 124, 124, 192 576 conv2d_298[0][0]
activation_286 (Activation) (None, 124, 124, 192 0 batch_normalization_298[0][0]
max_pooling2d_13 (MaxPooling2D) (None, 61, 61, 192) 0 activation_286[0][0]
conv2d_302 (Conv2D) (None, 61, 61, 64) 12288 max_pooling2d_13[0][0]
batch_normalization_302 (BatchN (None, 61, 61, 64) 192 conv2d_302[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:09 AM
12-05-2021

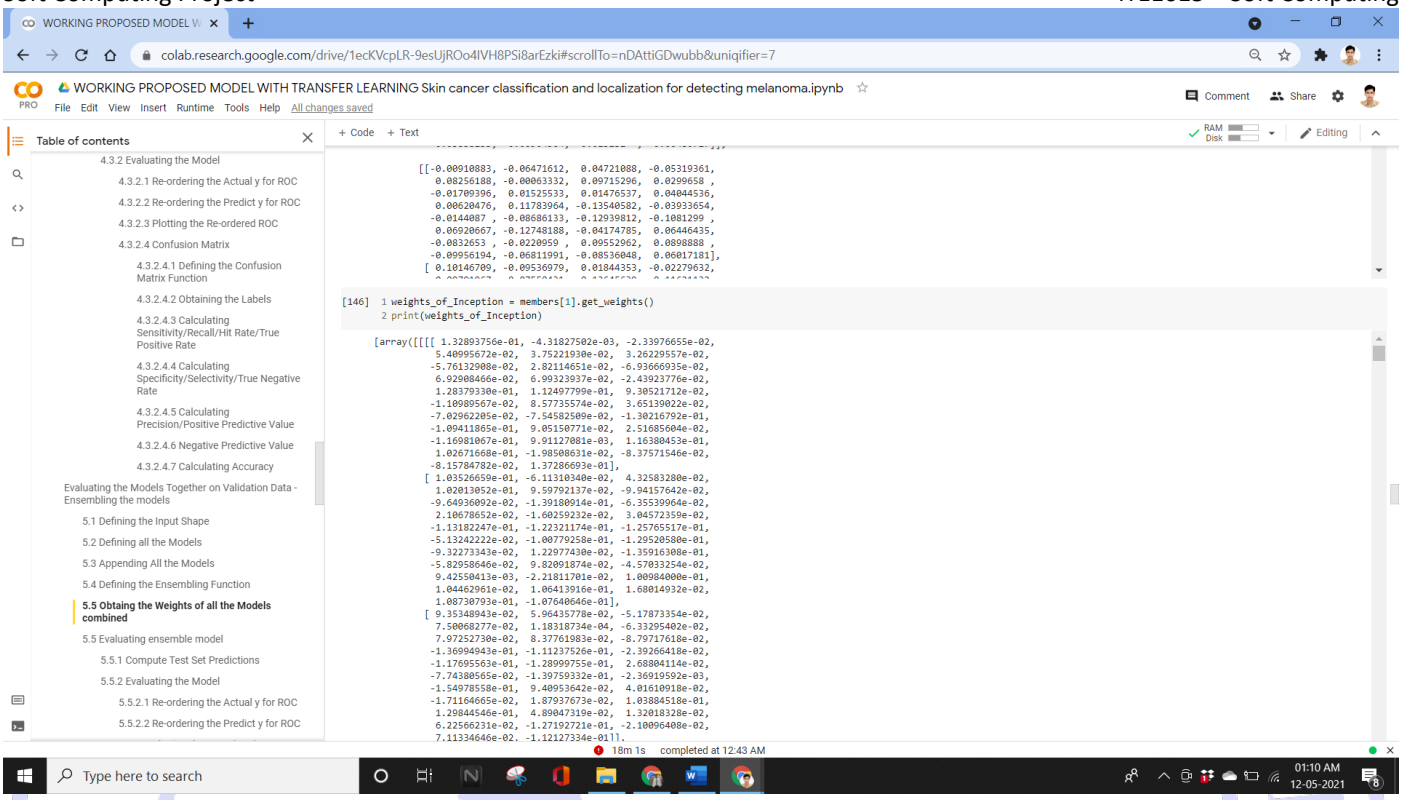

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=WdwcwcrP-0I_&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
activation_288 (Activation) (None, 61, 61, 48) 0 batch_normalization_300[0][0]
activation_291 (Activation) (None, 61, 61, 96) 0 batch_normalization_303[0][0]
average_pooling2d_27 (AveragePo (None, 61, 61, 192) 0 max_pooling2d_13[0][0]
1 def xception_architecture():
2 """
3 Pre-build architecture of inception for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.xception import Xception
7
8 # Pre-build model
9 base_model = Xception(include_top = False, weights = None, input_tensor = model_input)
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 xception_model = Model(base_model.input, output)
18
19 # Summary of the model
20 xception_model.summary()
21
22 # Compiling the model
23 xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
24 loss = 'categorical_crossentropy',
25 metrics = ['accuracy'])
26
27 return xception_model
[128] 1 # Model 3
2 xception_model = xception_architecture()
3 xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5")
Model: "model_11"
Layer (type) Output Shape Param # Connected to
input_10 (InputLayer) [(None, 512, 512, 3) 0
18m 1s completed at 12:43 AM
Type here to search
01:09 AM
12-05-2021

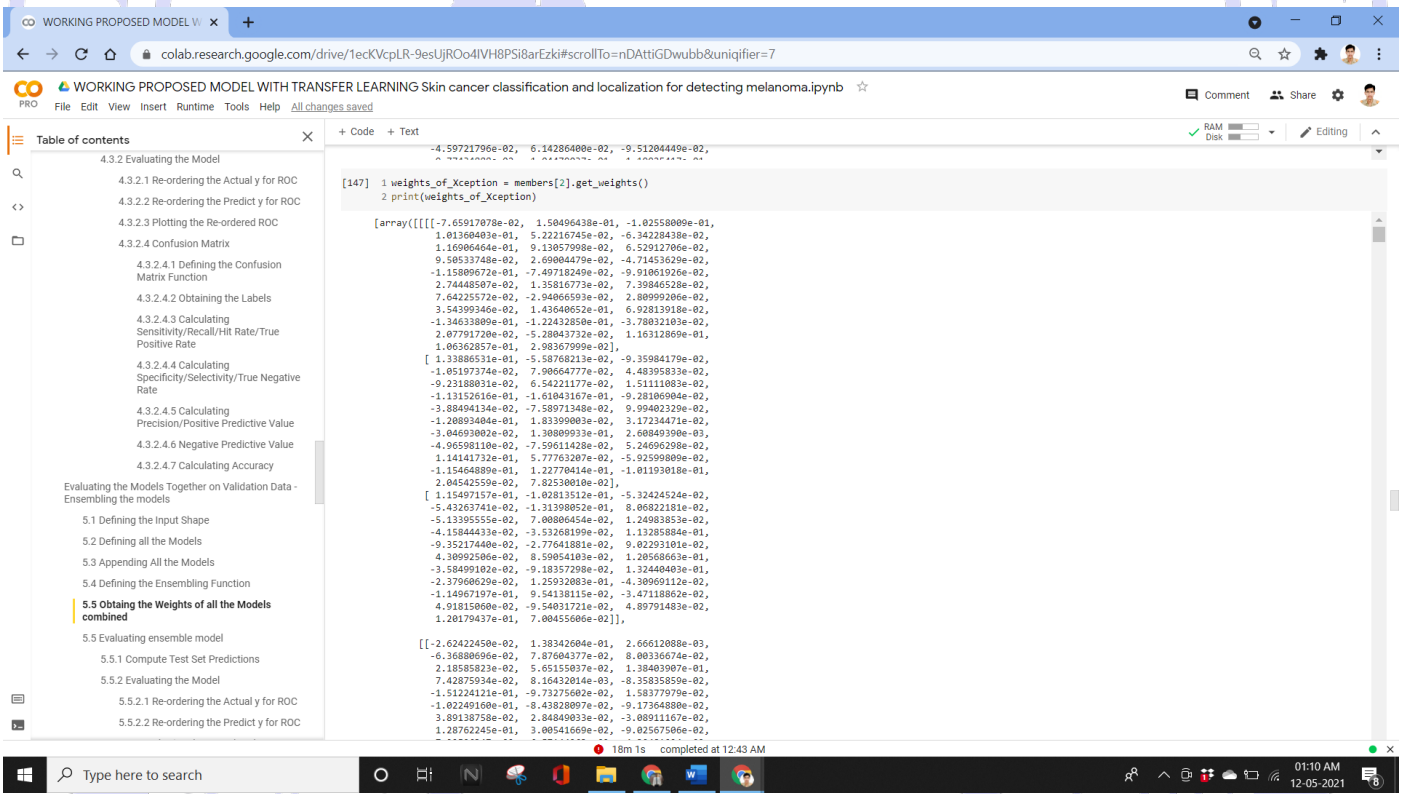

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=64NnUaxLNcHX&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
5.3 Appending All the Models
[129] 1 # Appending all models
2 models = [mobilenet_model, inception_model, xception_model]
5.4 Defining the Ensembling Function
[130] 1 def ensemble(models, model_input):
2 outputs = [model.outputs[0] for model in models]
3 print("Outputs: ")
4 print(outputs)
5 y = keras.layers.Average()(outputs)
6 print("y: ")
7 print(y)
8 model = Model(model_input, y, name='ensemble')
9 print("Model: ")
10 print(model)
11 return model
[131] 1 # Getting ensemble model
2 ensemble_model = ensemble(models, model_input)
Outputs:
[<KerasTensor: shape=(None, 2) dtype=float32 (created by layer 'dense_9')>, <KerasTensor: shape=(None, 2) dtype=float32 (created by layer 'dense_10')>, <KerasTensor: shape=(None, 2)
y:
KerasTensor(type_spec=TensorSpec(shape=(None, 2), dtype=tf.float32, name=None), name='average/truediv:0', description="created by layer 'average'")
Model:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
5.5 Obtaing the Weights of all the Models combined
[132] 1 # average the weights of multiple loaded models
2 from keras.models import load_model
3 from keras.models import clone_model
4 from numpy import average
Waiting for colab.research.google.com...
18m 1s completed at 12:43 AM
Type here to search
01:09 AM
12-05-2021

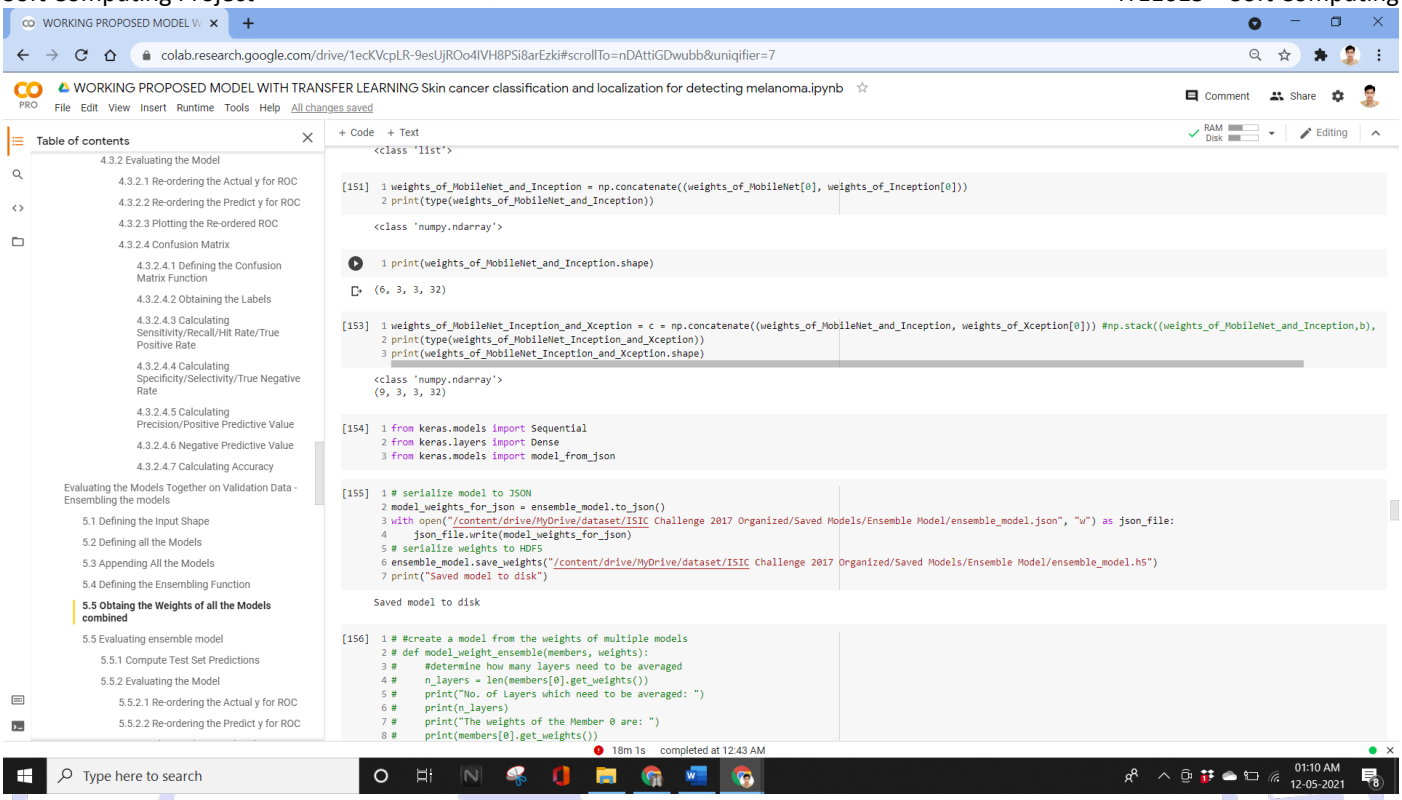

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
+ Code + Text
RAM Disk
Editing
[132] 1 # average the weights of multiple loaded models
2 from keras.models import load_model
3 from keras.models import clone_model
4 from numpy import average
5 from numpy import array
[133] 1 # load models from file
2 def load_all_models(weight_file_names_with_path):
3 all_models = list()
4 for epoch in range(0,len(weight_file_names_with_path)):
5 # define filename for this ensemble
6 filename = weight_file_names_with_path[epoch] #'model_' + str(epoch) + '.h5'
7 # load model from file
8 model = load_model(filename)
9 # add to list of members
10 all_models.append(model)
11 print('>loaded %s' % filename)
12 return all_models
[134] 1 weight_file_names_with_path = ["/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5","/content/drive/MyDrive/dataset/ISIC Chall
[135] 1 # load all models into memory
2 members = load_all_models(weight_file_names_with_path)
3 print('Loaded %d models' % len(members))
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Loaded 3 models
[136] 1 members
[<tensorflow.python.keras.engine.functional.Functional at 0x7f466f8269d0>,
<tensorflow.python.keras.engine.functional.Functional at 0x7f466f7a0fd0>,
<tensorflow.python.keras.engine.functional.Functional at 0x7f467b5eefd0>]
[137] 1 # prepare an array of equal weights
2 n_models = len(members)
3 total_weights = [1/n_models for i in range(1, n_models+1)] #[1/3 for i in range(1,(3+1))]
18m 1s completed at 12:43 AM
Type here to search
01:09 AM
12-05-2021

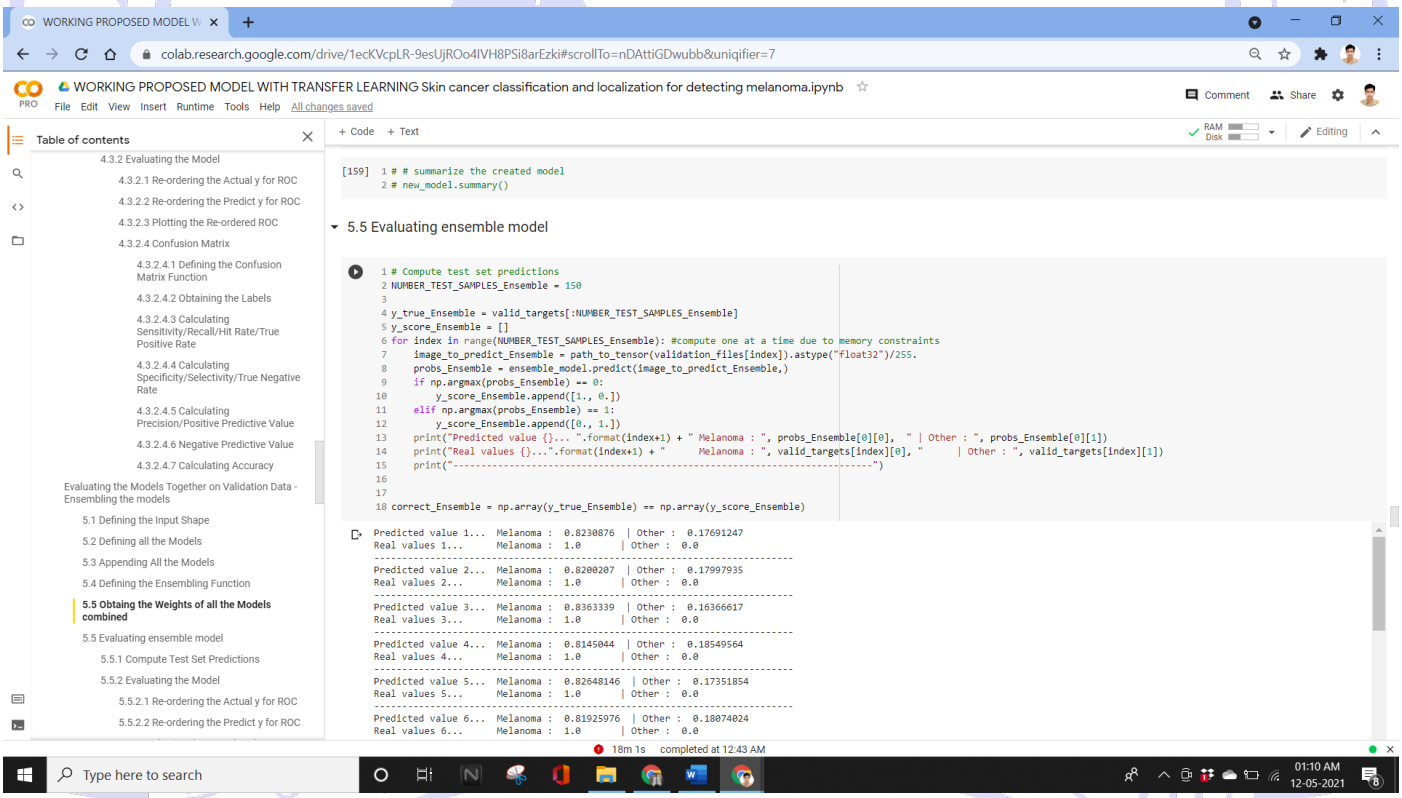

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
2 n_models = len(members)
3 total_weights = [1/n_models for i in range(1, n_models+1)] #[1/3 for i in range(1,(3+1))]
1 n_models
3
[139] 1 total_weights
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[140] 1 for weight in total_weights:
2 weight = float(weight)
[141] 1 total_weights
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[142] 1 type(total_weights)
list
[143] 1 type(total_weights[0])
float
[144] 1 new_model = None
[145] 1 weights_of_MobileNet = members[0].get_weights()
2 print(weights_of_MobileNet)
[array([[[[-0.11971878, -0.07243314, -0.00931063, -0.05260247,
-0.09184835, 0.04805358, 0.14107706, -0.00887971,
-0.09143536, 0.00837614, 0.1199092 , 0.11464043,
0.11605869, -0.11994024, 0.02231611, 0.04001793,
0.13641171, -0.04683056, -0.03927842, -0.10268189,
-0.02237379, 0.00104397, -0.07611453, 0.00190033,
-0.10446871, -0.00411973, 0.03459642, -0.04417636,
0.03892977, -0.07171208, 0.10128639, -0.04540596],
18m 1s completed at 12:43 AM
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
Type here to search
01:09 AM
12-05-2021

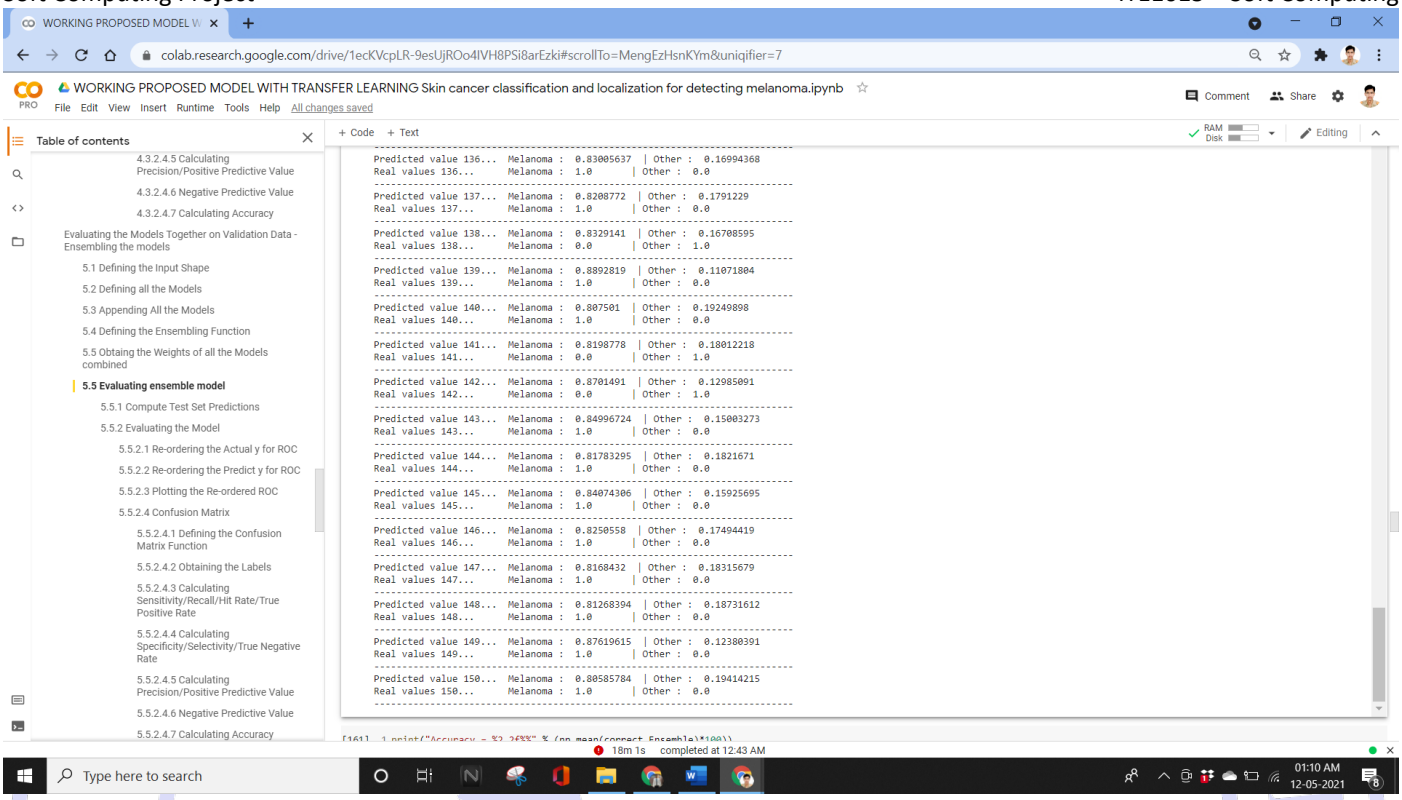

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
float
[144] 1 new_model = None
[145] 1 weights_of_MobileNet = members[0].get_weights()
2 print(weights_of_MobileNet)
[array([[[[-0.11971878, -0.07243314, -0.00931063, -0.05260247,
-0.09184835, 0.04805358, 0.14107706, -0.00887971,
-0.09143536, 0.00837614, 0.1199092 , 0.11464043,
0.11605869, -0.11994024, 0.02231611, 0.04001793,
0.13641171, -0.04683056, -0.03927842, -0.10268189,
-0.02237379, 0.00104397, -0.07611453, 0.00190033,
-0.10446871, -0.00411973, 0.03459642, -0.04417636,
0.03892977, -0.07171208, 0.10128639, -0.04540596],
[0.11640632, -0.12533598, -0.04948569, 0.10825769,
-0.02692668, -0.11890509, -0.07322315, -0.00508407,
-0.00074123, 0.07562456, 0.12659979, -0.08273346,
0.05879592, -0.05802067, -0.05434462, 0.08303202,
0.11625438, -0.13150284, -0.03210389, 0.05686308,
0.09511022, -0.02798906, -0.01917575, 0.04577744,
-0.02681929, -0.06721166, 0.04069663, -0.0418233 ,
-0.05290898, 0.03456314, -0.14143737, 0.01621296],
[-0.04712941, -0.08872017, 0.10426557, 0.06728572,
0.14083979, -0.07803418, -0.07761469, -0.09133533,
0.04274577, 0.01067018, -0.00313356, 0.05078977,
0.01955131, -0.12283743, -0.12241825, 0.01210632,
0.13405132, 0.03542639, -0.00225287, 0.14175749,
0.05008676, 0.12000353, 0.15530561, -0.11812922,
-0.01803972, 0.1254737 , -0.05584703, 0.04272608,
0.09460464, 0.03919523, -0.14759722, 0.0625485]],
[[-0.10017265, -0.11339729, 0.08174751, -0.11382574,
-0.06532287, -0.09734555, -0.08719536, 0.06943901,
0.11594714, 0.13200685, 0.09811393, -0.09108359,
0.02217039, 0.07400059, -0.06177127, -0.05345332,
0.13977069, 0.04000145, 0.07073174, -0.13625045,
0.12091081, 0.08546607, 0.0833987 , 0.07634881,
0.08015955, 0.11907763, 0.12273905, -0.02660816,
0.09753653, 0.03899601, -0.01677092, 0.09659223],
[0.07264031, -0.06251374, -0.0089081 , -0.0635227 ,
-0.01255943, -0.04012539, 0.0614998 , 0.00472697,
0.11418387, -0.01044715, 0.03769156, -0.04983133,
0.12317675, 0.11286674, 0.03780217, 0.02215925,
0.06672995, -0.04329975, -0.02739362, -0.02203638,
18m 1s completed at 12:43 AM
Table of contents
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
Type here to search
01:10 AM
12-05-2021

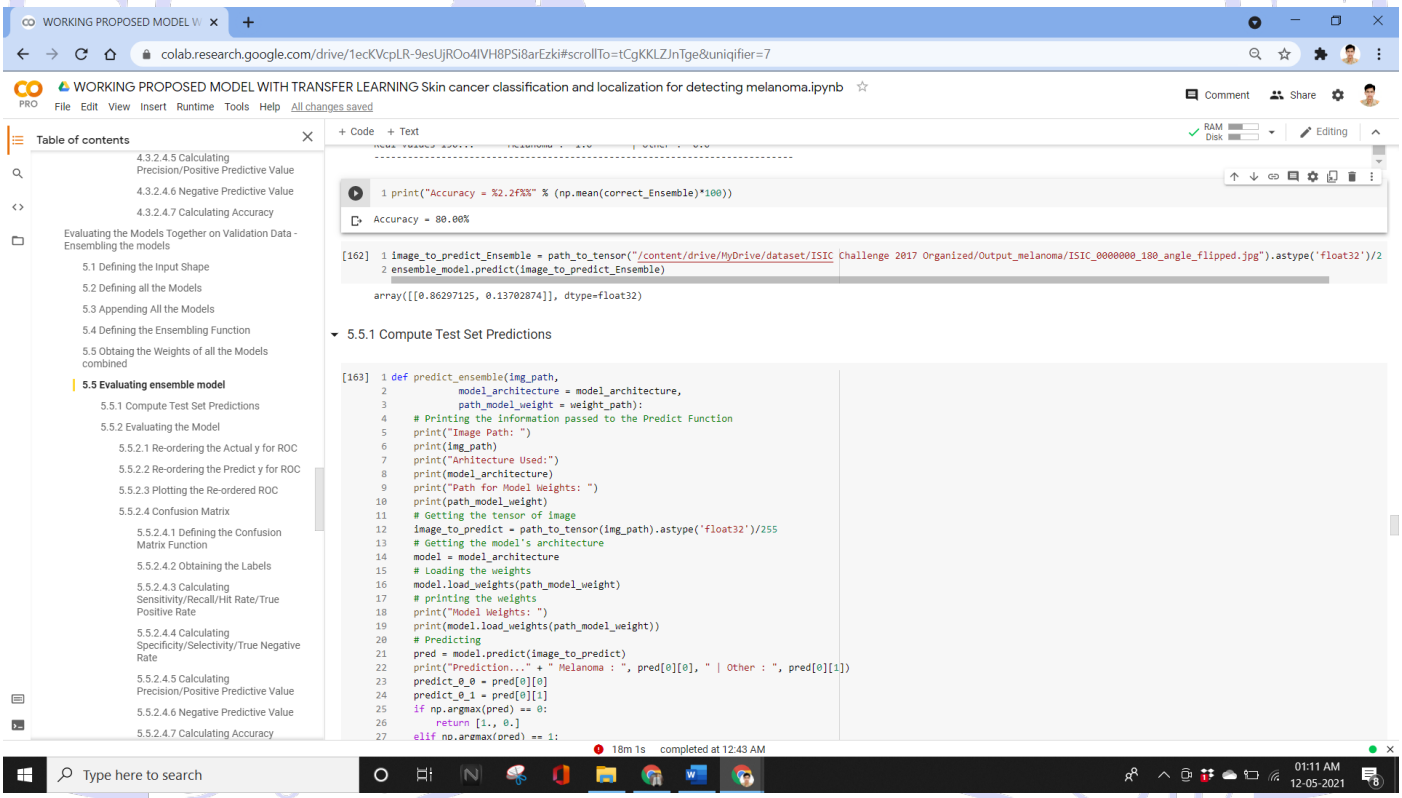

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
[146] 1 weights_of_Inception = members[1].get_weights()
2 print(weights_of_Inception)
18m 1s completed at 12:43 AM
Type here to search
01:10 AM
12-05-2021

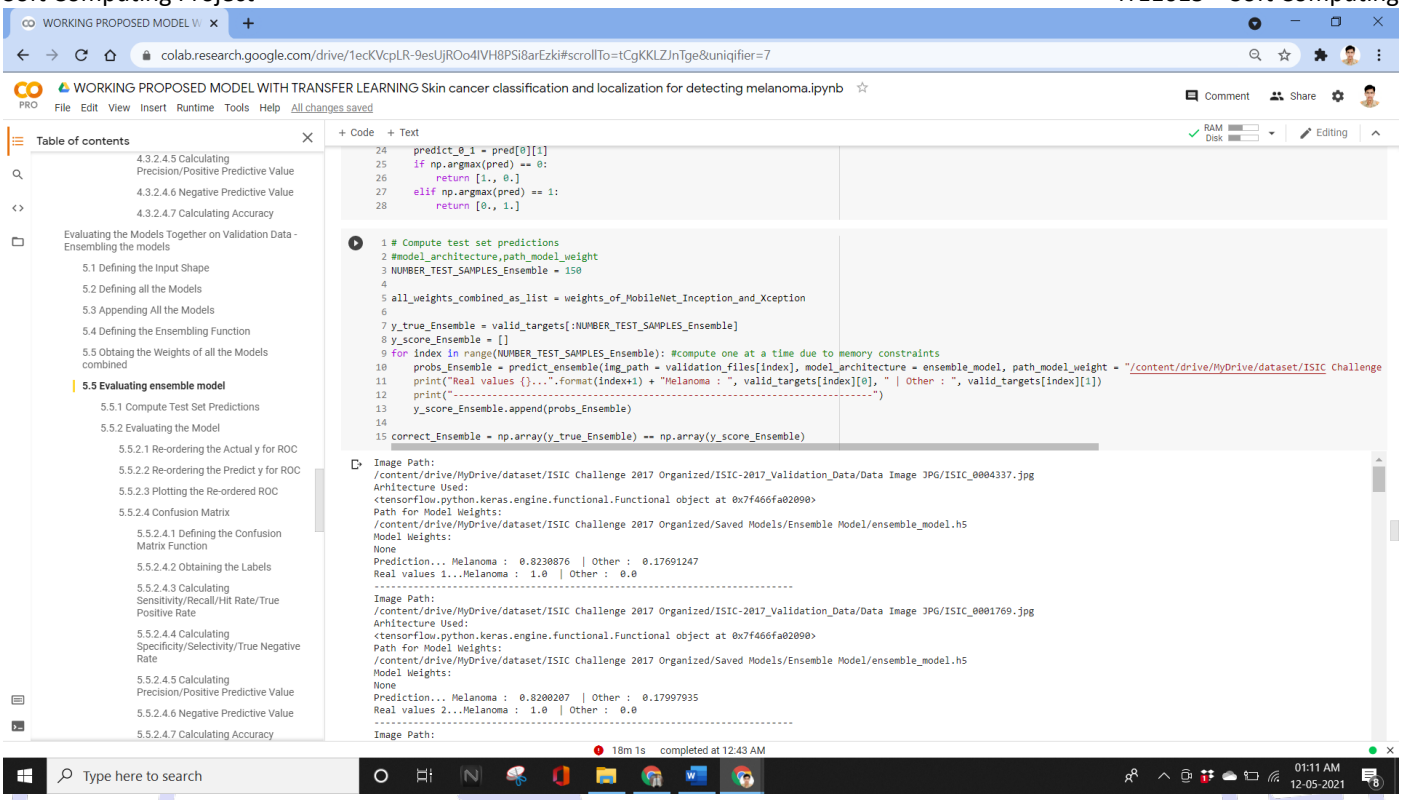

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
4.3.2 Evaluating the Model
4.3.2.1 Re-ordering the Actual y for ROC
4.3.2.2 Re-ordering the Predict y for ROC
4.3.2.3 Plotting the Re-ordered ROC
4.3.2.4 Confusion Matrix
4.3.2.4.1 Defining the Confusion Matrix Function
4.3.2.4.2 Obtaining the Labels
4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaing the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
[147] 1 weights_of_Xception = members[2].get_weights()
2 print(weights_of_Xception)
18m 1s completed at 12:43 AM
Type here to search
01:10 AM
12-05-2021

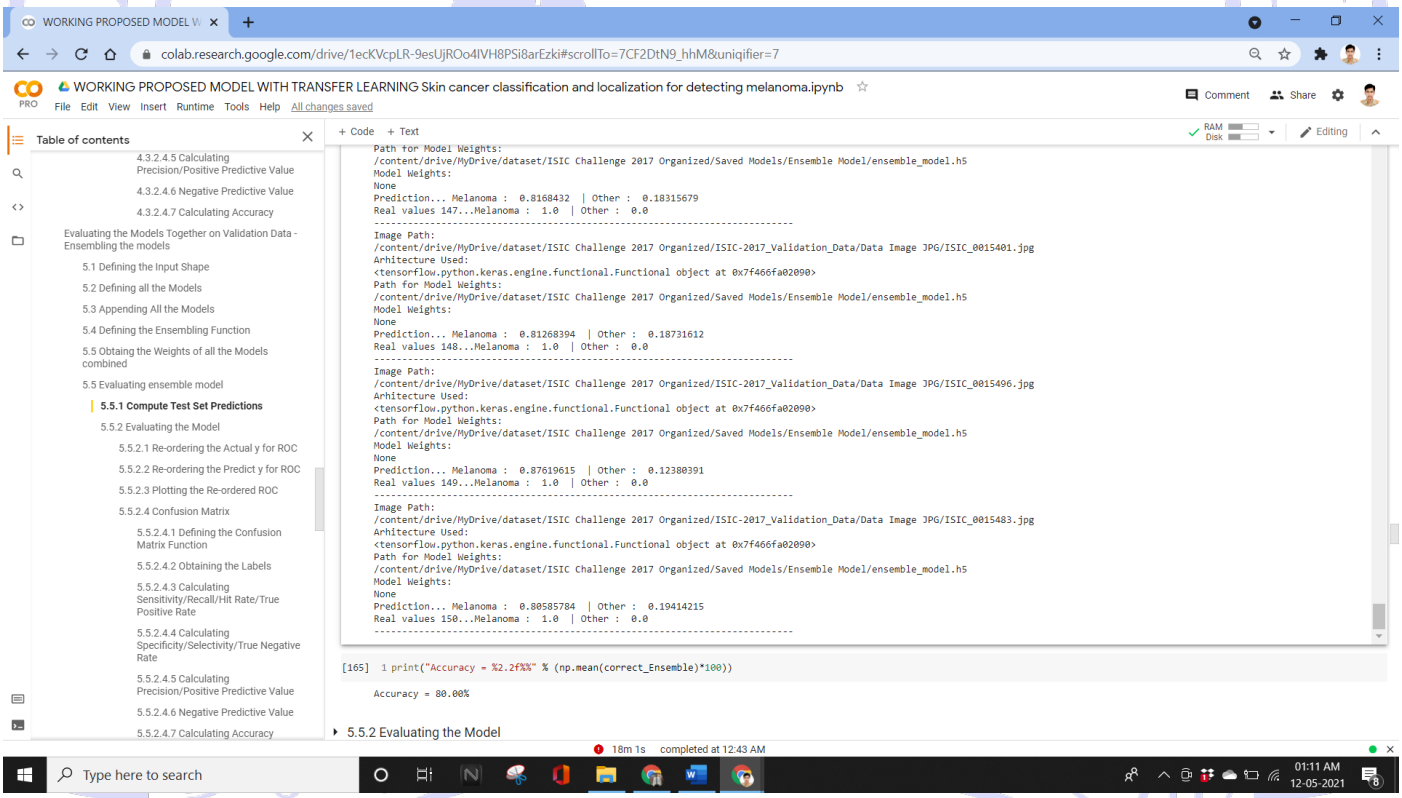

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
<class 'list'>
[151] 1 weights_of_MobileNet_and_Inception = np.concatenate((weights_of_MobileNet[0], weights_of_Inception[0]))
2 print(type(weights_of_MobileNet_and_Inception))
<class 'numpy.ndarray'>
1 print(weights_of_MobileNet_and_Inception.shape)
(6, 3, 3, 32)
[153] 1 weights_of_MobileNet_Inception_and_Xception = c = np.concatenate((weights_of_MobileNet_and_Inception, weights_of_Xception[0])) #np.stack((weights_of_MobileNet_and_Inception,b),
2 print(type(weights_of_MobileNet_Inception_and_Xception))
3 print(weights_of_MobileNet_Inception_and_Xception.shape)
<class 'numpy.ndarray'>
(9, 3, 3, 32)
[154] 1 from keras.models import Sequential
2 from keras.layers import Dense
3 from keras.models import model_from_json
[155] 1 # serialize model to JSON
2 model_weights_for_json = ensemble_model.to_json()
3 with open("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.json", "w") as json_file:
4 json_file.write(model_weights_for_json)
5 # serialize weights to HDF5
6 ensemble_model.save_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5")
7 print("Saved model to disk")
Saved model to disk
[156] 1 # #create a model from the weights of multiple models
2 # def model_weight_ensemble(members, weights):
3 # #determine how many layers need to be averaged
4 # n_layers = len(members[0].get_weights())
5 # print("No. of Layers which need to be averaged: ")
6 # print(n_layers)
7 # print("The weights of the Member 0 are: ")
8 # print(members[0].get_weights())
18m 1s completed at 12:43 AM
Type here to search
01:10 AM
12-05-2021

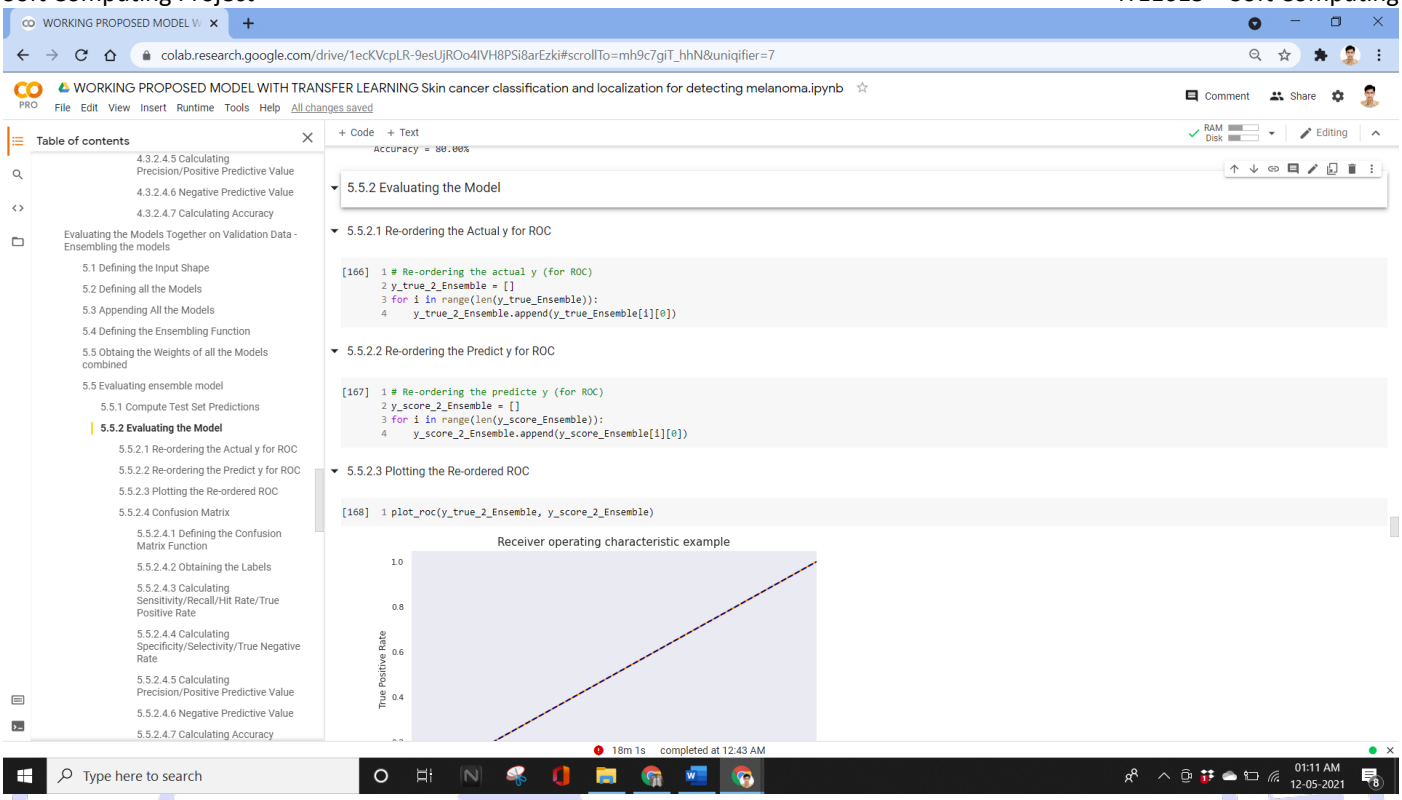

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=nDAttiGDwubb&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
[159] 1 # # summarize the created model
2 # new_model.summary()
5.5 Evaluating ensemble model
1 # Compute test set predictions
2 NUMBER_TEST_SAMPLES_Ensemble = 150
3
4 y_true_Ensemble = valid_targets[:NUMBER_TEST_SAMPLES_Ensemble]
5 y_score_Ensemble = []
6 for index in range(NUMBER_TEST_SAMPLES_Ensemble): #compute one at a time due to memory constraints
7 image_to_predict_Ensemble = path_to_tensor(validation_files[index]).astype("float32")/255.
8 probs_Ensemble = ensemble_model.predict(image_to_predict_Ensemble,)
9 if np.argmax(probs_Ensemble) == 0:
10 y_score_Ensemble.append([1., 0.])
11 elif np.argmax(probs_Ensemble) == 1:
12 y_score_Ensemble.append([0., 1.])
13 print("Predicted value {}... ".format(index+1) + " Melanoma : ", probs_Ensemble[0][0], " | Other : ", probs_Ensemble[0][1])
14 print("Real values {}...".format(index+1) + " Melanoma : ", valid_targets[index][0], " | Other : ", valid_targets[index][1])
15 print("---")
16
17
18 correct_Ensemble = np.array(y_true_Ensemble) == np.array(y_score_Ensemble)
Predicted value 1... Melanoma : 0.8230876 | Other : 0.17691247
Real values 1... Melanoma : 1.0 | Other : 0.0
Predicted value 2... Melanoma : 0.8200207 | Other : 0.17997935
Real values 2... Melanoma : 1.0 | Other : 0.0
Predicted value 3... Melanoma : 0.8363339 | Other : 0.16366617
Real values 3... Melanoma : 1.0 | Other : 0.0
Predicted value 4... Melanoma : 0.8145044 | Other : 0.18549564
Real values 4... Melanoma : 1.0 | Other : 0.0
Predicted value 5... Melanoma : 0.82648146 | Other : 0.17351854
Real values 5... Melanoma : 1.0 | Other : 0.0
Predicted value 6... Melanoma : 0.81925976 | Other : 0.18074024
Real values 6... Melanoma : 1.0 | Other : 0.0
18m 1s completed at 12:43 AM
Type here to search
01:10 AM
12-05-2021

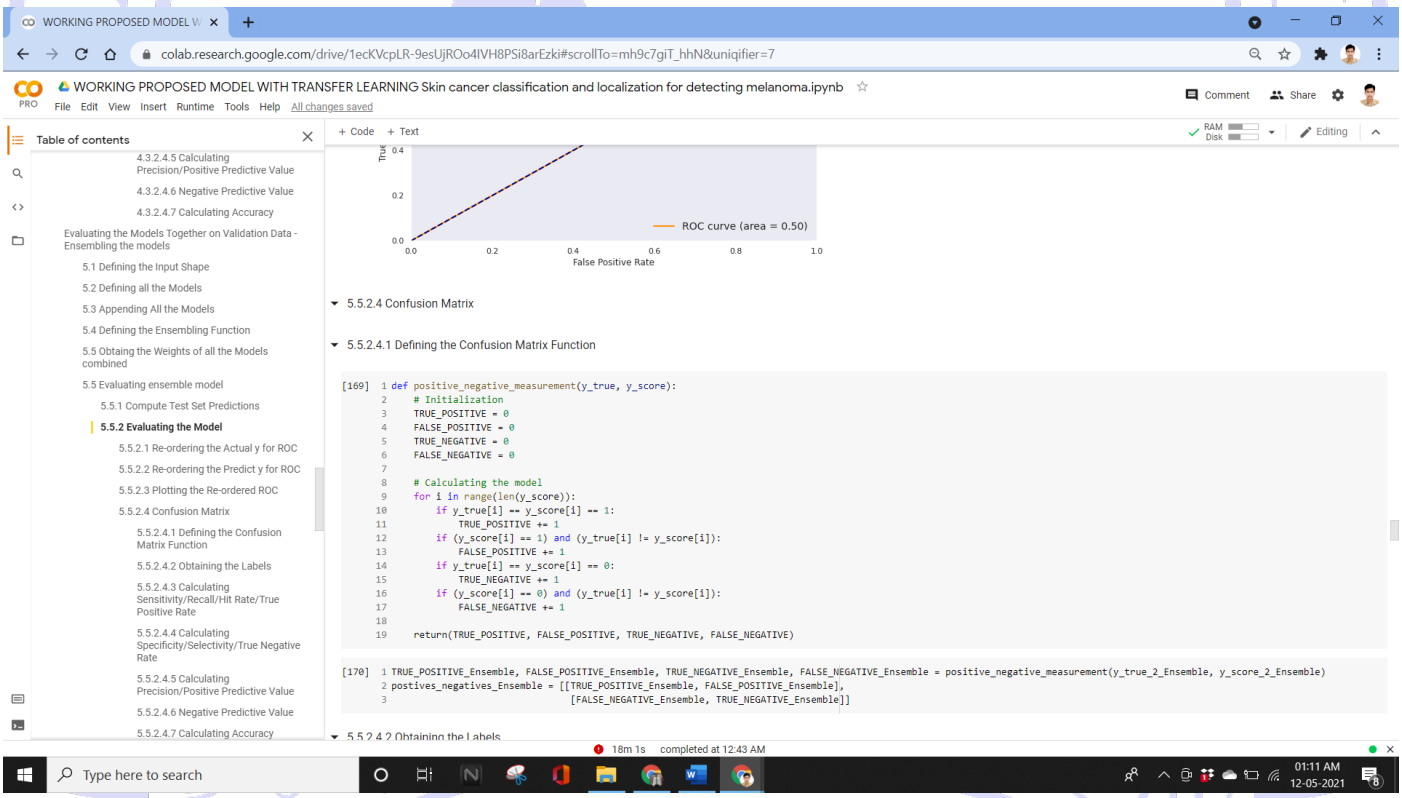

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=MengEzHsnKYm&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Predicted value 136... Melanoma : 0.83005637 | Other : 0.16994368
Real values 136... Melanoma : 1.0 | Other : 0.0
Predicted value 137... Melanoma : 0.8208772 | Other : 0.1791229
Real values 137... Melanoma : 1.0 | Other : 0.0
Predicted value 138... Melanoma : 0.8329141 | Other : 0.16708595
Real values 138... Melanoma : 0.0 | Other : 1.0
Predicted value 139... Melanoma : 0.8892819 | Other : 0.11071804
Real values 139... Melanoma : 1.0 | Other : 0.0
Predicted value 140... Melanoma : 0.807501 | Other : 0.19249898
Real values 140... Melanoma : 1.0 | Other : 0.0
Predicted value 141... Melanoma : 0.8198778 | Other : 0.18012218
Real values 141... Melanoma : 0.0 | Other : 1.0
Predicted value 142... Melanoma : 0.8701491 | Other : 0.12985091
Real values 142... Melanoma : 0.0 | Other : 1.0
Predicted value 143... Melanoma : 0.84996724 | Other : 0.15003273
Real values 143... Melanoma : 1.0 | Other : 0.0
Predicted value 144... Melanoma : 0.81783295 | Other : 0.1821671
Real values 144... Melanoma : 1.0 | Other : 0.0
Predicted value 145... Melanoma : 0.84074306 | Other : 0.15925695
Real values 145... Melanoma : 1.0 | Other : 0.0
Predicted value 146... Melanoma : 0.8250558 | Other : 0.17494419
Real values 146... Melanoma : 1.0 | Other : 0.0
Predicted value 147... Melanoma : 0.8168432 | Other : 0.18315679
Real values 147... Melanoma : 1.0 | Other : 0.0
Predicted value 148... Melanoma : 0.81268394 | Other : 0.18731612
Real values 148... Melanoma : 1.0 | Other : 0.0
Predicted value 149... Melanoma : 0.8761961 | Other : 0.12380391
Real values 149... Melanoma : 1.0 | Other : 0.0
Predicted value 150... Melanoma : 0.80585784 | Other : 0.19414215
Real values 150... Melanoma : 1.0 | Other : 0.0
18m 1s completed at 12:43 AM
Type here to search
01:10 AM
12-05-2021

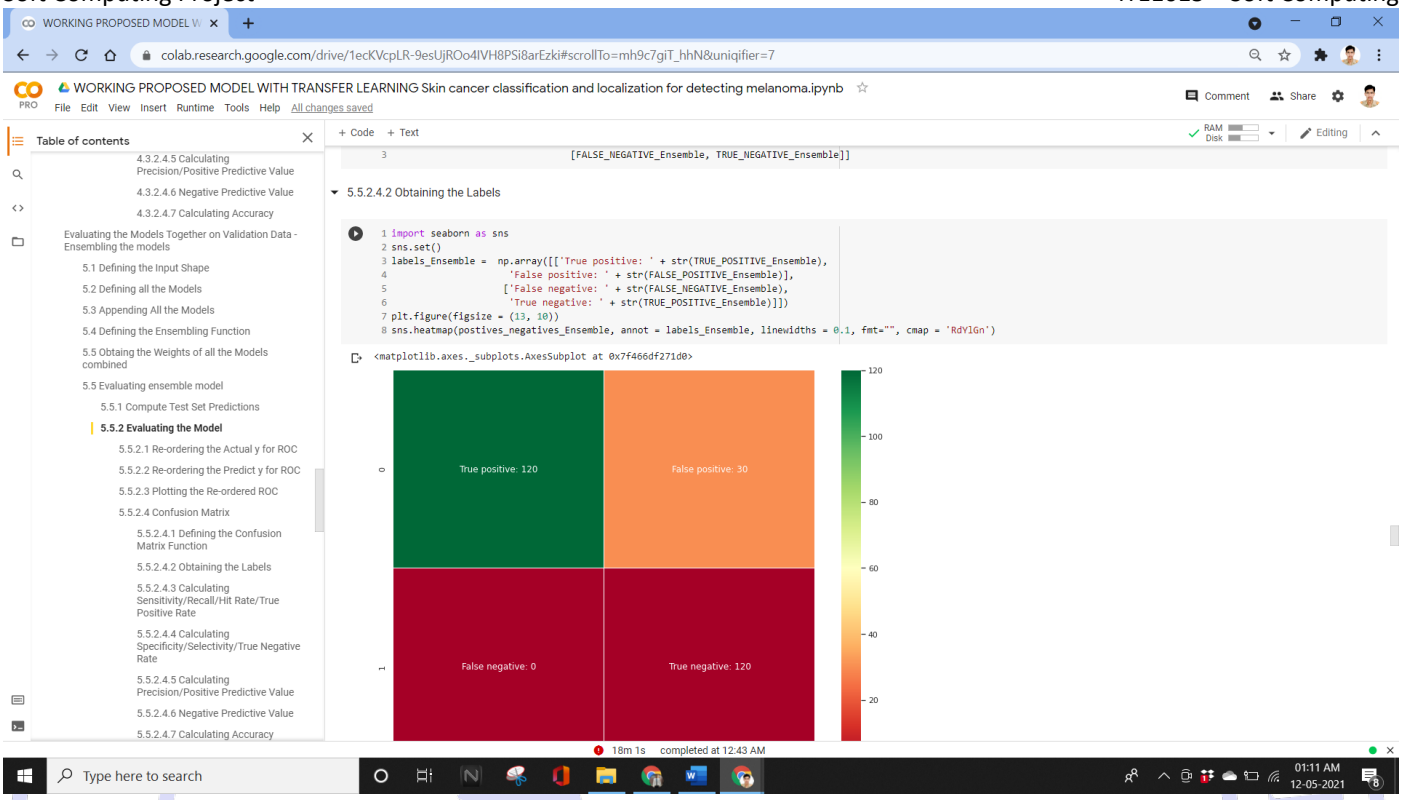

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=tCgKKLZJnTge&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
1 print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble)*100))
Accuracy = 80.00%
[162] 1 image_to_predict_Ensemble = path_to_tensor("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Output_melanoma/ISIC_0000000_180_angle_flipped.jpg").astype('float32')/2
2 ensemble_model.predict(image_to_predict_Ensemble)
array([[0.86297125, 0.13702874]], dtype=float32)
5.5.1 Compute Test Set Predictions
[163] 1 def predict_ensemble(img_path,
2 model_architecture = model_architecture,
3 path_model_weight = weight_path):
4 # Printing the information passed to the Predict Function
5 print("Image Path: ")
6 print(img_path)
7 print("Architecture Used:")
8 print(model_architecture)
9 print("Path for Model Weights: ")
10 print(path_model_weight)
11 # Getting the tensor of image
12 image_to_predict = path_to_tensor(img_path).astype('float32')/255
13 # Getting the model's architecture
14 model = model_architecture
15 # Loading the weights
16 model.load_weights(path_model_weight)
17 # printing the weights
18 print("Model Weights: ")
19 print(model.load_weights(path_model_weight))
20 # Predicting
21 pred = model.predict(image_to_predict)
22 print("Prediction..." + " Melanoma : ", pred[0][0], " | Other : ", pred[0][1])
23 predict_0_0 = pred[0][0]
24 predict_0_1 = pred[0][1]
25 if np.argmax(pred) == 0:
26 return [1., 0.]
27 elif np.argmax(pred) == 1:
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

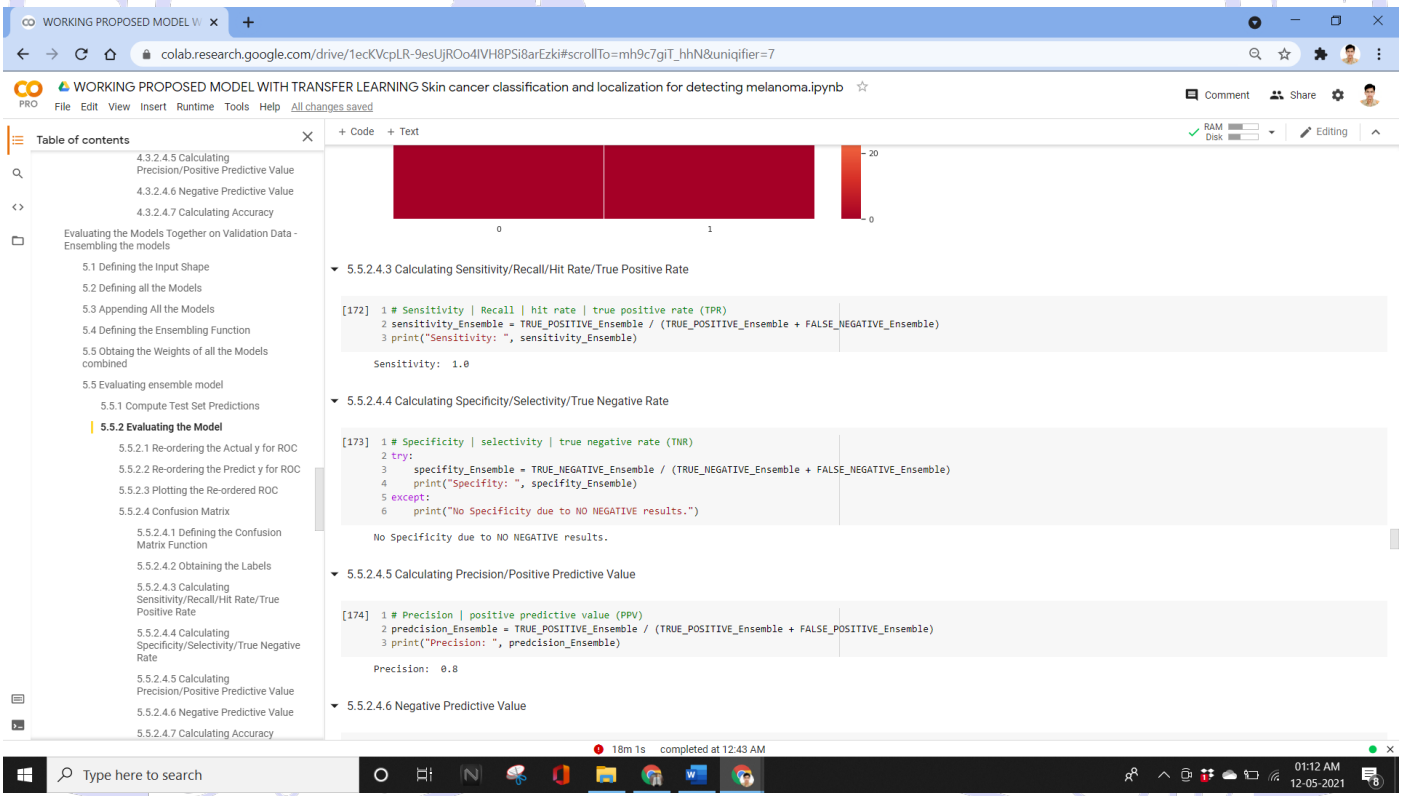

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=tCgKKLZJnTge&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
predict_0_1 = pred[0][1]
if np.argmax(pred) == 0:
return [1., 0.]
elif np.argmax(pred) == 1:
return [0., 1.]
Compute test set predictions
#model_architecture,path_model_weight
NUMBER_TEST_SAMPLES_Ensemble = 150
all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception
y_true_Ensemble = valid_targets[:NUMBER_TEST_SAMPLES_Ensemble]
y_score_Ensemble = []
for index in range(NUMBER_TEST_SAMPLES_Ensemble): #compute one at a time due to memory constraints
probs_Ensemble = predict_ensemble(img_path = validation_files[index], model_architecture = ensemble_model, path_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge
print("Real values {}...".format(index+1) + "Melanoma : ", valid_targets[index][0], " | Other : ", valid_targets[index][1])
print("----------")
y_score_Ensemble.append(probs_Ensemble)
correct_Ensemble = np.array(y_true_Ensemble) == np.array(y_score_Ensemble)
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0004337.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8230876 | Other : 0.17691247
Real values 1...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8200207 | Other : 0.17997935
Real values 2...Melanoma : 1.0 | Other : 0.0
Image Path:
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

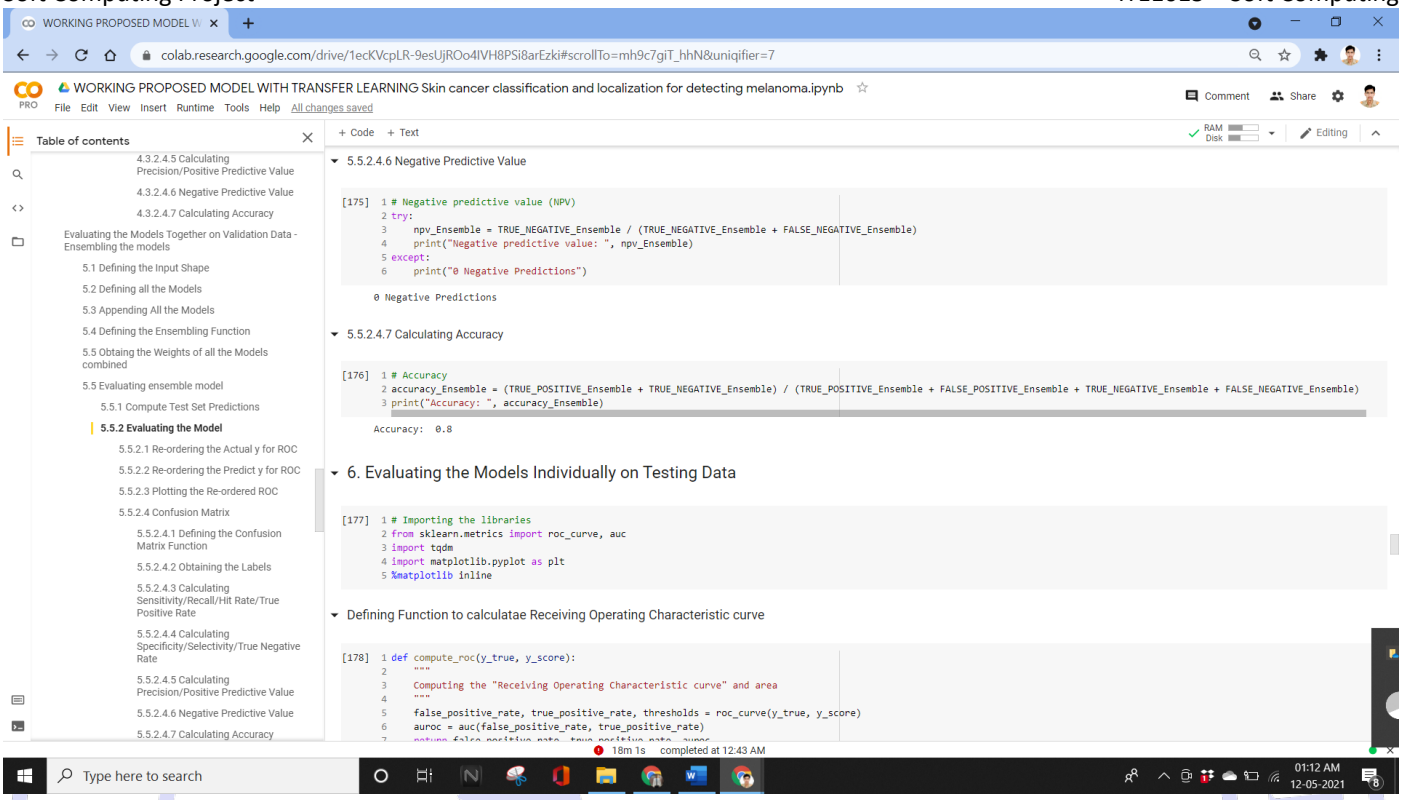

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=7CF2DtN9_hhM&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8168432 | Other : 0.18315679
Real values 147...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015401.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.81268394 | Other : 0.18731612
Real values 148...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015496.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.87619615 | Other : 0.12380391
Real values 149...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0015483.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466fa02090>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.80585784 | Other : 0.19414215
Real values 150...Melanoma : 1.0 | Other : 0.0
print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble)*100))
Accuracy = 80.00%
5.5.2 Evaluating the Model
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

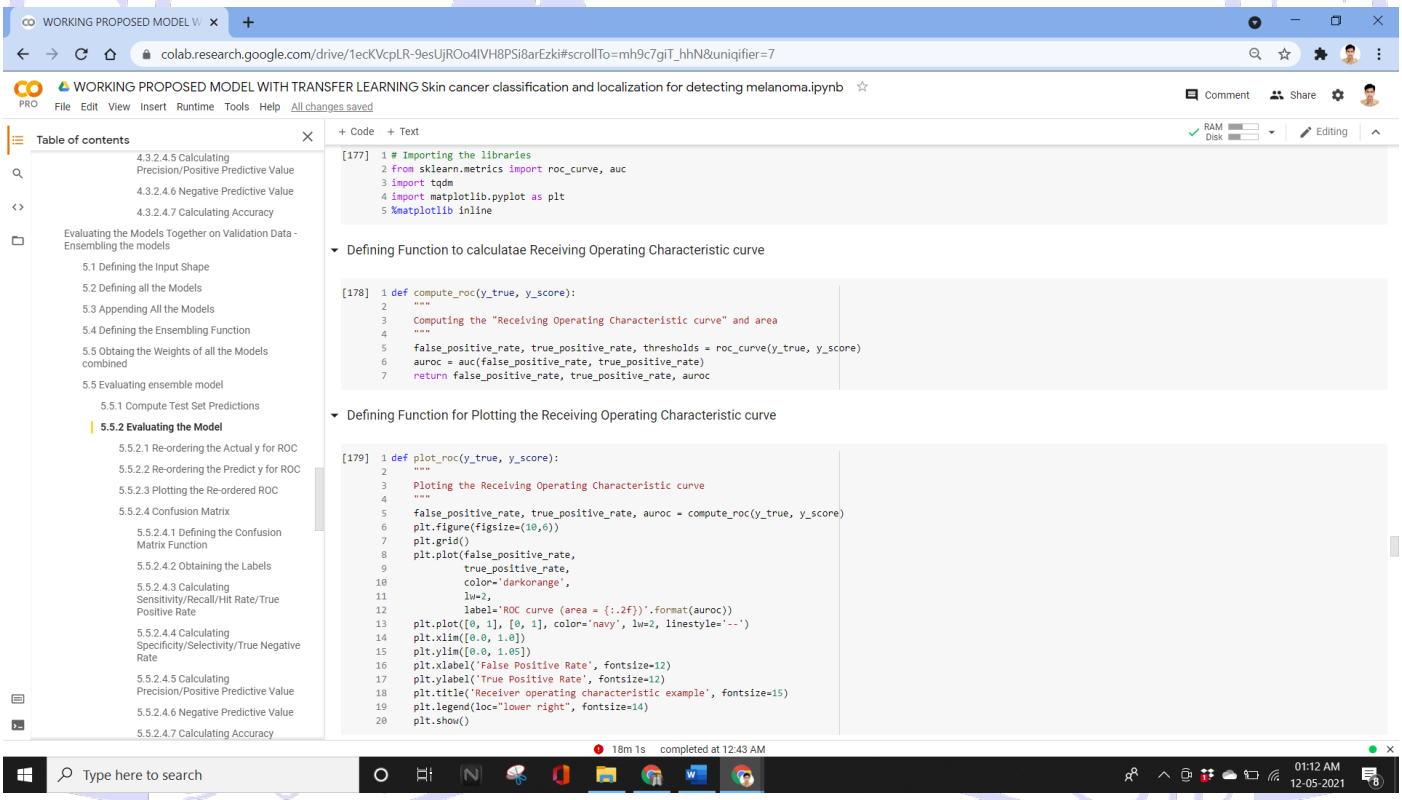

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaining the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
5.5.2.3 Plotting the Re-ordered ROC
5.5.2.4 Confusion Matrix
5.5.2.4.1 Defining the Confusion Matrix Function
5.5.2.4.2 Obtaining the Labels
5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
5.5.2.4.5 Calculating Precision/Positive Predictive Value
5.5.2.4.6 Negative Predictive Value
5.5.2.4.7 Calculating Accuracy
+ Code + Text
Accuracy = 80.00%
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
[166] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Ensemble = []
3 for i in range(len(y_true_Ensemble)):
4 y_true_2_Ensemble.append(y_true_Ensemble[i][0])
5.5.2.2 Re-ordering the Predict y for ROC
[167] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Ensemble = []
3 for i in range(len(y_score_Ensemble)):
4 y_score_2_Ensemble.append(y_score_Ensemble[i][0])
5.5.2.3 Plotting the Re-ordered ROC
[168] 1 plot_roc(y_true_2_Ensemble, y_score_2_Ensemble)
Receiver operating characteristic example
True Positive Rate
1.0
0.8
0.6
0.4
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

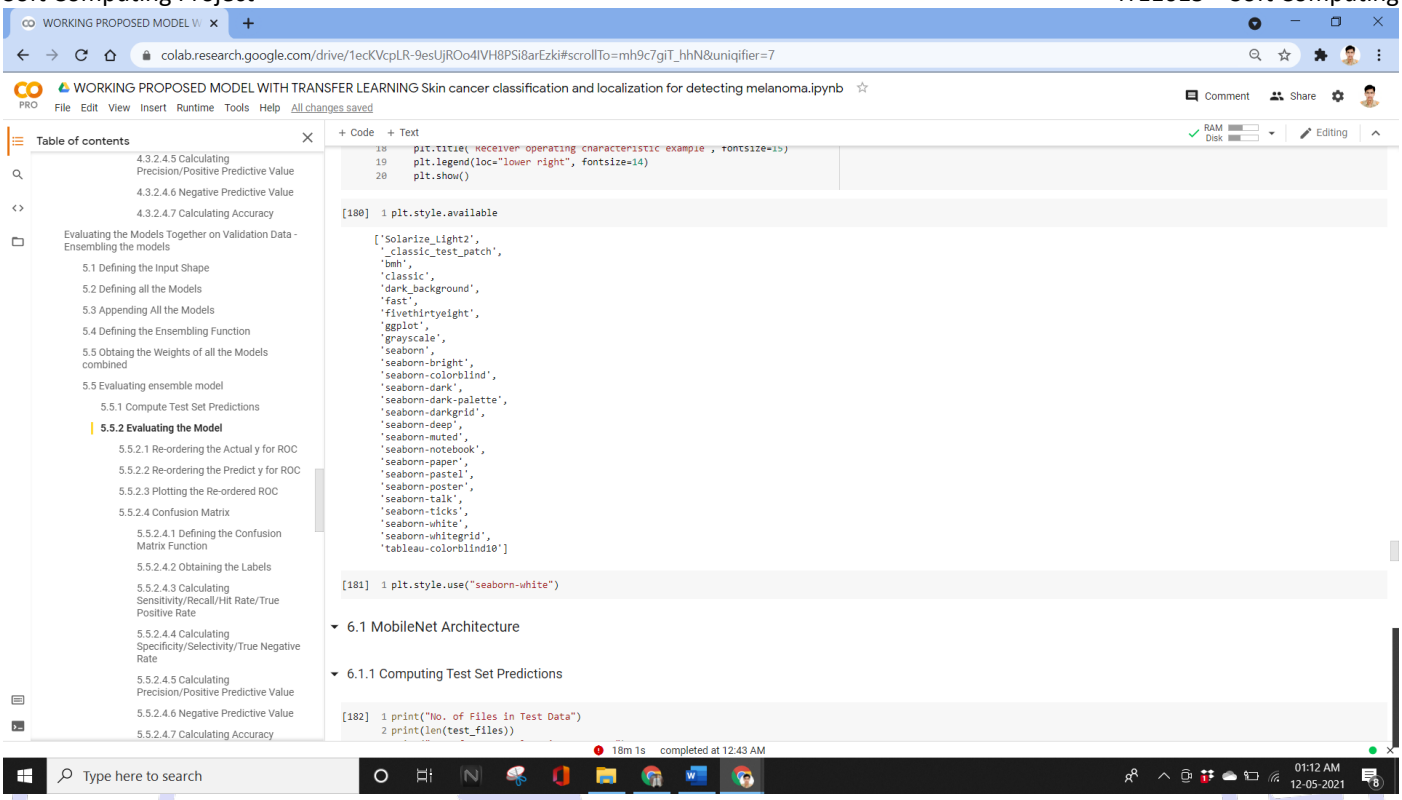

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaining the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
5.5.2.3 Plotting the Re-ordered ROC
5.5.2.4 Confusion Matrix
5.5.2.4.1 Defining the Confusion Matrix Function
5.5.2.4.2 Obtaining the Labels
5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
5.5.2.4.5 Calculating Precision/Positive Predictive Value
5.5.2.4.6 Negative Predictive Value
5.5.2.4.7 Calculating Accuracy
+ Code + Text
ROC curve (area = 0.50)
False Positive Rate
5.5.2.4 Confusion Matrix
5.5.2.4.1 Defining the Confusion Matrix Function
[169] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[170] 1 TRUE_POSITIVE_Ensemble, FALSE_POSITIVE_Ensemble, TRUE_NEGATIVE_Ensemble, FALSE_NEGATIVE_Ensemble = positive_negative_measurement(y_true_2_Ensemble, y_score_2_Ensemble)
2 postives_negatives_Ensemble = [[TRUE_POSITIVE_Ensemble, FALSE_POSITIVE_Ensemble],
3 [FALSE_NEGATIVE_Ensemble, TRUE_NEGATIVE_Ensemble]]
5.5.2.4.2 Obtaining the Labels
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

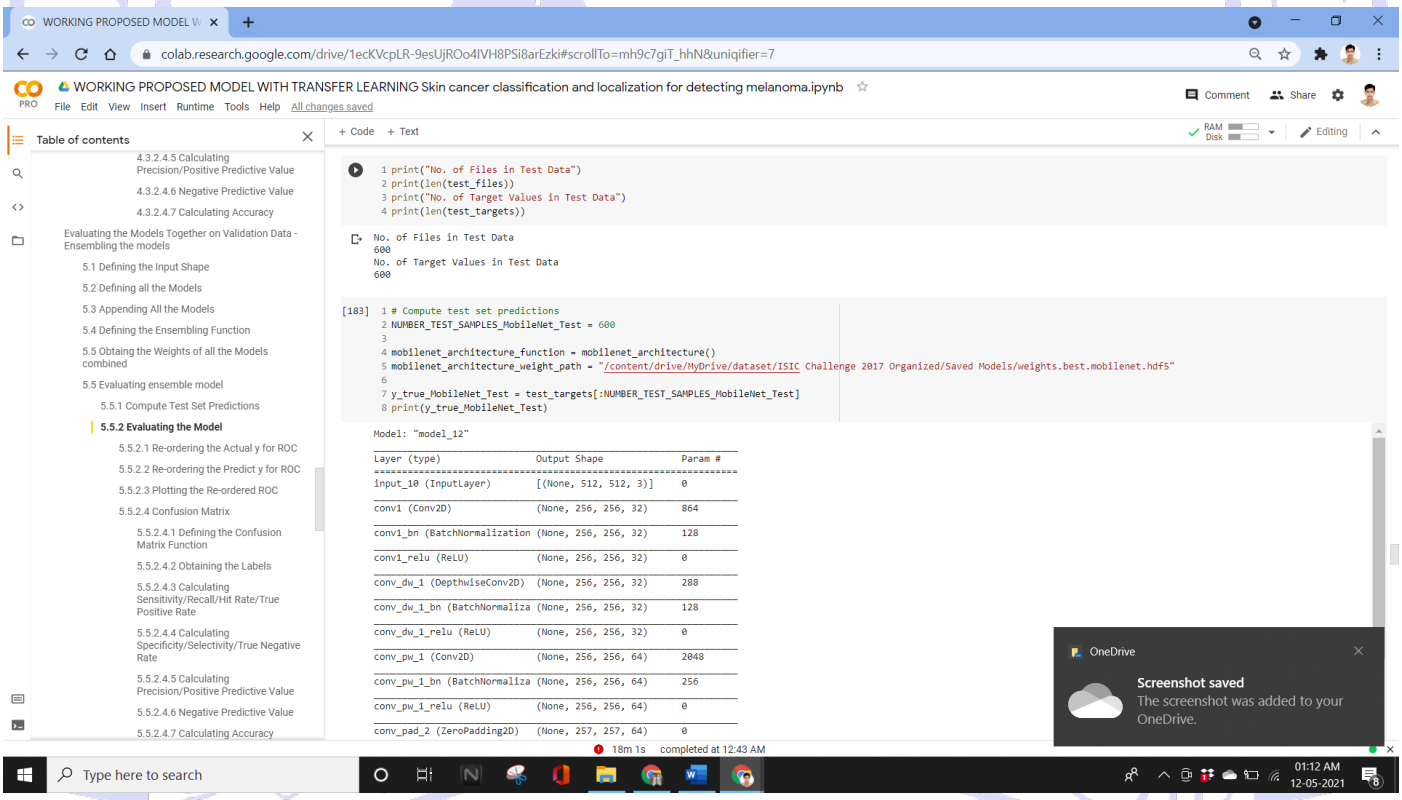

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
[FALSE_NEGATIVE_Ensemble, TRUE_NEGATIVE_Ensemble]]
5.5.2.4.2 Obtaining the Labels
1 import seaborn as sns
2 sns.set()
3 labels_Ensemble = np.array([['True positive: ' + str(TRUE_POSITIVE_Ensemble),
4 'False positive: ' + str(FALSE_POSITIVE_Ensemble)],
5 ['False negative: ' + str(FALSE_NEGATIVE_Ensemble),
6 'True negative: ' + str(TRUE_POSITIVE_Ensemble)]])
7 plt.figure(figsize = (13, 10))
8 sns.heatmap(postives_negatives_Ensemble, annot = labels_Ensemble, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f466df271d0>
True positive: 120
False positive: 30
False negative: 0
True negative: 120
18m 1s completed at 12:43 AM
Type here to search
01:11 AM
12-05-2021

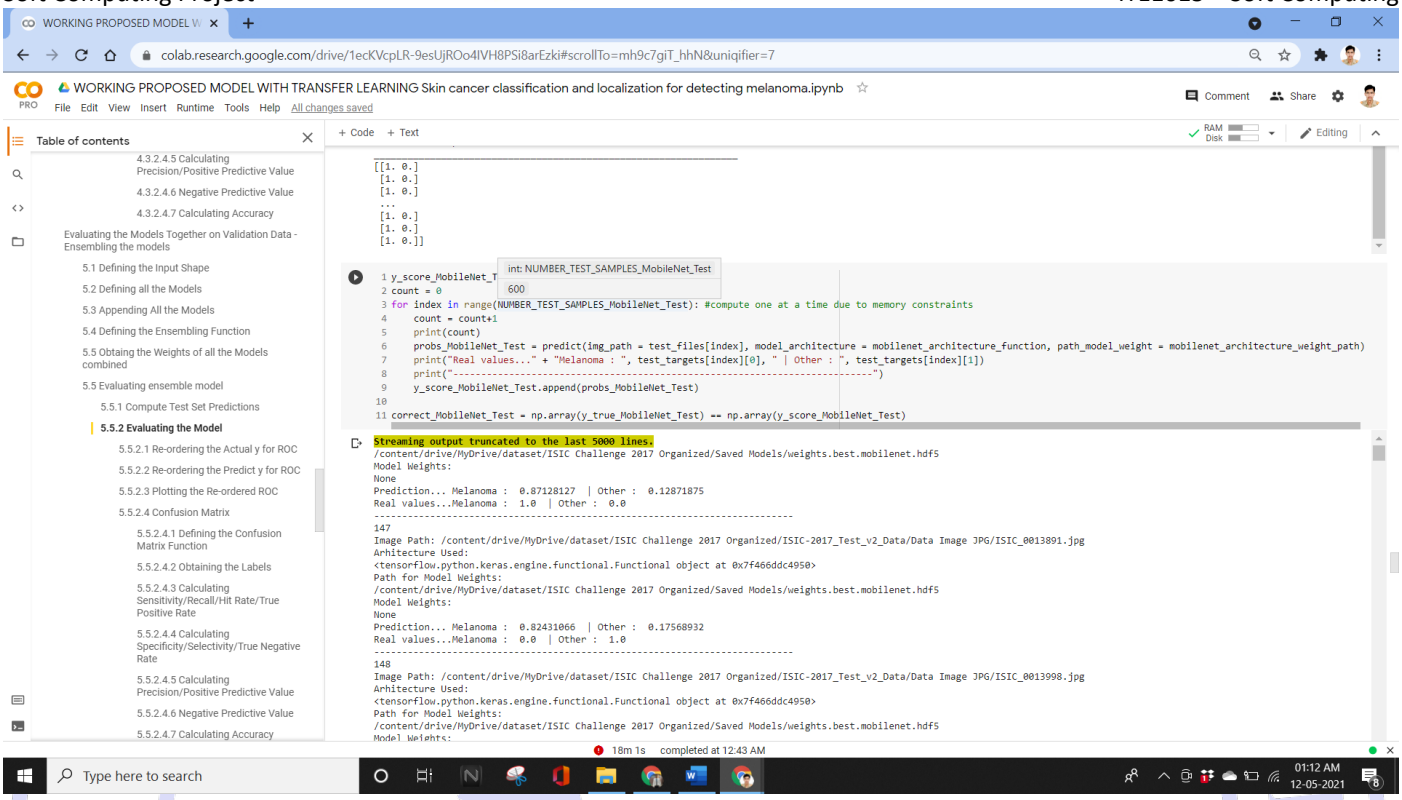

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[172] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
3 print("Sensitivity: ", sensitivity_Ensemble)
Sensitivity: 1.0
5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[173] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
4 print("Specifity: ", specifity_Ensemble)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
5.5.2.4.5 Calculating Precision/Positive Predictive Value
[174] 1 # Precision | positive predictive value (PPV)
2 predcision_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble + FALSE_POSITIVE_Ensemble)
3 print("Precision: ", predcision_Ensemble)
Precision: 0.8
5.5.2.4.6 Negative Predictive Value
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

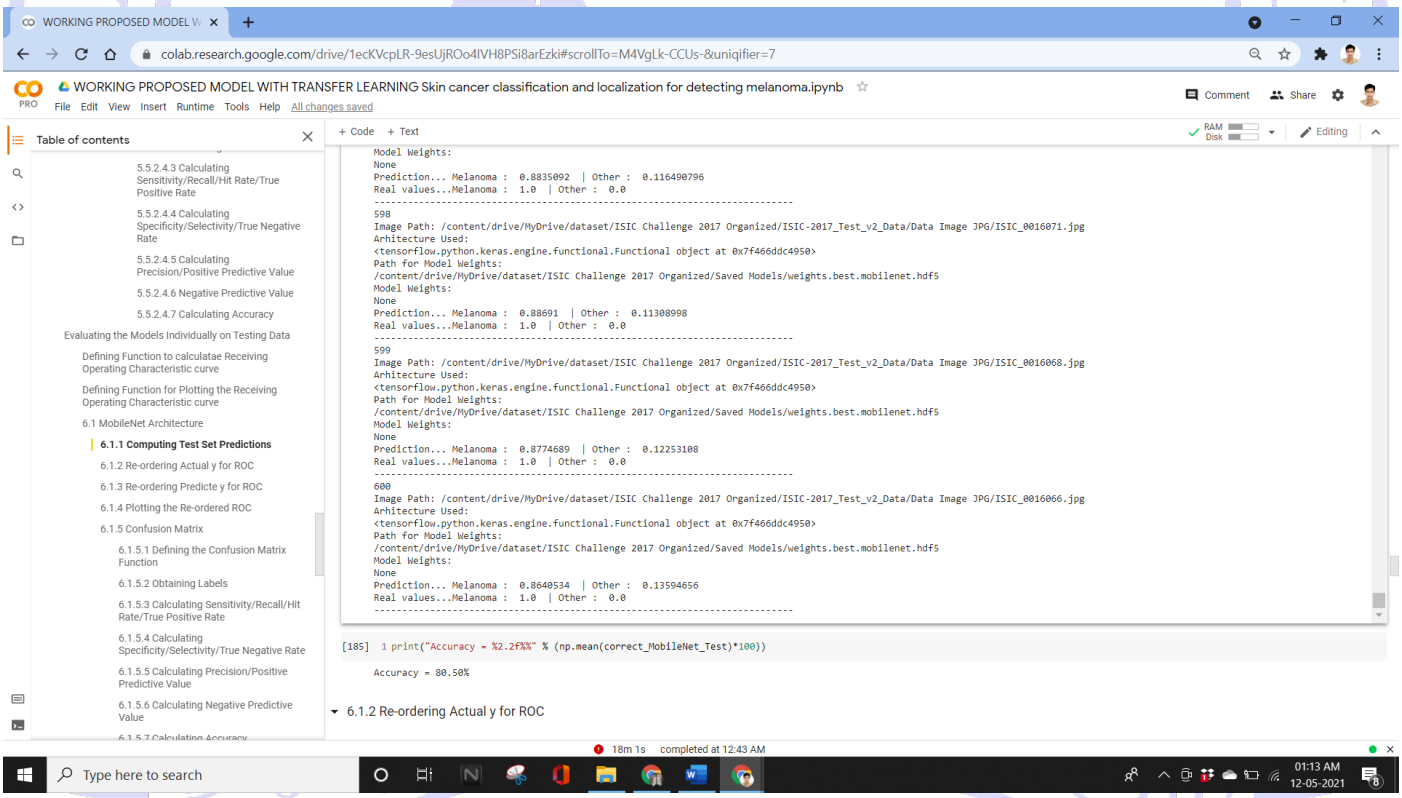

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaining the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
5.5.2.3 Plotting the Re-ordered ROC
5.5.2.4 Confusion Matrix
5.5.2.4.1 Defining the Confusion Matrix Function
5.5.2.4.2 Obtaining the Labels
5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
5.5.2.4.5 Calculating Precision/Positive Predictive Value
5.5.2.4.6 Negative Predictive Value
5.5.2.4.7 Calculating Accuracy
+ Code + Text
RAM Disk
Editing
5.5.2.4.6 Negative Predictive Value
[175] 1 # Negative predictive value (NPV)
2 try:
3 npv_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
4 print("Negative predictive value: ", npv_Ensemble)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
5.5.2.4.7 Calculating Accuracy
[176] 1 # Accuracy
2 accuracy_Ensemble = (TRUE_POSITIVE_Ensemble + TRUE_NEGATIVE_Ensemble) / (TRUE_POSITIVE_Ensemble + FALSE_POSITIVE_Ensemble + TRUE_NEGATIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
3 print("Accuracy: ", accuracy_Ensemble)
Accuracy: 0.8
6. Evaluating the Models Individually on Testing Data
[177] 1 # Importing the libraries
2 from sklearn.metrics import roc_curve, auc
3 import tqdm
4 import matplotlib.pyplot as plt
5 %matplotlib inline
Defining Function to calculatae Receiving Operating Characteristic curve
[178] 1 def compute_roc(y_true, y_score):
2 """
3 Computing the "Receiving Operating Characteristic curve" and area
4 """
5 false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)
6 auroc = auc(false_positive_rate, true_positive_rate)
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

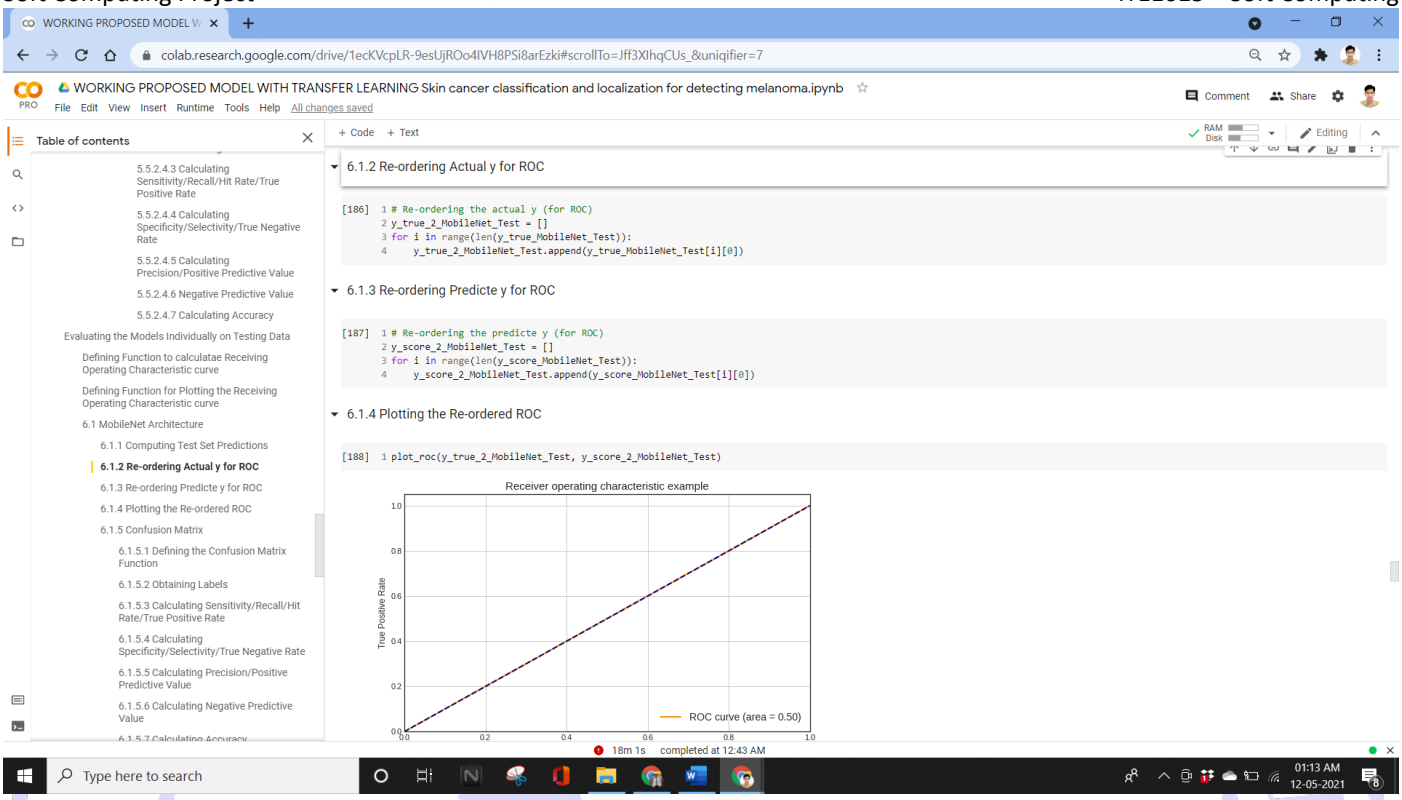

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
4.3.2.4.5 Calculating Precision/Positive Predictive Value
4.3.2.4.6 Negative Predictive Value
4.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Validation Data - Ensembling the models
5.1 Defining the Input Shape
5.2 Defining all the Models
5.3 Appending All the Models
5.4 Defining the Ensembling Function
5.5 Obtaining the Weights of all the Models combined
5.5 Evaluating ensemble model
5.5.1 Compute Test Set Predictions
5.5.2 Evaluating the Model
5.5.2.1 Re-ordering the Actual y for ROC
5.5.2.2 Re-ordering the Predict y for ROC
5.5.2.3 Plotting the Re-ordered ROC
5.5.2.4 Confusion Matrix
5.5.2.4.1 Defining the Confusion Matrix Function
5.5.2.4.2 Obtaining the Labels
5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
5.5.2.4.5 Calculating Precision/Positive Predictive Value
5.5.2.4.6 Negative Predictive Value
5.5.2.4.7 Calculating Accuracy
+ Code + Text
RAM Disk
Editing
[177] 1 # Importing the libraries
2 from sklearn.metrics import roc_curve, auc
3 import tqdm
4 import matplotlib.pyplot as plt
5 %matplotlib inline
Defining Function to calculatae Receiving Operating Characteristic curve
[178] 1 def compute_roc(y_true, y_score):
2 """
3 Computing the "Receiving Operating Characteristic curve" and area
4 """
5 false_positive_rate, true_positive_rate, thresholds = roc_curve(y_true, y_score)
6 auroc = auc(false_positive_rate, true_positive_rate)
7 return false_positive_rate, true_positive_rate, auroc
Defining Function for Plotting the Receiving Operating Characteristic curve
[179] 1 def plot_roc(y_true, y_score):
2 """
3 Ploting the Receiving Operating Characteristic curve
4 """
5 false_positive_rate, true_positive_rate, auroc = compute_roc(y_true, y_score)
6 plt.figure(figsize=(10,6))
7 plt.grid()
8 plt.plot(false_positive_rate,
9 true_positive_rate,
10 color='darkorange',
11 lw=2,
12 label='ROC curve (area = {:.2f})'.format(auroc))
13 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
14 plt.xlim([0.0, 1.0])
15 plt.ylim([0.0, 1.05])
16 plt.xlabel('False Positive Rate', fontsize=12)
17 plt.ylabel('True Positive Rate', fontsize=12)
18 plt.title('Receiver operating characteristic example', fontsize=15)
19 plt.legend(loc="lower right", fontsize=14)
20 plt.show()
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

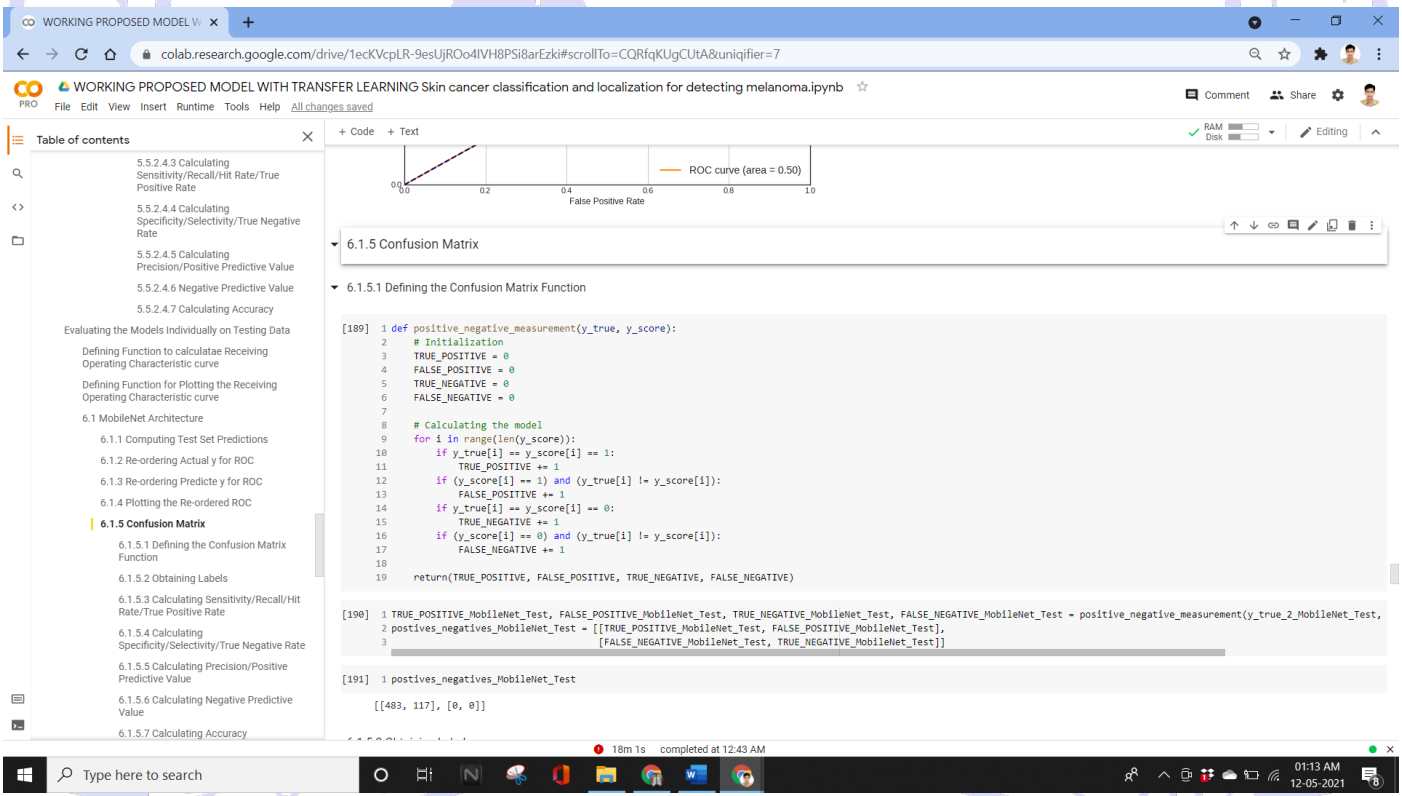

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
plt.title('Receiver operating characteristic example', fontsize=15)
plt.legend(loc="lower right", fontsize=14)
plt.show()
[180] 1 plt.style.available
['Solarize_Light2',
'_classic_test_patch',
'bmh',
'classic',
'dark_background',
'fast',
'fivethirtyeight',
'ggplot',
'grayscale',
'seaborn',
'seaborn-bright',
'seaborn-colorblind',
'seaborn-dark',
'seaborn-dark-palette',
'seaborn-darkgrid',
'seaborn-deep',
'seaborn-muted',
'seaborn-notebook',
'seaborn-paper',
'seaborn-pastel',
'seaborn-poster',
'seaborn-talk',
'seaborn-ticks',
'seaborn-white',
'seaborn-whitegrid',
'tableau-colorblind10']
[181] 1 plt.style.use("seaborn-white")
6.1 MobileNet Architecture
6.1.1 Computing Test Set Predictions
[182] 1 print("No. of Files in Test Data")
2 print(len(test_files))
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

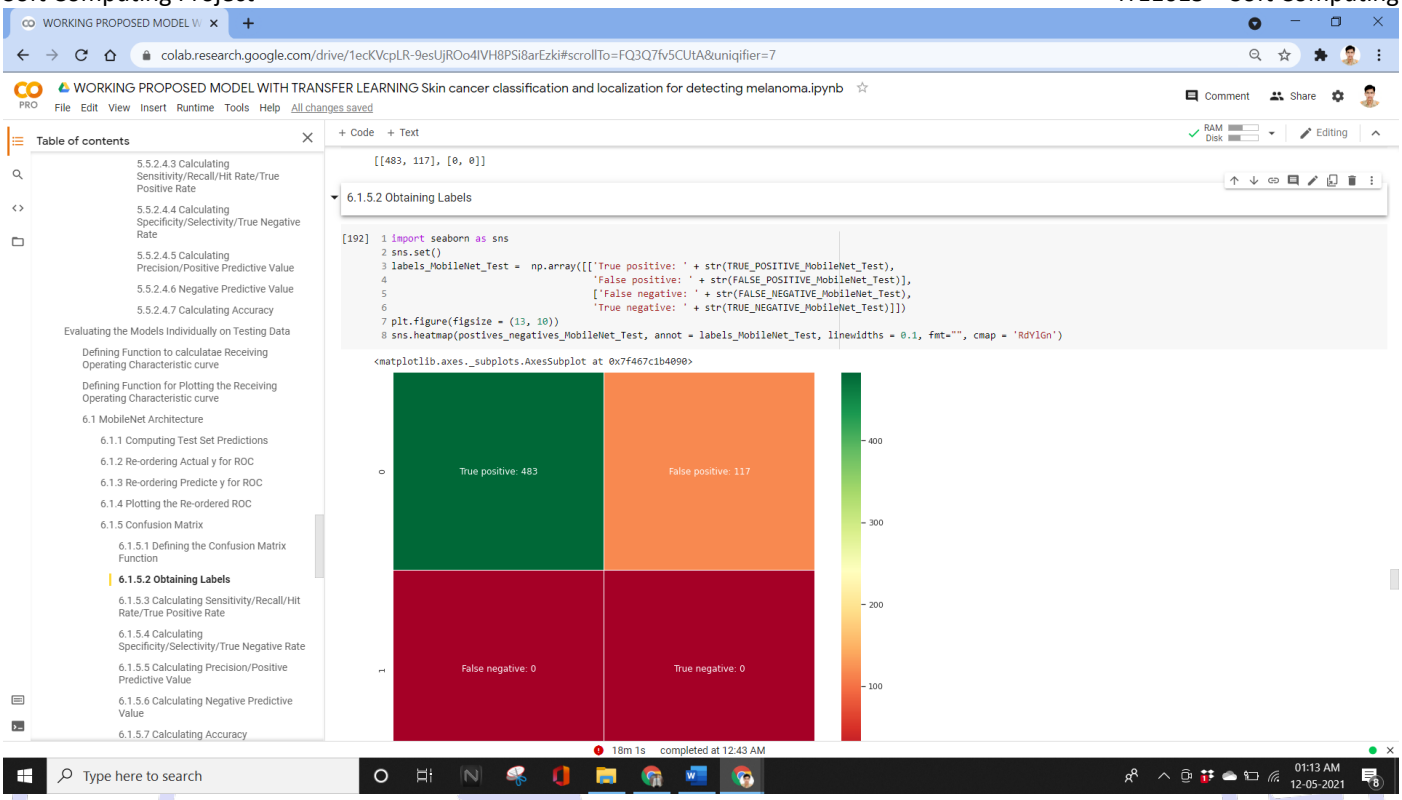

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
1 print("No. of Files in Test Data")
2 print(len(test_files))
3 print("No. of Target Values in Test Data")
4 print(len(test_targets))
No. of Files in Test Data
600
No. of Target Values in Test Data
600
[183] 1 # Compute test set predictions
2 NUMBER_TEST_SAMPLES_MobileNet_Test = 600
3
4 mobilenet_architecture_function = mobilenet_architecture()
5 mobilenet_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
6
7 y_true_MobileNet_Test = test_targets[:NUMBER_TEST_SAMPLES_MobileNet_Test]
8 print(y_true_MobileNet_Test)
Model: "model_12"
Layer (type) Output Shape Param #
input_10 (InputLayer) [(None, 512, 512, 3)] 0
conv1 (Conv2D) (None, 256, 256, 32) 864
conv1_bn (BatchNormalization (None, 256, 256, 32) 128
conv1_relu (ReLU) (None, 256, 256, 32) 0
conv_dw_1 (DepthwiseConv2D) (None, 256, 256, 32) 288
conv_dw_1_bn (BatchNormaliza (None, 256, 256, 32) 128
conv_dw_1_relu (ReLU) (None, 256, 256, 32) 0
conv_pw_1 (Conv2D) (None, 256, 256, 64) 2048
conv_pw_1_bn (BatchNormaliza (None, 256, 256, 64) 256
conv_pw_1_relu (ReLU) (None, 256, 256, 64) 0
conv_pad_2 (ZeroPadding2D) (None, 257, 257, 64) 0
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

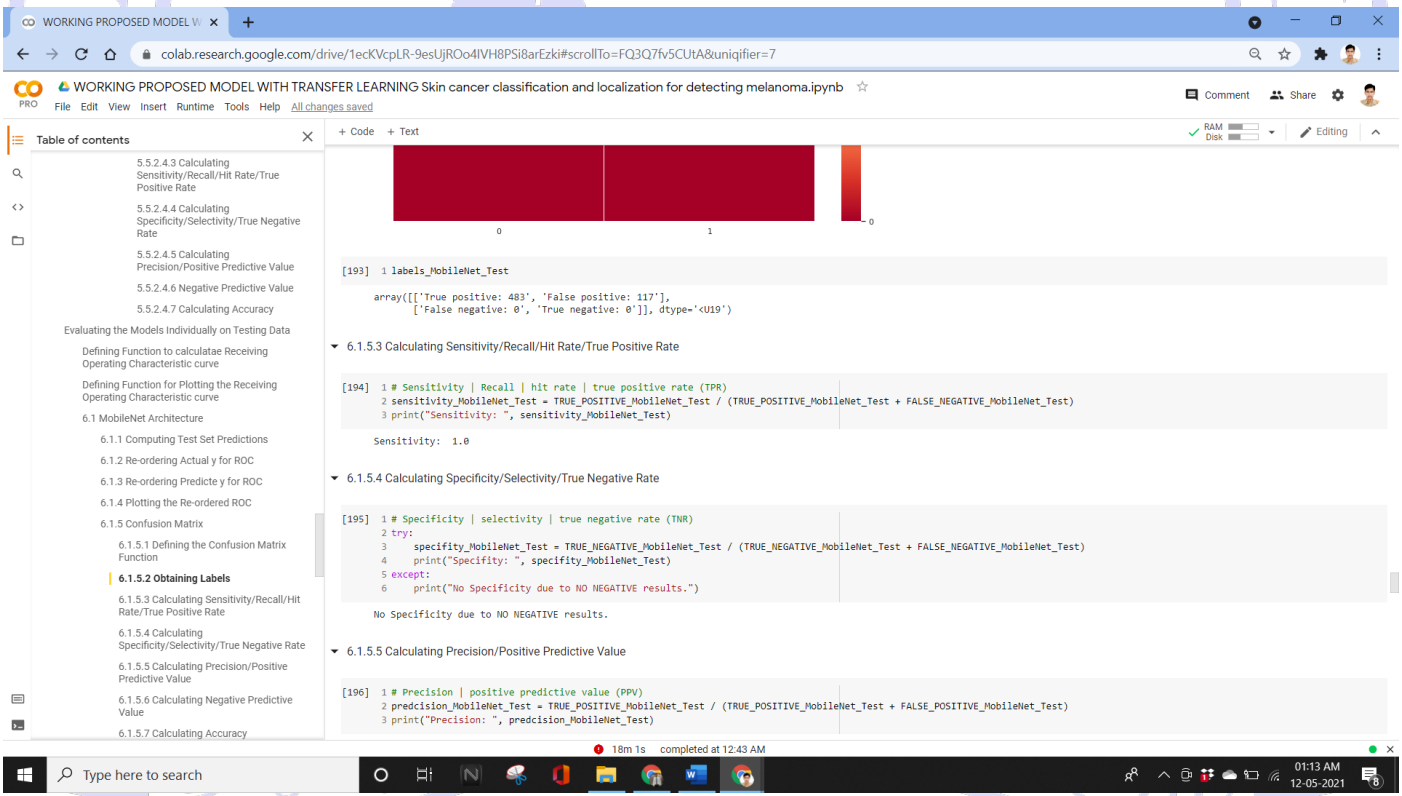

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=mh9c7giT_hhN&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
[[1. 0.]
[1. 0.]
[1. 0.]
...
[1. 0.]
[1. 0.]
[1. 0.]]
int: NUMBER_TEST_SAMPLES_MobileNet_Test
600
1 y_score_MobileNet_T
2 count = 0
3 for index in range(NUMBER_TEST_SAMPLES_MobileNet_Test): #compute one at a time due to memory constraints
4 count = count+1
5 print(count)
6 probs_MobileNet_Test = predict(img_path = test_files[index], model_architecture = mobilenet_architecture_function, path_model_weight = mobilenet_architecture_weight_path)
7 print("Real values..." + "Melanoma : ", test_targets[index][0], " | Other : ", test_targets[index][1])
8 print("---")
9 y_score_MobileNet_Test.append(probs_MobileNet_Test)
10
11 correct_MobileNet_Test = np.array(y_true_MobileNet_Test) == np.array(y_score_MobileNet_Test)
Streaming output truncated to the last 5000 lines.
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.87128127 | Other : 0.12871875
Real values...Melanoma : 1.0 | Other : 0.0
147
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013891.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466ddc4950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.82431066 | Other : 0.17568932
Real values...Melanoma : 0.0 | Other : 1.0
148
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013998.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466ddc4950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
18m 1s completed at 12:43 AM
Type here to search
01:12 AM
12-05-2021

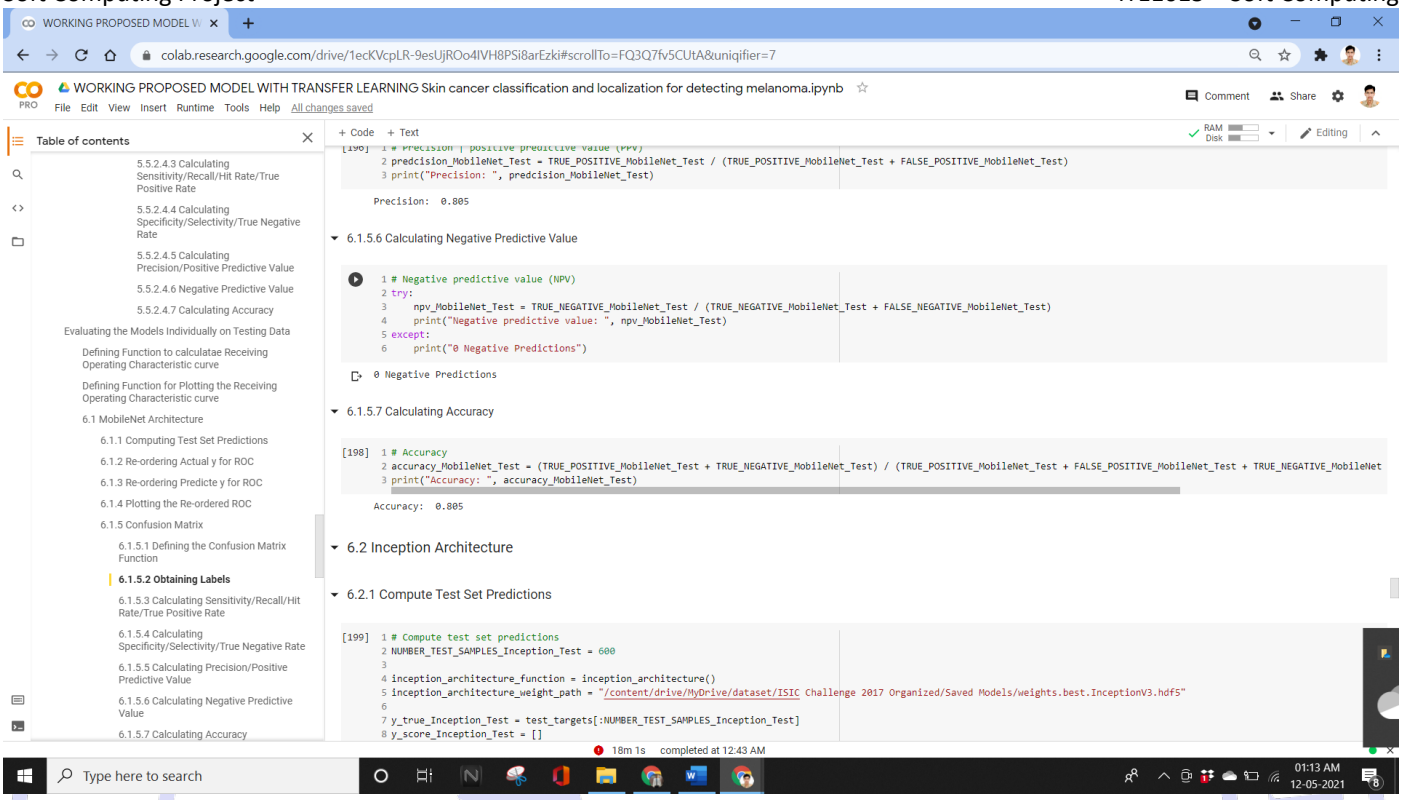

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=M4VgLk-CCUs-&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Model Weights:
None
Prediction... Melanoma : 0.8835092 | Other : 0.116490796
Real values...Melanoma : 1.0 | Other : 0.0
598
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016071.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466ddc4950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.88691 | Other : 0.11308998
Real values...Melanoma : 1.0 | Other : 0.0
599
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016068.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466ddc4950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8774689 | Other : 0.12253108
Real values...Melanoma : 1.0 | Other : 0.0
600
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016066.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466ddc4950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8640534 | Other : 0.13594656
Real values...Melanoma : 1.0 | Other : 0.0
[185] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_MobileNet_Test)*100))
Accuracy = 80.50%
6.1.2 Re-ordering Actual y for ROC
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

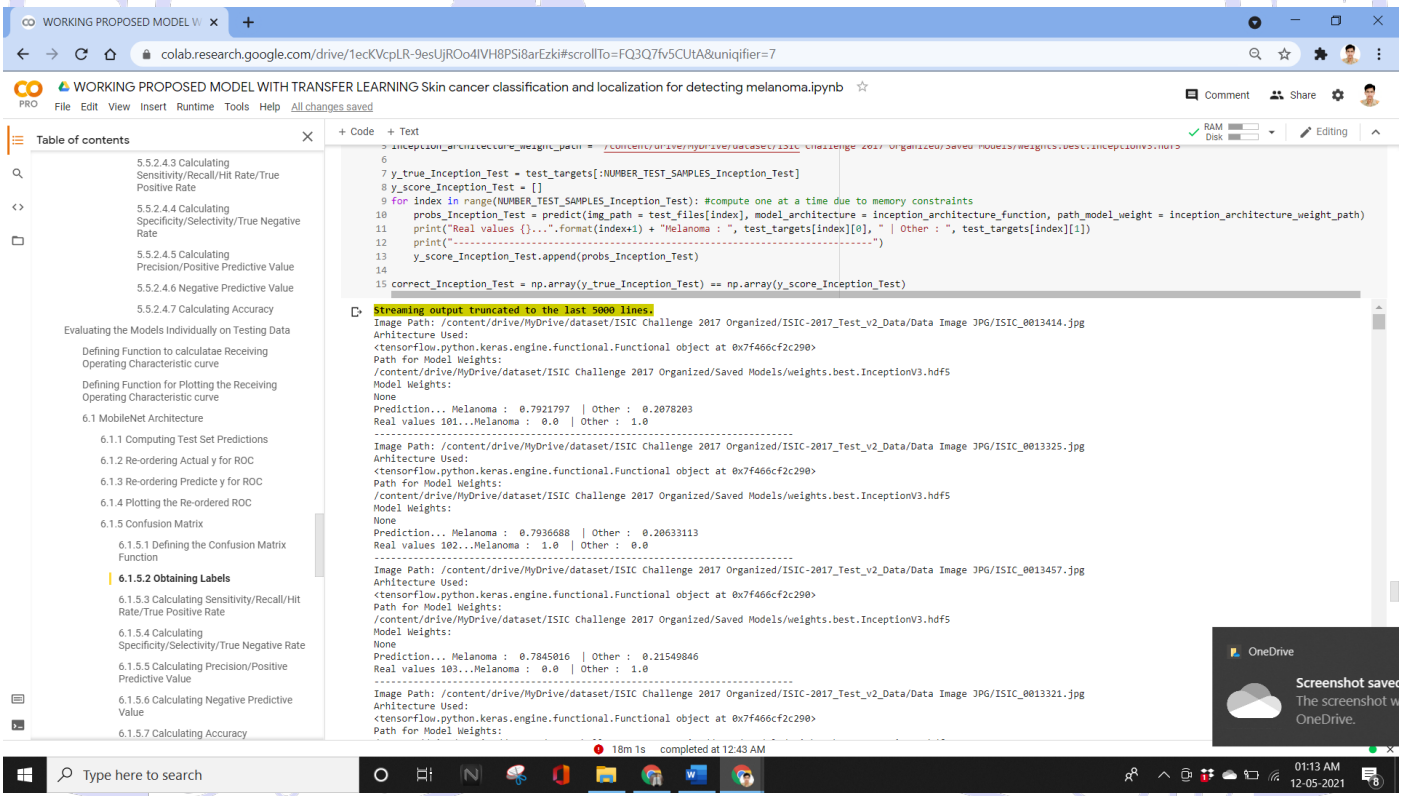

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=Jff3XIhqCUs_&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.1.2 Re-ordering Actual y for ROC
[186] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_MobileNet_Test = []
3 for i in range(len(y_true_MobileNet_Test)):
4 y_true_2_MobileNet_Test.append(y_true_MobileNet_Test[i][0])
6.1.3 Re-ordering Predicte y for ROC
[187] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_MobileNet_Test = []
3 for i in range(len(y_score_MobileNet_Test)):
4 y_score_2_MobileNet_Test.append(y_score_MobileNet_Test[i][0])
6.1.4 Plotting the Re-ordered ROC
[188] 1 plot_roc(y_true_2_MobileNet_Test, y_score_2_MobileNet_Test)
Receiver operating characteristic example
True Positive Rate
ROC curve (area = 0.50)
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

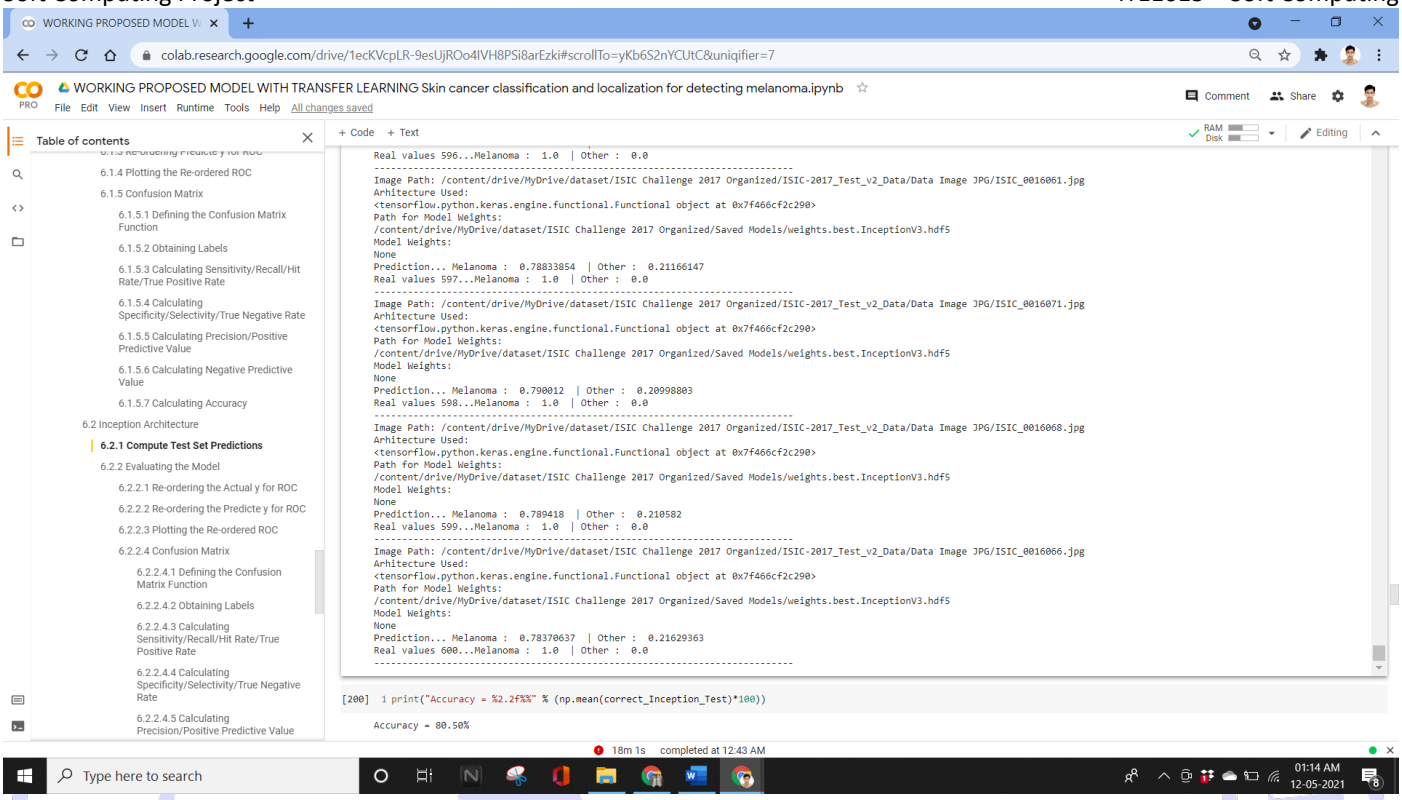

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=CQRfqKUgCUtA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
ROC curve (area = 0.50)
False Positive Rate
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
[189] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[190] 1 TRUE_POSITIVE_MobileNet_Test, FALSE_POSITIVE_MobileNet_Test, TRUE_NEGATIVE_MobileNet_Test, FALSE_NEGATIVE_MobileNet_Test = positive_negative_measurement(y_true_2_MobileNet_Test,
2 postives_negatives_MobileNet_Test = [[TRUE_POSITIVE_MobileNet_Test, FALSE_POSITIVE_MobileNet_Test],
3 [FALSE_NEGATIVE_MobileNet_Test, TRUE_NEGATIVE_MobileNet_Test]]
[191] 1 postives_negatives_MobileNet_Test
[[483, 117], [0, 0]]
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

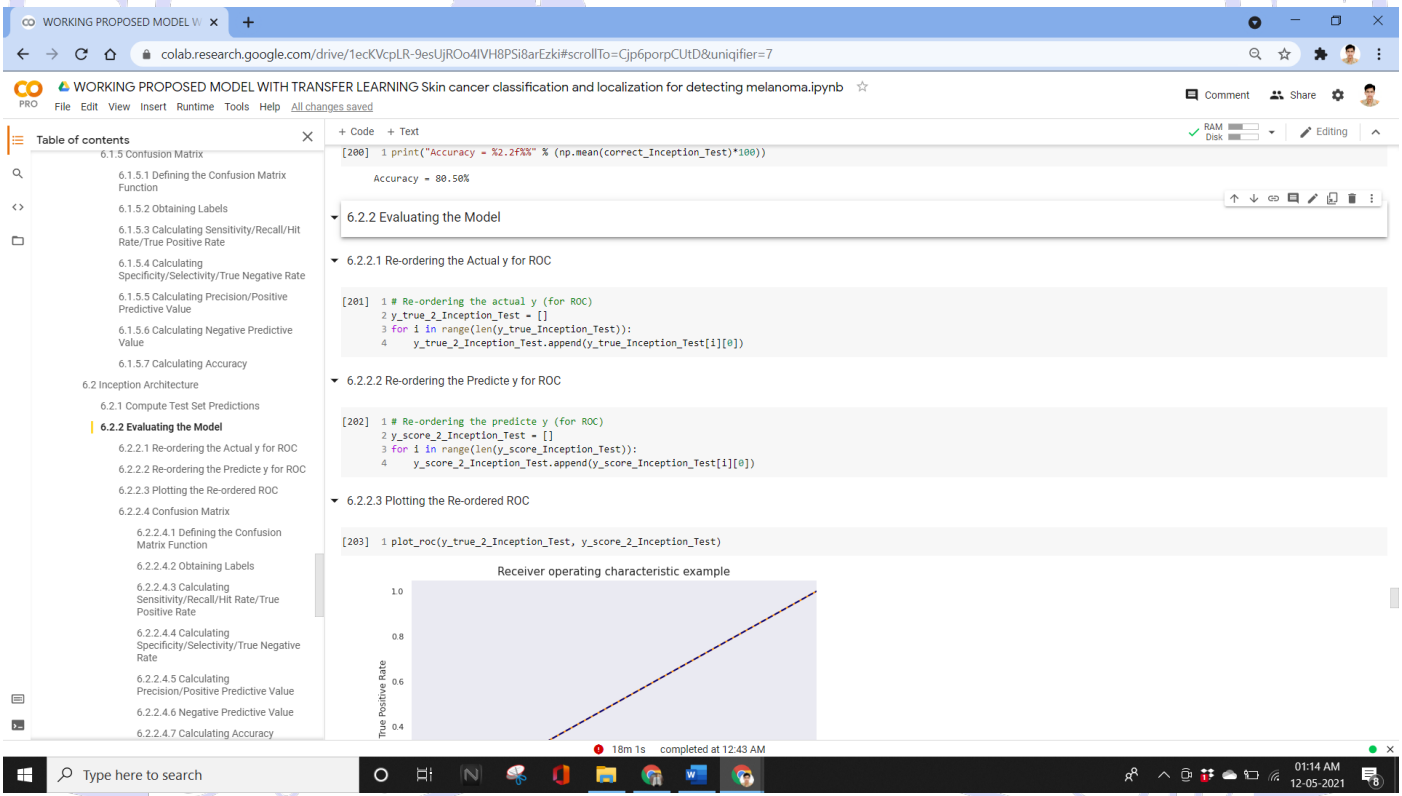

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=FQ3Q7fv5CUtA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
[[483, 117], [0, 0]]
6.1.5.2 Obtaining Labels
[192] 1 import seaborn as sns
2 sns.set()
3 labels_MobileNet_Test = np.array([['True positive: ' + str(TRUE_POSITIVE_MobileNet_Test),
4 'False positive: ' + str(FALSE_POSITIVE_MobileNet_Test)],
5 ['False negative: ' + str(FALSE_NEGATIVE_MobileNet_Test),
6 'True negative: ' + str(TRUE_NEGATIVE_MobileNet_Test)]])
7 plt.figure(figsize = (13, 10))
8 sns.heatmap(postives_negatives_MobileNet_Test, annot = labels_MobileNet_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f467c1b4090>
True positive: 483
False positive: 117
False negative: 0
True negative: 0
400
300
200
100
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

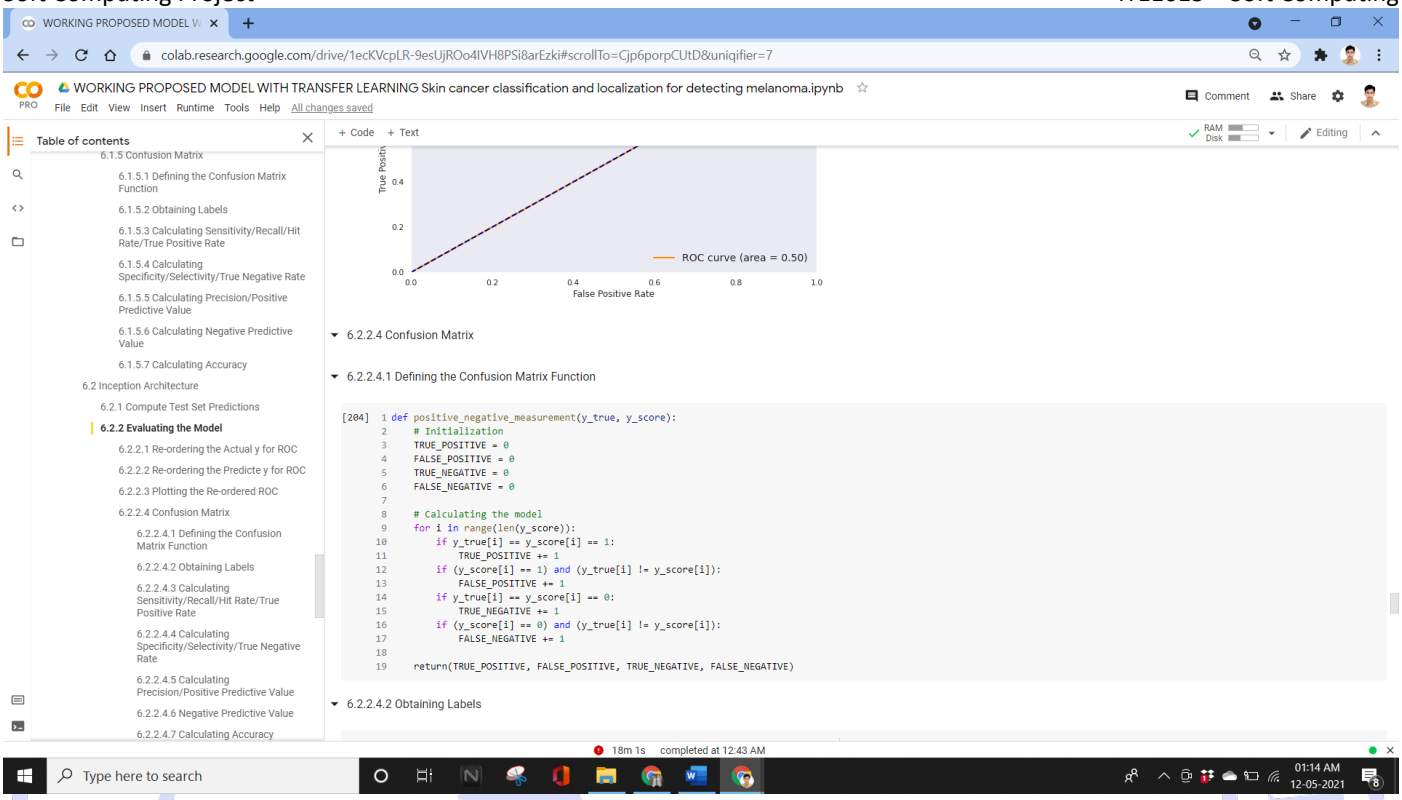

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=FQ3Q7fv5CUtA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
0
1
0
[193] 1 labels_MobileNet_Test
array([['True positive: 483', 'False positive: 117'],
['False negative: 0', 'True negative: 0']], dtype='<U19')
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[194] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test / (TRUE_POSITIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)
3 print("Sensitivity: ", sensitivity_MobileNet_Test)
Sensitivity: 1.0
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
[195] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_MobileNet_Test = TRUE_NEGATIVE_MobileNet_Test / (TRUE_NEGATIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)
4 print("Specifity: ", specifity_MobileNet_Test)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
6.1.5.5 Calculating Precision/Positive Predictive Value
[196] 1 # Precision | positive predictive value (PPV)
2 predcision_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test / (TRUE_POSITIVE_MobileNet_Test + FALSE_POSITIVE_MobileNet_Test)
3 print("Precision: ", predcision_MobileNet_Test)
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

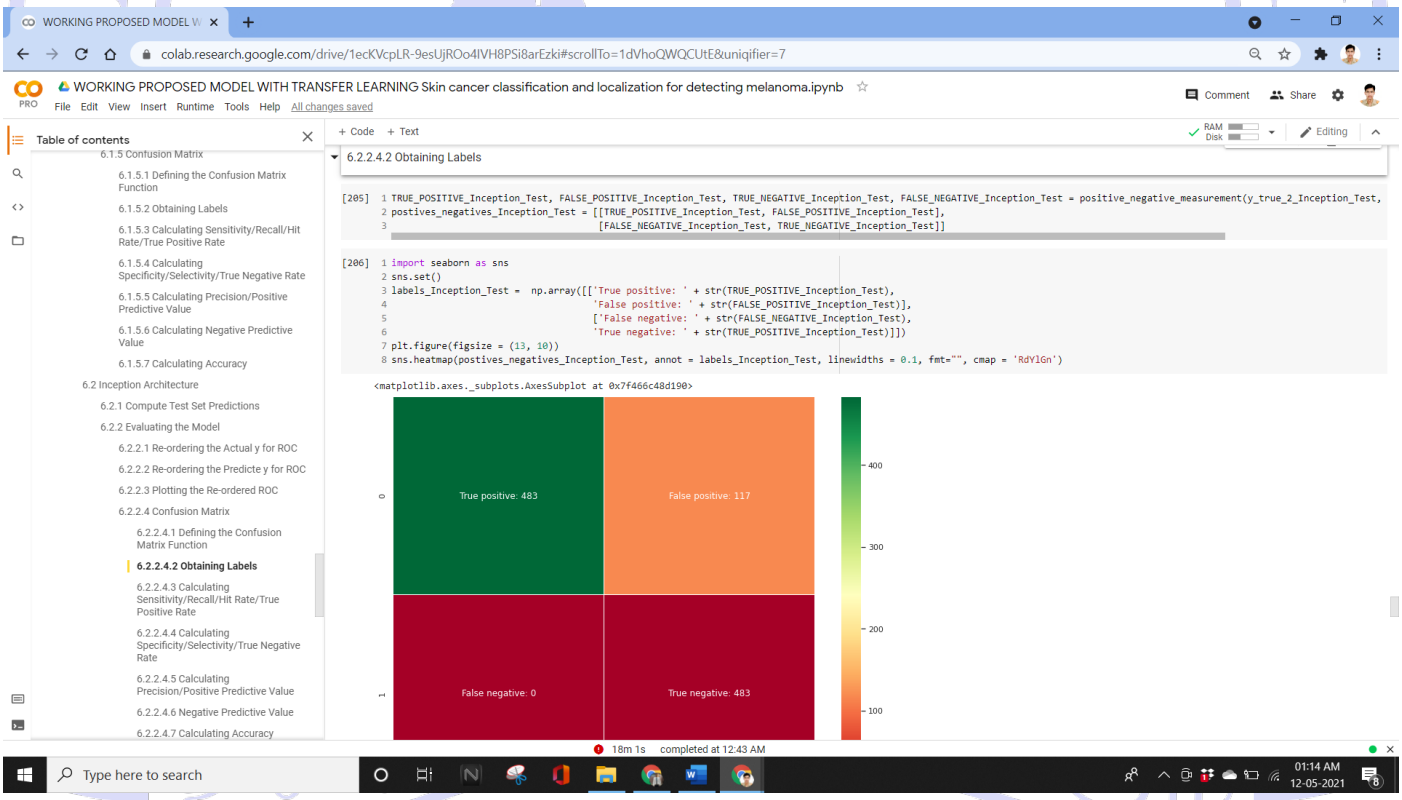

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=FQ3Q7fv5CUtA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Precision: 0.805
6.1.5.6 Calculating Negative Predictive Value
0 Negative Predictions
6.1.5.7 Calculating Accuracy
Accuracy: 0.805
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

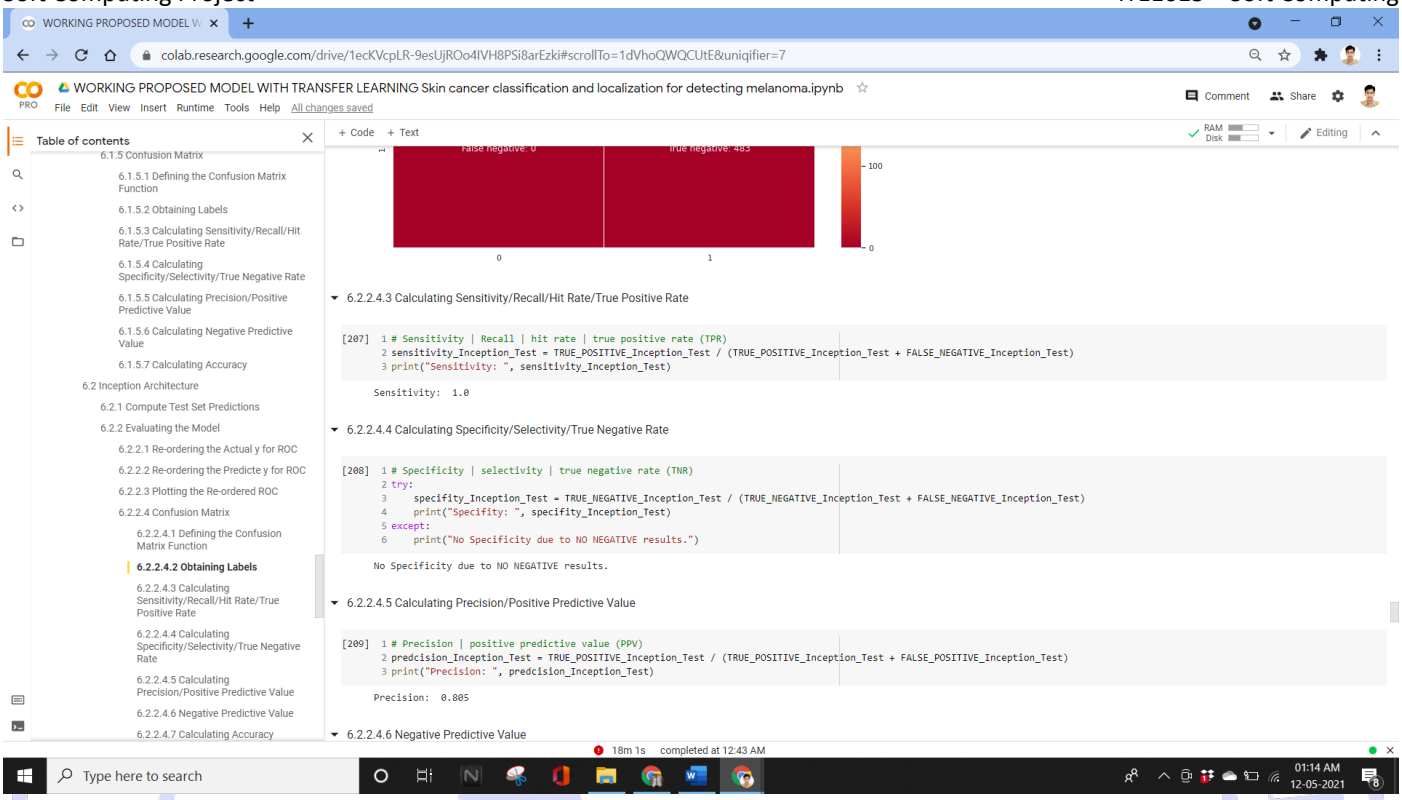

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=FQ3Q7fv5CUtA&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Streaming output truncated to the last 5000 lines.
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013414.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.7921797 | Other : 0.2078203
Real values 101...Melanoma : 0.0 | Other : 1.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013325.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.7936688 | Other : 0.20633113
Real values 102...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013457.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.7845016 | Other : 0.21549846
Real values 103...Melanoma : 0.0 | Other : 1.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0013321.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
OneDrive
Screenshot saved
The screenshot w
OneDrive.
18m 1s completed at 12:43 AM
Type here to search
01:13 AM
12-05-2021

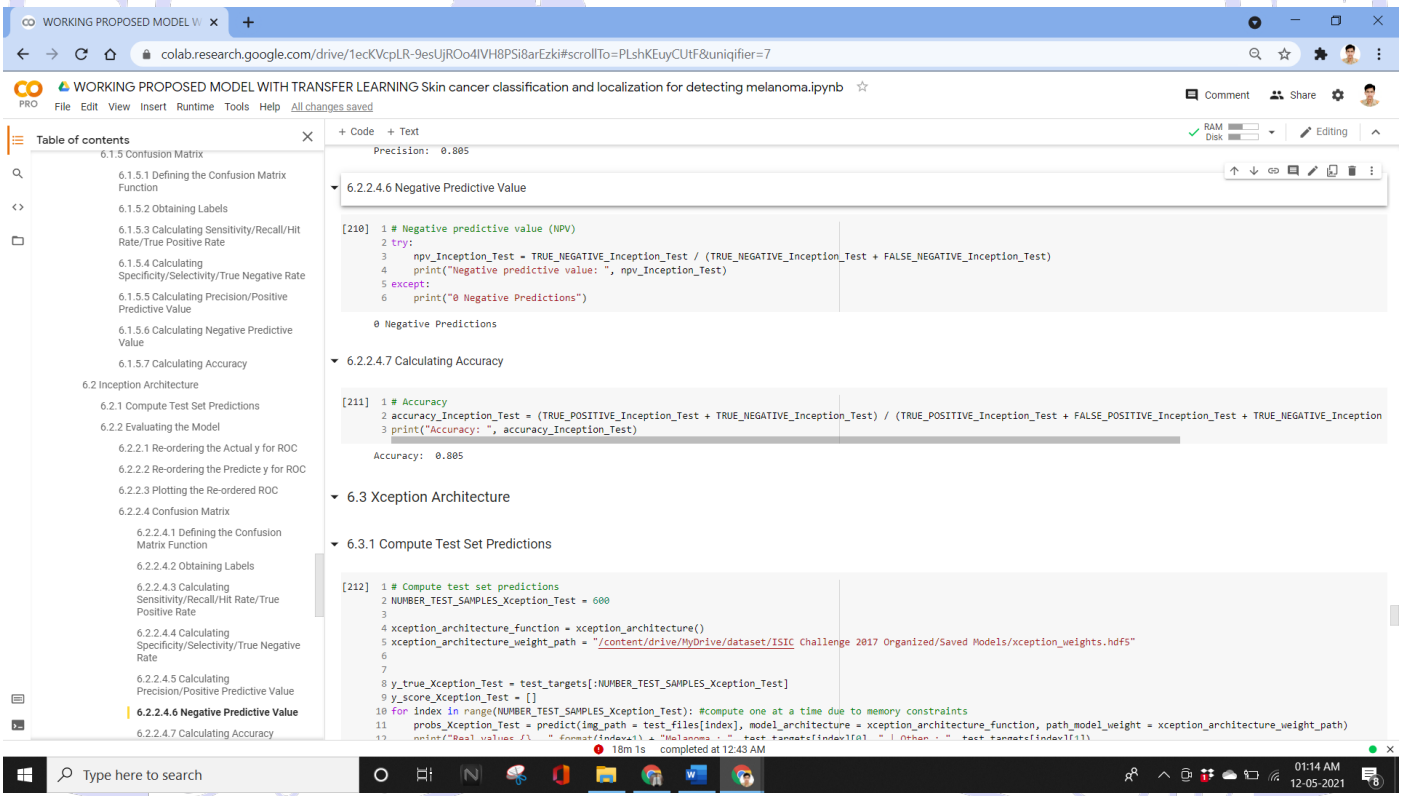

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=yKb6S2nYCUtC&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Real values 596...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016061.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.78833854 | Other : 0.21166147
Real values 597...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016071.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.790012 | Other : 0.20998803
Real values 598...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016068.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.789418 | Other : 0.210582
Real values 599...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016066.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466cf2c290>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma : 0.78370637 | Other : 0.21629363
Real values 600...Melanoma : 1.0 | Other : 0.0
[200] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Inception_Test)*100))
Accuracy = 80.50%
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

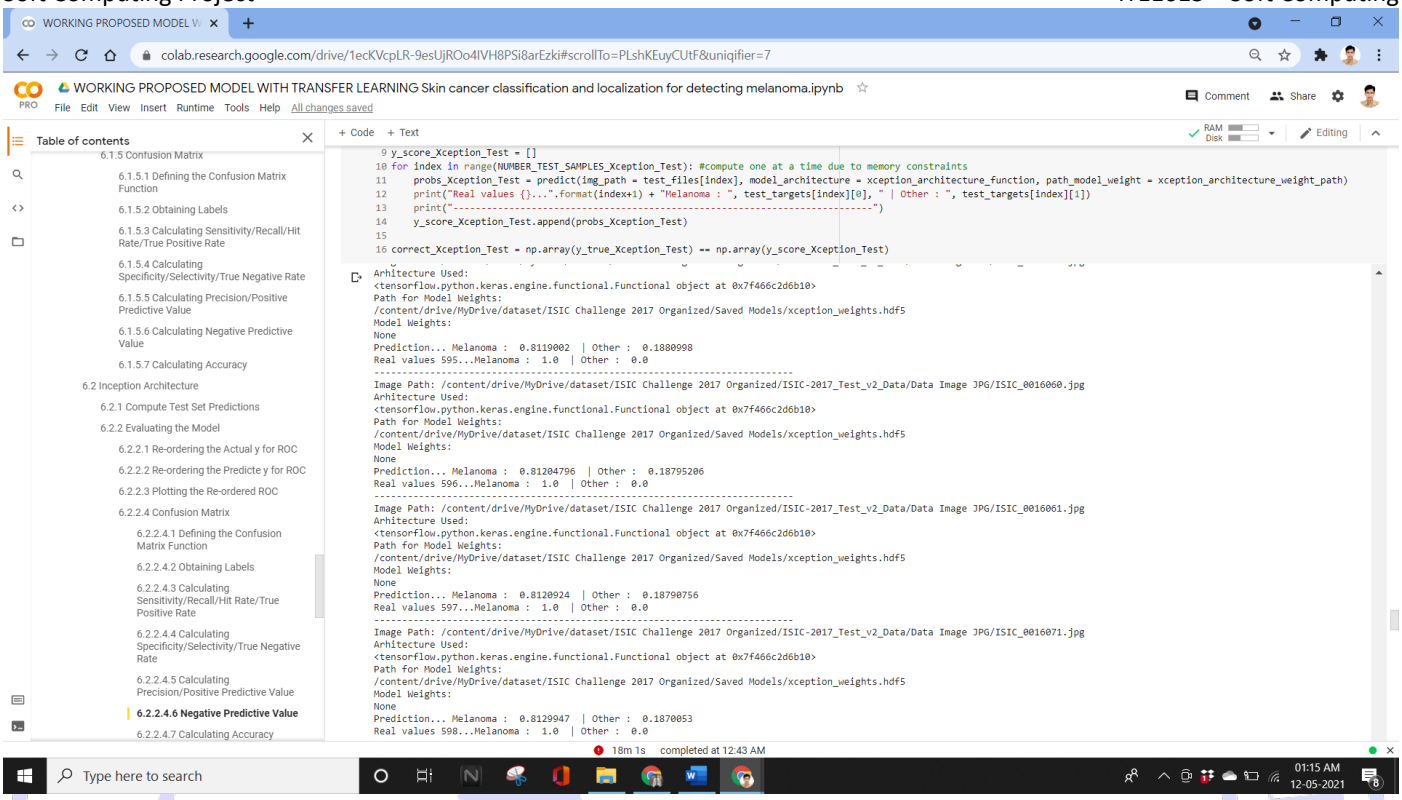

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=Cjp6porpCUtD&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
[200] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Inception_Test)*100))
Accuracy = 80.50%
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
[201] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Inception_Test = []
3 for i in range(len(y_true_Inception_Test)):
4 y_true_2_Inception_Test.append(y_true_Inception_Test[i][0])
6.2.2.2 Re-ordering the Predicte y for ROC
[202] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Inception_Test = []
3 for i in range(len(y_score_Inception_Test)):
4 y_score_2_Inception_Test.append(y_score_Inception_Test[i][0])
6.2.2.3 Plotting the Re-ordered ROC
[203] 1 plot_roc(y_true_2_Inception_Test, y_score_2_Inception_Test)
Receiver operating characteristic example
True Positive Rate
1.0
0.8
0.6
0.4
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

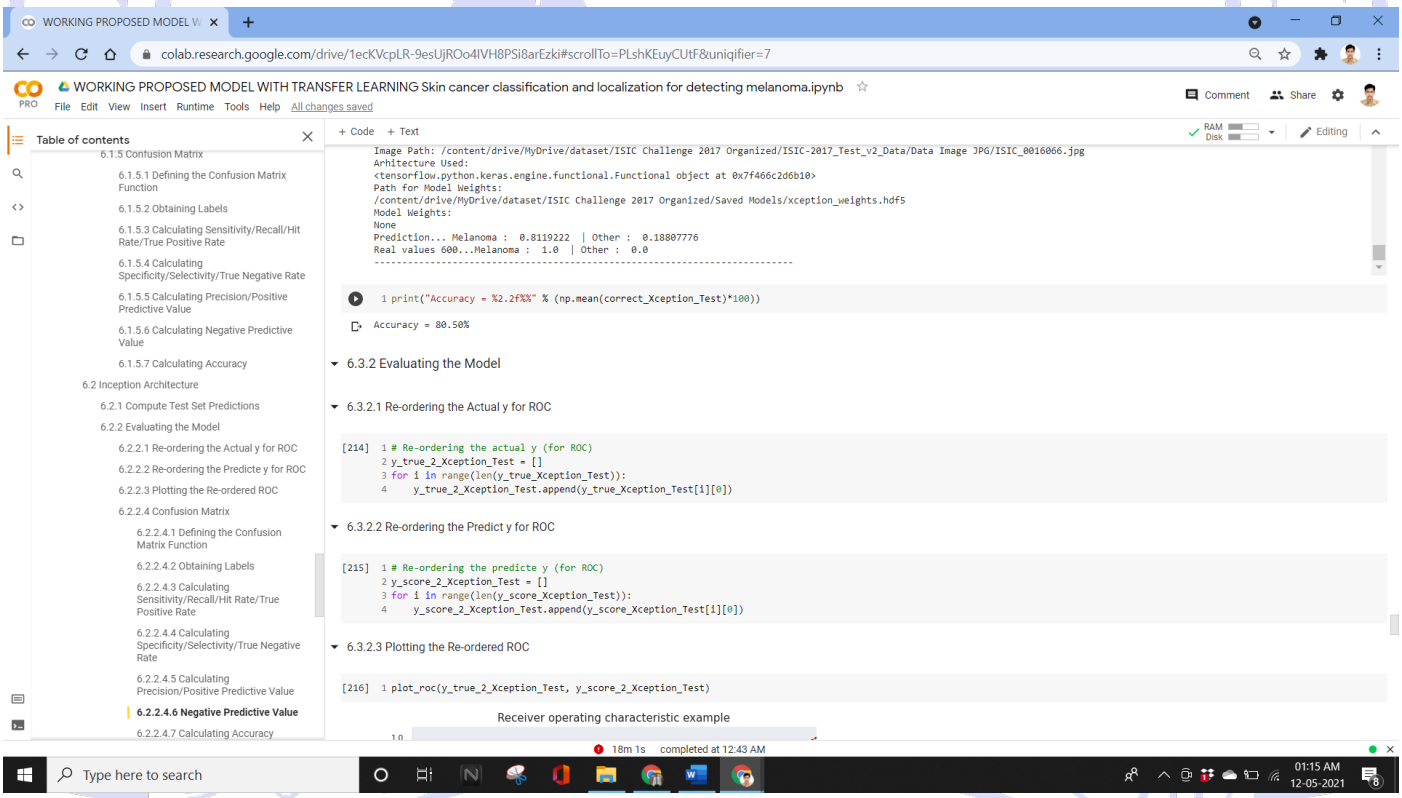

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=Cjp6porpCUtD&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
+ Code + Text
RAM Disk
Editing
True Positive Rate
False Positive Rate
ROC curve (area = 0.50)
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
[204]
def positive_negative_measurement(y_true, y_score):
# Initialization
TRUE_POSITIVE = 0
FALSE_POSITIVE = 0
TRUE_NEGATIVE = 0
FALSE_NEGATIVE = 0
# Calculating the model
for i in range(len(y_score)):
if y_true[i] == y_score[i] == 1:
TRUE_POSITIVE += 1
if (y_score[i] == 1) and (y_true[i] != y_score[i]):
FALSE_POSITIVE += 1
if y_true[i] == y_score[i] == 0:
TRUE_NEGATIVE += 1
if (y_score[i] == 0) and (y_true[i] != y_score[i]):
FALSE_NEGATIVE += 1
return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
6.2.2.4.2 Obtaining Labels
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

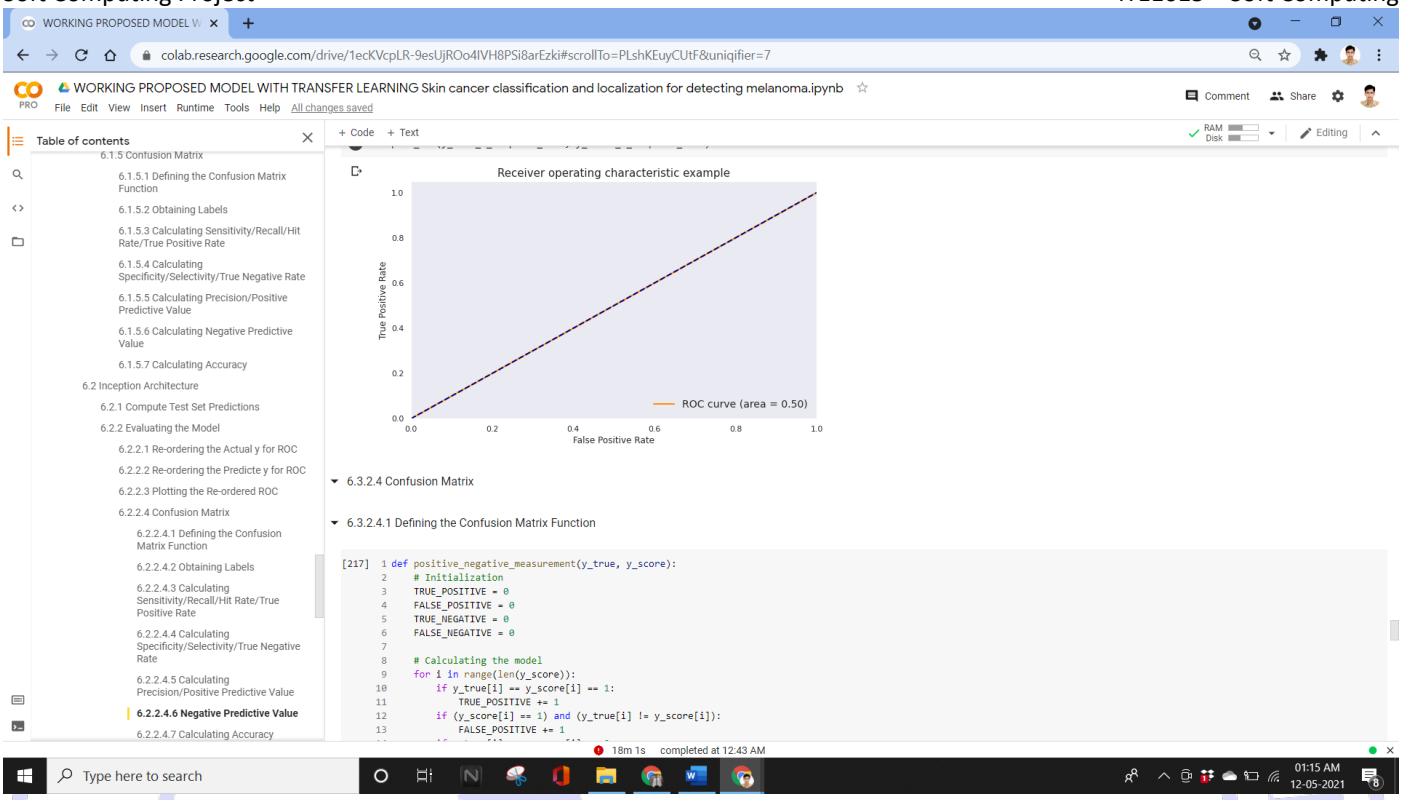

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1dVhoQWQCUtE&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
+ Code + Text
RAM Disk
Editing
6.2.2.4.2 Obtaining Labels
[205]
TRUE_POSITIVE_Inception_Test, FALSE_POSITIVE_Inception_Test, TRUE_NEGATIVE_Inception_Test, FALSE_NEGATIVE_Inception_Test = positive_negative_measurement(y_true_2_Inception_Test,
postives_negatives_Inception_Test = [[TRUE_POSITIVE_Inception_Test, FALSE_POSITIVE_Inception_Test],
[FALSE_NEGATIVE_Inception_Test, TRUE_NEGATIVE_Inception_Test]]
[206]
import seaborn as sns
sns.set()
labels_Inception_Test = np.array([['True positive: ' + str(TRUE_POSITIVE_Inception_Test),
'False positive: ' + str(FALSE_POSITIVE_Inception_Test)],
['False negative: ' + str(FALSE_NEGATIVE_Inception_Test),
'True negative: ' + str(TRUE_POSITIVE_Inception_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Inception_Test, annot = labels_Inception_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f466c48d190>
True positive: 483
False positive: 117
False negative: 0
True negative: 483
400
300
200
100
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

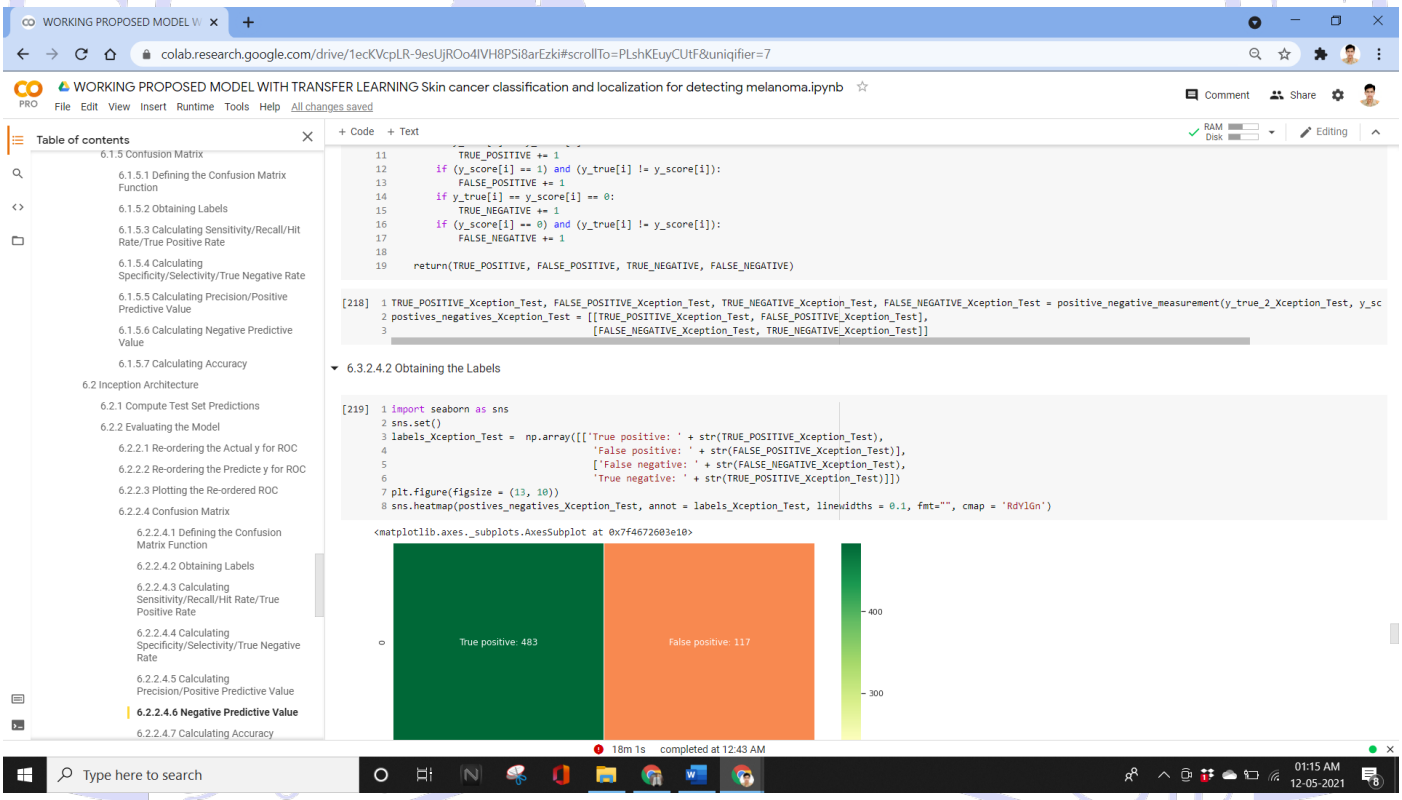

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1dVhoQWQCUtE&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
False Negative: 0
True Negative: 483
100
0
0
1
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[207] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_Inception_Test = TRUE_POSITIVE_Inception_Test / (TRUE_POSITIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
3 print("Sensitivity: ", sensitivity_Inception_Test)
Sensitivity: 1.0
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[208] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_Inception_Test = TRUE_NEGATIVE_Inception_Test / (TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
4 print("Specifity: ", specifity_Inception_Test)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
6.2.2.4.5 Calculating Precision/Positive Predictive Value
[209] 1 # Precision | positive predictive value (PPV)
2 predcision_Inception_Test = TRUE_POSITIVE_Inception_Test / (TRUE_POSITIVE_Inception_Test + FALSE_POSITIVE_Inception_Test)
3 print("Precision: ", predcision_Inception_Test)
Precision: 0.805
6.2.2.4.6 Negative Predictive Value
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

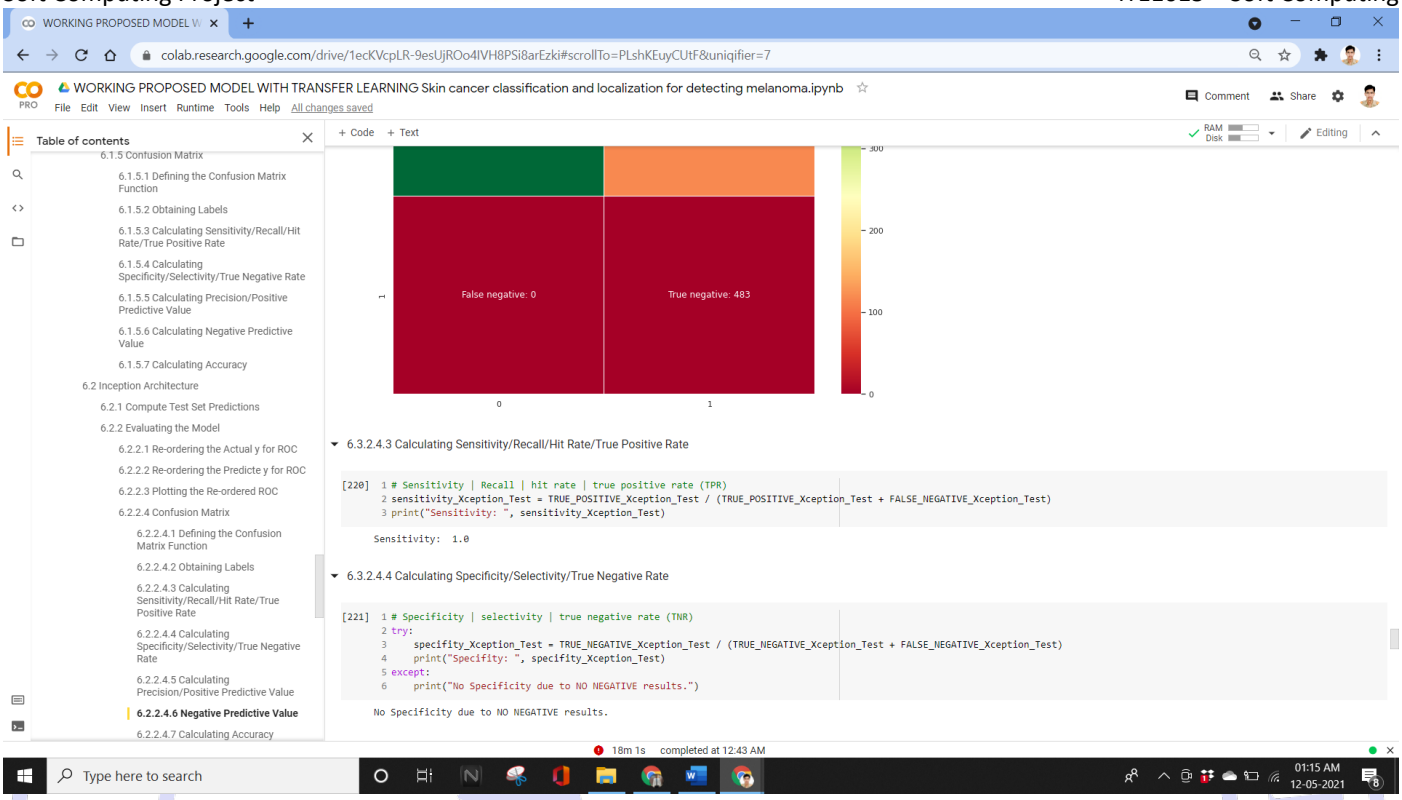

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
Precision: 0.805
6.2.2.4.6 Negative Predictive Value
[210] 1 # Negative predictive value (NPV)
2 try:
3 npv_Inception_Test = TRUE_NEGATIVE_Inception_Test / (TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
4 print("Negative predictive value: ", npv_Inception_Test)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
6.2.2.4.7 Calculating Accuracy
[211] 1 # Accuracy
2 accuracy_Inception_Test = (TRUE_POSITIVE_Inception_Test + TRUE_NEGATIVE_Inception_Test) / (TRUE_POSITIVE_Inception_Test + FALSE_POSITIVE_Inception_Test + TRUE_NEGATIVE_Inception
3 print("Accuracy: ", accuracy_Inception_Test)
Accuracy: 0.805
6.3 Xception Architecture
6.3.1 Compute Test Set Predictions
[212] 1 # Compute test set predictions
2 NUMBER_TEST_SAMPLES_Xception_Test = 600
3
4 xception_architecture_function = xception_architecture()
5 xception_architecture_weight_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5"
6
7
8 y_true_Xception_Test = test_targets[:NUMBER_TEST_SAMPLES_Xception_Test]
9 y_score_Xception_Test = []
10 for index in range(NUMBER_TEST_SAMPLES_Xception_Test): #compute one at a time due to memory constraints
11 probs_Xception_Test = predict(img_path = test_files[index], model_architecture = xception_architecture_function, path_model_weight = xception_architecture_weight_path)
18m 1s completed at 12:43 AM
Type here to search
01:14 AM
12-05-2021

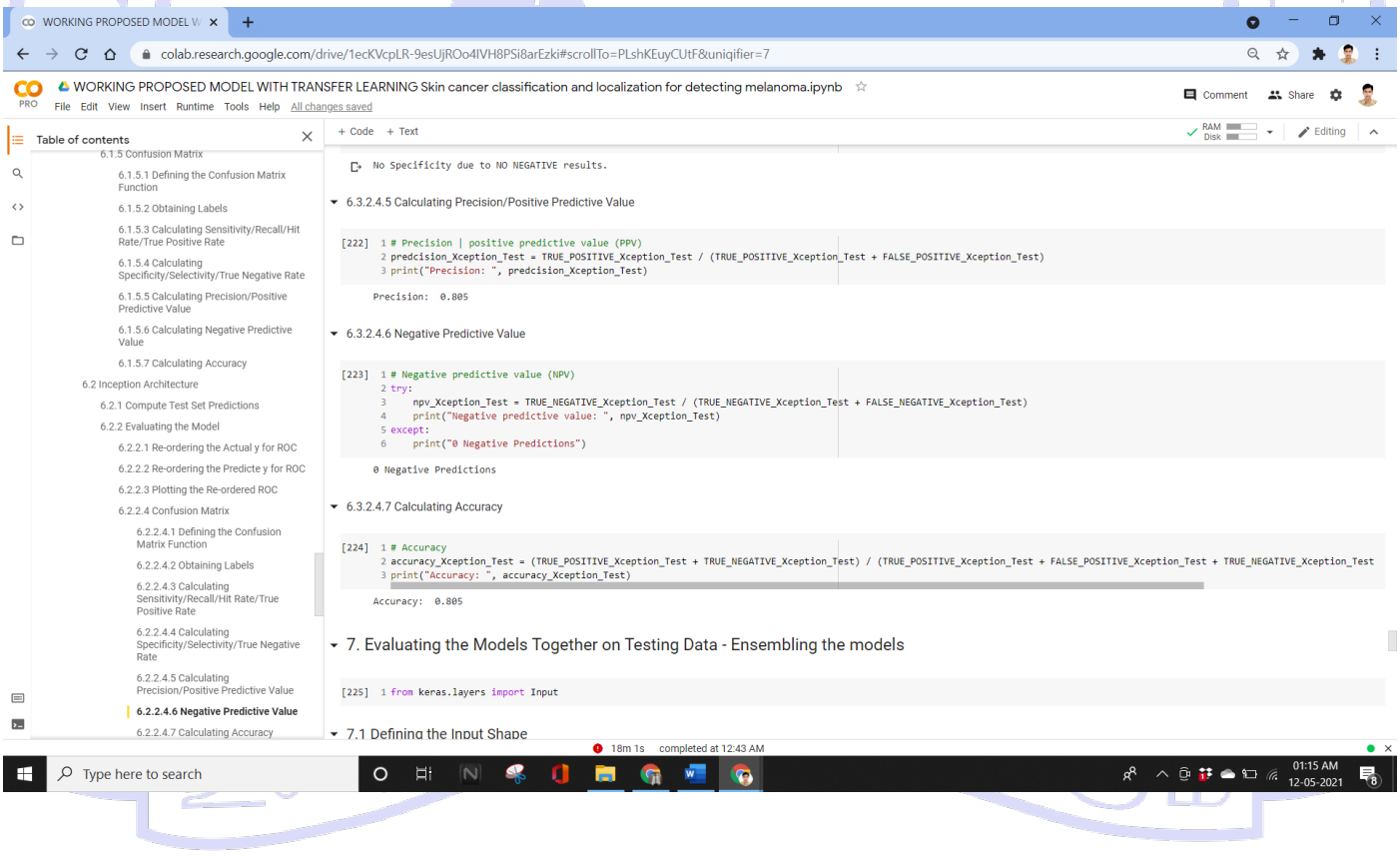

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
9 y_score_Xception_Test = []
10 for index in range(NUMBER_TEST_SAMPLES_Xception_Test): #compute one at a time due to memory constraints
11 probs_Xception_Test = predict(img_path = test_files[index], model_architecture = xception_architecture_function, path_model_weight = xception_architecture_weight_path)
12 print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0], " | Other : ", test_targets[index][1])
13 print("---")
14 y_score_Xception_Test.append(probs_Xception_Test)
15
16 correct_Xception_Test = np.array(y_true_Xception_Test) == np.array(y_score_Xception_Test)
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f466c2d6b10>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma : 0.8119002 | Other : 0.1880998
Real values 595...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016060.jpg
Prediction... Melanoma : 0.81204796 | Other : 0.18795206
Real values 596...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016061.jpg
Prediction... Melanoma : 0.8120924 | Other : 0.18790756
Real values 597...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016071.jpg
Prediction... Melanoma : 0.8129947 | Other : 0.1870053
Real values 598...Melanoma : 1.0 | Other : 0.0
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

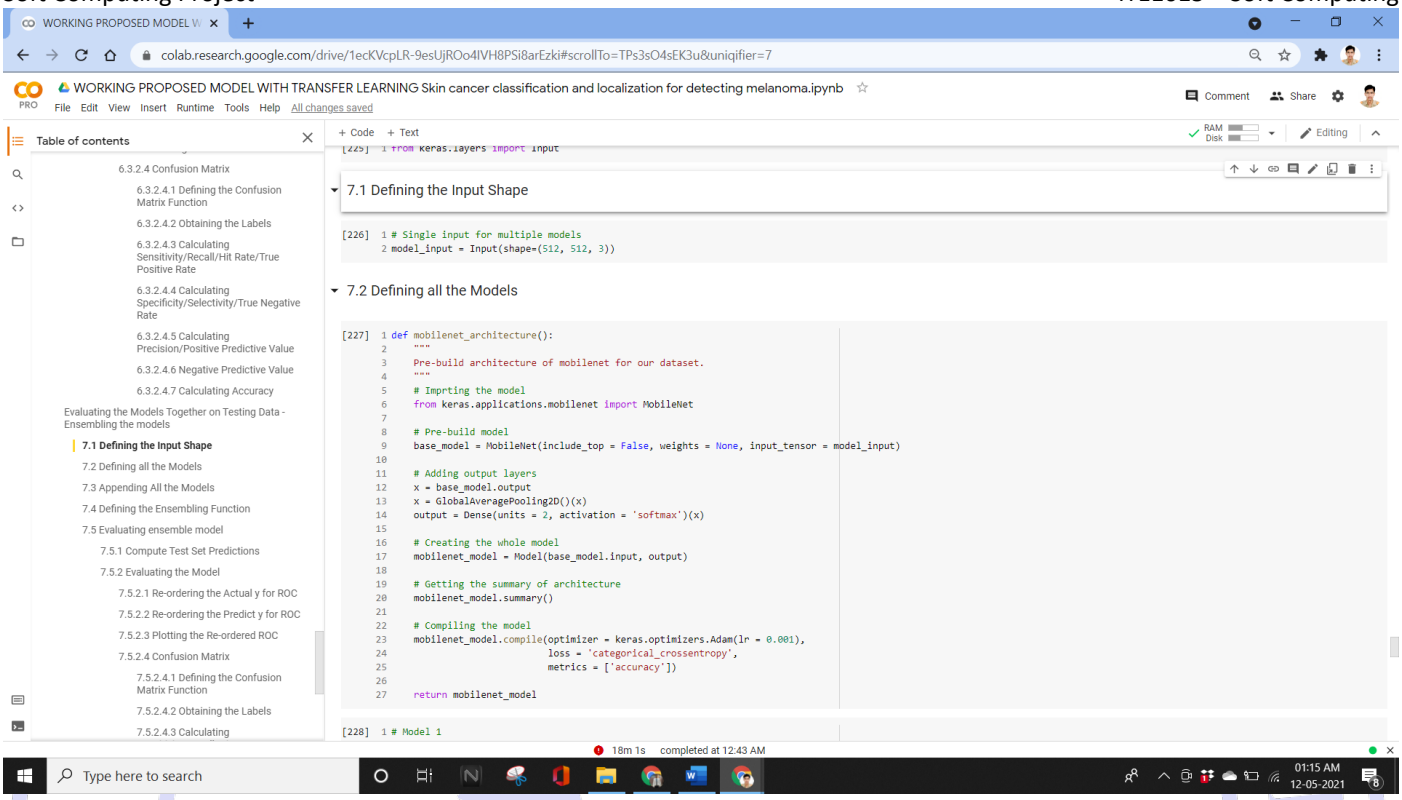

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016066.jpg
Prediction... Melanoma : 0.8119222 | Other : 0.18807776
Real values 600...Melanoma : 1.0 | Other : 0.0
1 print("Accuracy = %2.2f%%" % (np.mean(correct_Xception_Test)*100))
Accuracy = 80.50%
6.3.2 Evaluating the Model
6.3.2.1 Re-ordering the Actual y for ROC
[214] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Xception_Test = []
3 for i in range(len(y_true_Xception_Test)):
4 y_true_2_Xception_Test.append(y_true_Xception_Test[i][0])
6.3.2.2 Re-ordering the Predict y for ROC
[215] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Xception_Test = []
3 for i in range(len(y_score_Xception_Test)):
4 y_score_2_Xception_Test.append(y_score_Xception_Test[i][0])
6.3.2.3 Plotting the Re-ordered ROC
[216] 1 plot_roc(y_true_2_Xception_Test, y_score_2_Xception_Test)
Receiver operating characteristic example
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

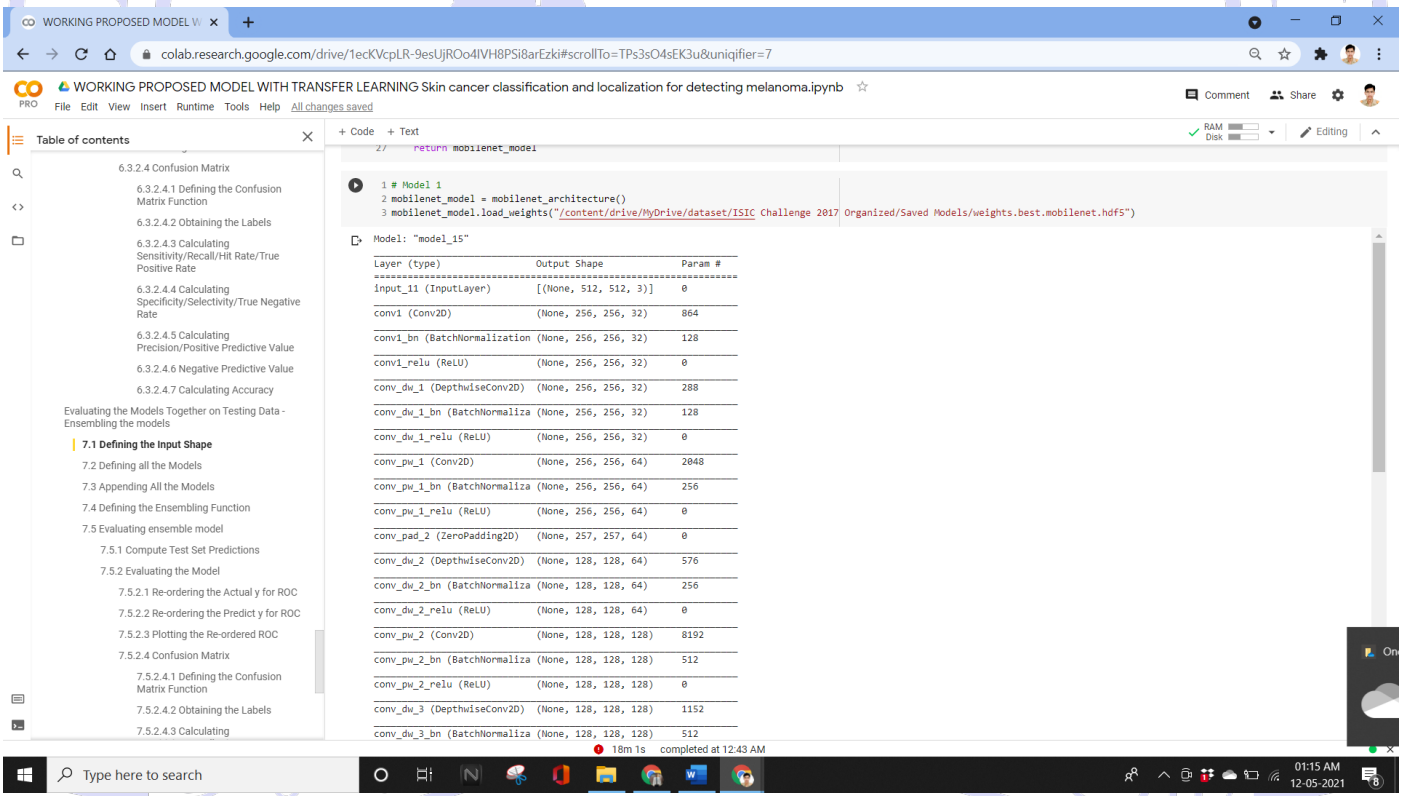

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
+ Code + Text
RAM Disk
Editing
Receiver operating characteristic example
True Positive Rate
False Positive Rate
ROC curve (area = 0.50)
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
[217] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

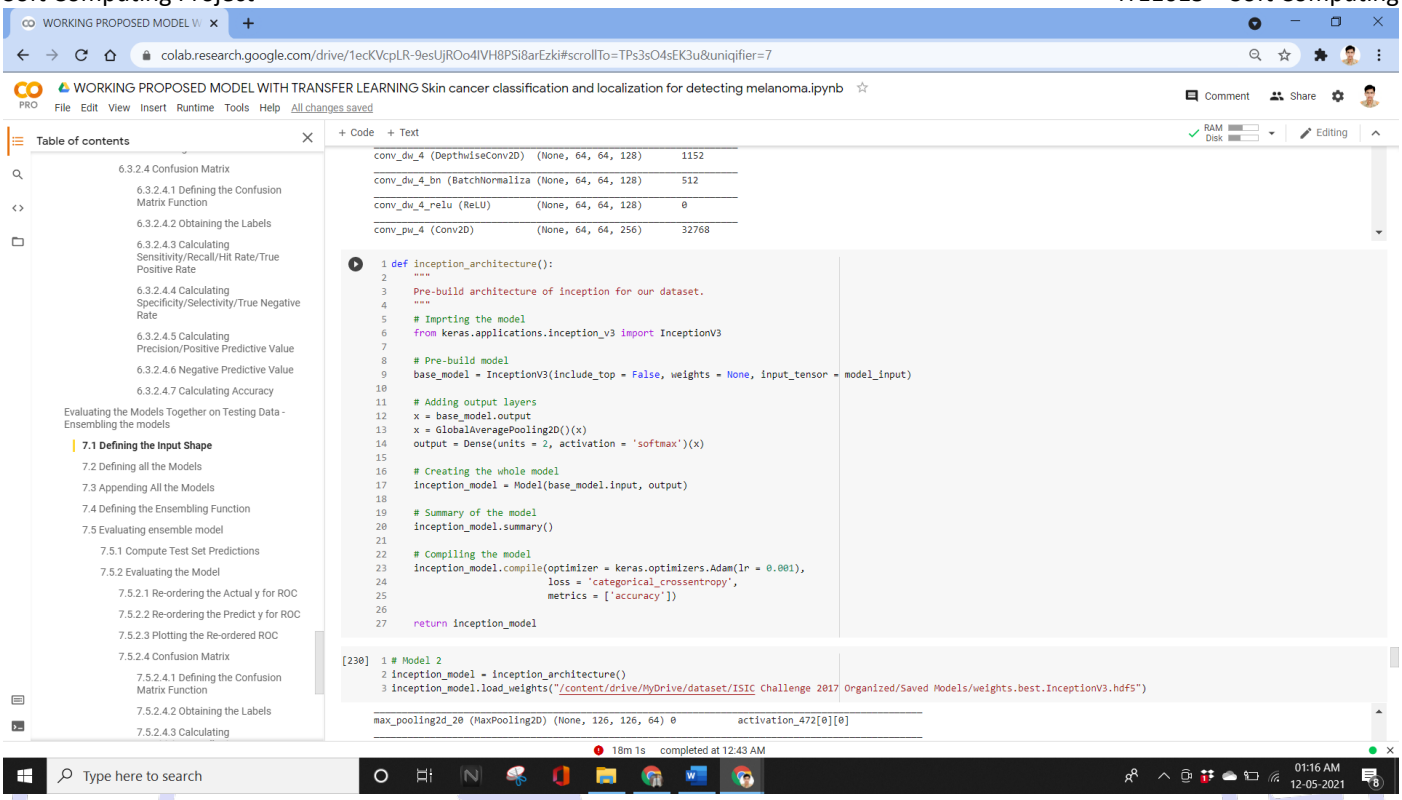

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM Disk
Editing
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[218] 1 TRUE_POSITIVE_Xception_Test, FALSE_POSITIVE_Xception_Test, TRUE_NEGATIVE_Xception_Test, FALSE_NEGATIVE_Xception_Test = positive_negative_measurement(y_true_2_Xception_Test, y_sc
2 postives_negatives_Xception_Test = [[TRUE_POSITIVE_Xception_Test, FALSE_POSITIVE_Xception_Test],
3 [FALSE_NEGATIVE_Xception_Test, TRUE_NEGATIVE_Xception_Test]]
6.3.2.4.2 Obtaining the Labels
[219] 1 import seaborn as sns
2 sns.set()
3 labels_Xception_Test = np.array([['True positive: ' + str(TRUE_POSITIVE_Xception_Test),
4 'False positive: ' + str(FALSE_POSITIVE_Xception_Test)],
5 ['False negative: ' + str(FALSE_NEGATIVE_Xception_Test),
6 'True negative: ' + str(TRUE_POSITIVE_Xception_Test)]])
7 plt.figure(figsize = (13, 10))
8 sns.heatmap(postives_negatives_Xception_Test, annot = labels_Xception_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f4672603e10>
True positive: 483
False positive: 117
400
300
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

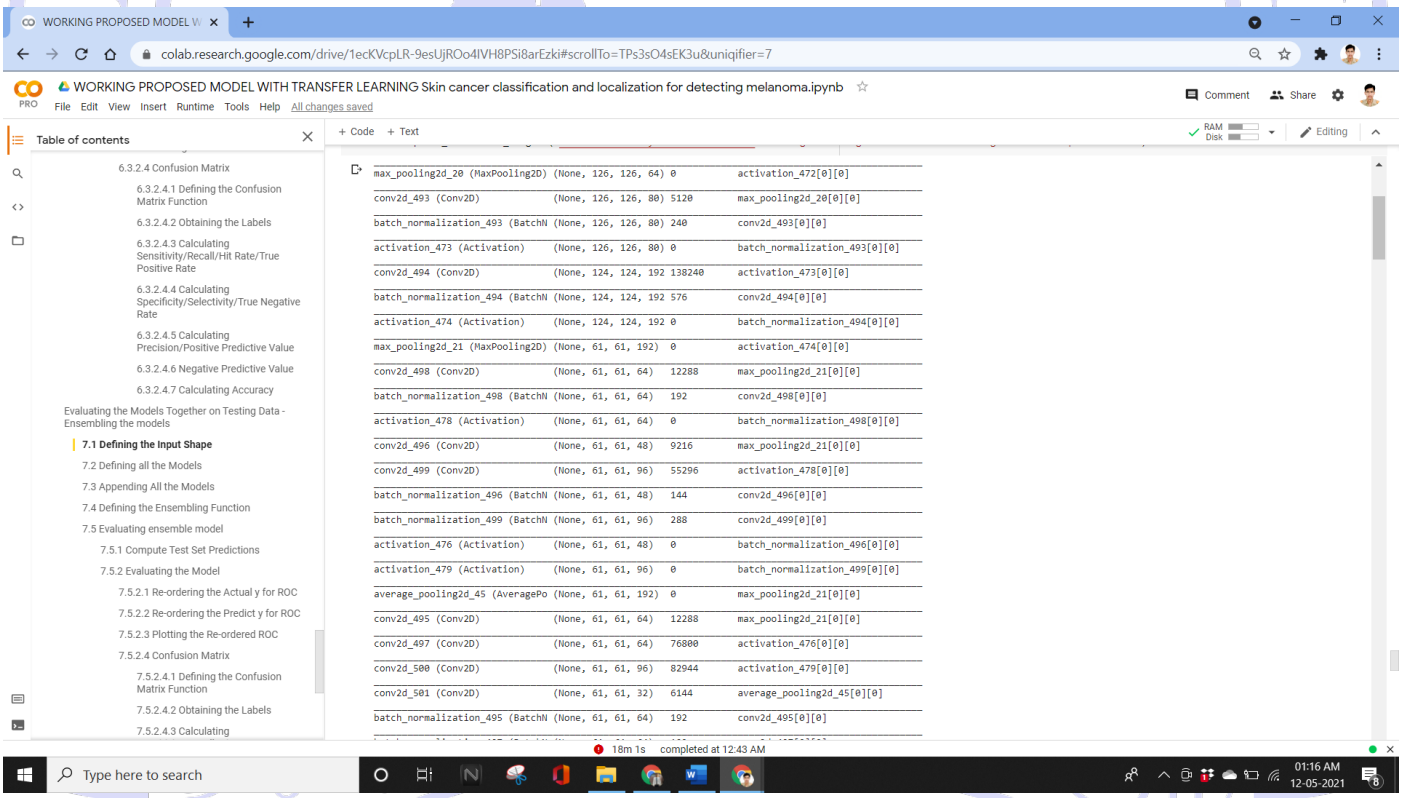

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
False negative: 0
True negative: 483
300
200
100
0
1
0
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[220] 1 # Sensitivity | Recall | hit rate | true positive rate (TPR)
2 sensitivity_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
3 print("Sensitivity: ", sensitivity_Xception_Test)
Sensitivity: 1.0
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[221] 1 # Specificity | selectivity | true negative rate (TNR)
2 try:
3 specifity_Xception_Test = TRUE_NEGATIVE_Xception_Test / (TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
4 print("Specifity: ", specifity_Xception_Test)
5 except:
6 print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

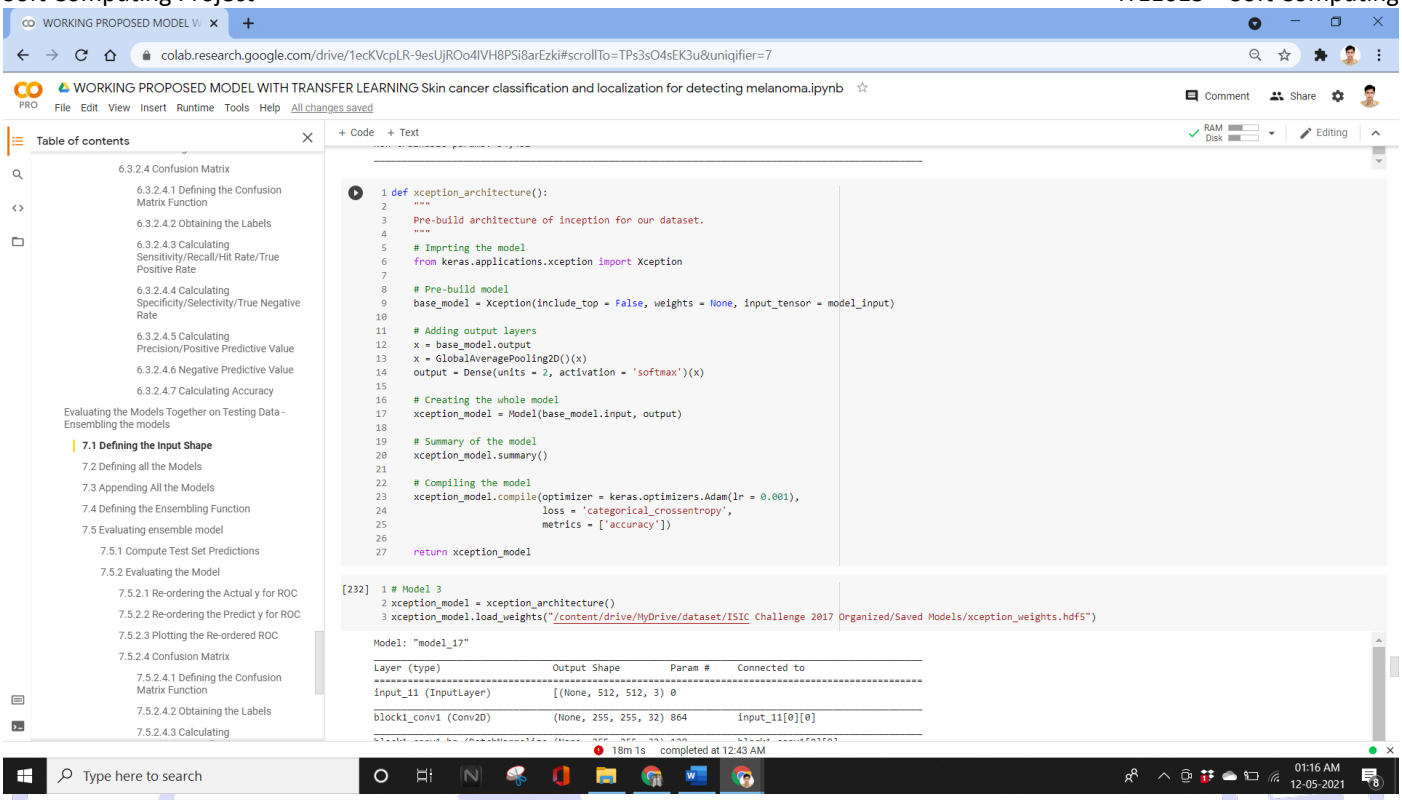

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=PLshKEuyCUtF&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.1.5 Confusion Matrix
6.1.5.1 Defining the Confusion Matrix Function
6.1.5.2 Obtaining Labels
6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate
6.1.5.5 Calculating Precision/Positive Predictive Value
6.1.5.6 Calculating Negative Predictive Value
6.1.5.7 Calculating Accuracy
6.2 Inception Architecture
6.2.1 Compute Test Set Predictions
6.2.2 Evaluating the Model
6.2.2.1 Re-ordering the Actual y for ROC
6.2.2.2 Re-ordering the Predicte y for ROC
6.2.2.3 Plotting the Re-ordered ROC
6.2.2.4 Confusion Matrix
6.2.2.4.1 Defining the Confusion Matrix Function
6.2.2.4.2 Obtaining Labels
6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.2.2.4.5 Calculating Precision/Positive Predictive Value
6.2.2.4.6 Negative Predictive Value
6.2.2.4.7 Calculating Accuracy
No Specificity due to NO NEGATIVE results.
6.3.2.4.5 Calculating Precision/Positive Predictive Value
[222] 1 # Precision | positive predictive value (PPV)
2 predcision_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test + FALSE_POSITIVE_Xception_Test)
3 print("Precision: ", predcision_Xception_Test)
Precision: 0.805
6.3.2.4.6 Negative Predictive Value
[223] 1 # Negative predictive value (NPV)
2 try:
3 npv_Xception_Test = TRUE_NEGATIVE_Xception_Test / (TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
4 print("Negative predictive value: ", npv_Xception_Test)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
6.3.2.4.7 Calculating Accuracy
[224] 1 # Accuracy
2 accuracy_Xception_Test = (TRUE_POSITIVE_Xception_Test + TRUE_NEGATIVE_Xception_Test) / (TRUE_POSITIVE_Xception_Test + FALSE_POSITIVE_Xception_Test + TRUE_NEGATIVE_Xception_Test
3 print("Accuracy: ", accuracy_Xception_Test)
Accuracy: 0.805
7. Evaluating the Models Together on Testing Data - Ensembling the models
[225] 1 from keras.layers import Input
7.1 Defining the Input Shape
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

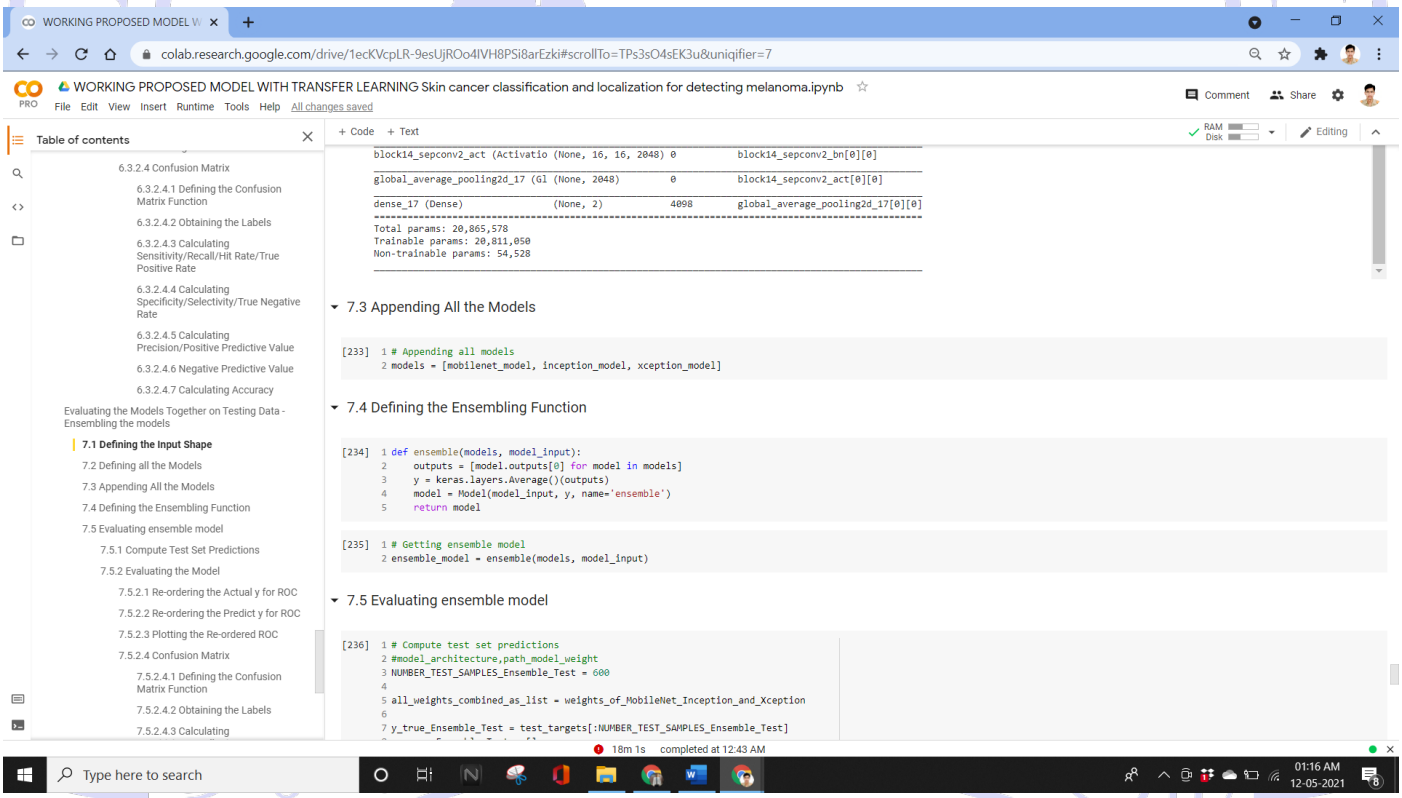

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
[225] 1 from keras.layers import Input
7.1 Defining the Input Shape
[226] 1 # Single input for multiple models
2 model_input = Input(shape=(512, 512, 3))
7.2 Defining all the Models
[227] 1 def mobilenet_architecture():
2 """
3 Pre-build architecture of mobilenet for our dataset.
4 """
5 # Imprting the model
6 from keras.applications.mobilenet import MobileNet
7
8 # Pre-build model
9 base_model = MobileNet(include_top = False, weights = None, input_tensor = model_input)
10
11 # Adding output layers
12 x = base_model.output
13 x = GlobalAveragePooling2D()(x)
14 output = Dense(units = 2, activation = 'softmax')(x)
15
16 # Creating the whole model
17 mobilenet_model = Model(base_model.input, output)
18
19 # Getting the summary of architecture
20 mobilenet_model.summary()
21
22 # Compiling the model
23 mobilenet_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
24 loss = 'categorical_crossentropy',
25 metrics = ['accuracy'])
26
27 return mobilenet_model
[228] 1 # Model 1
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

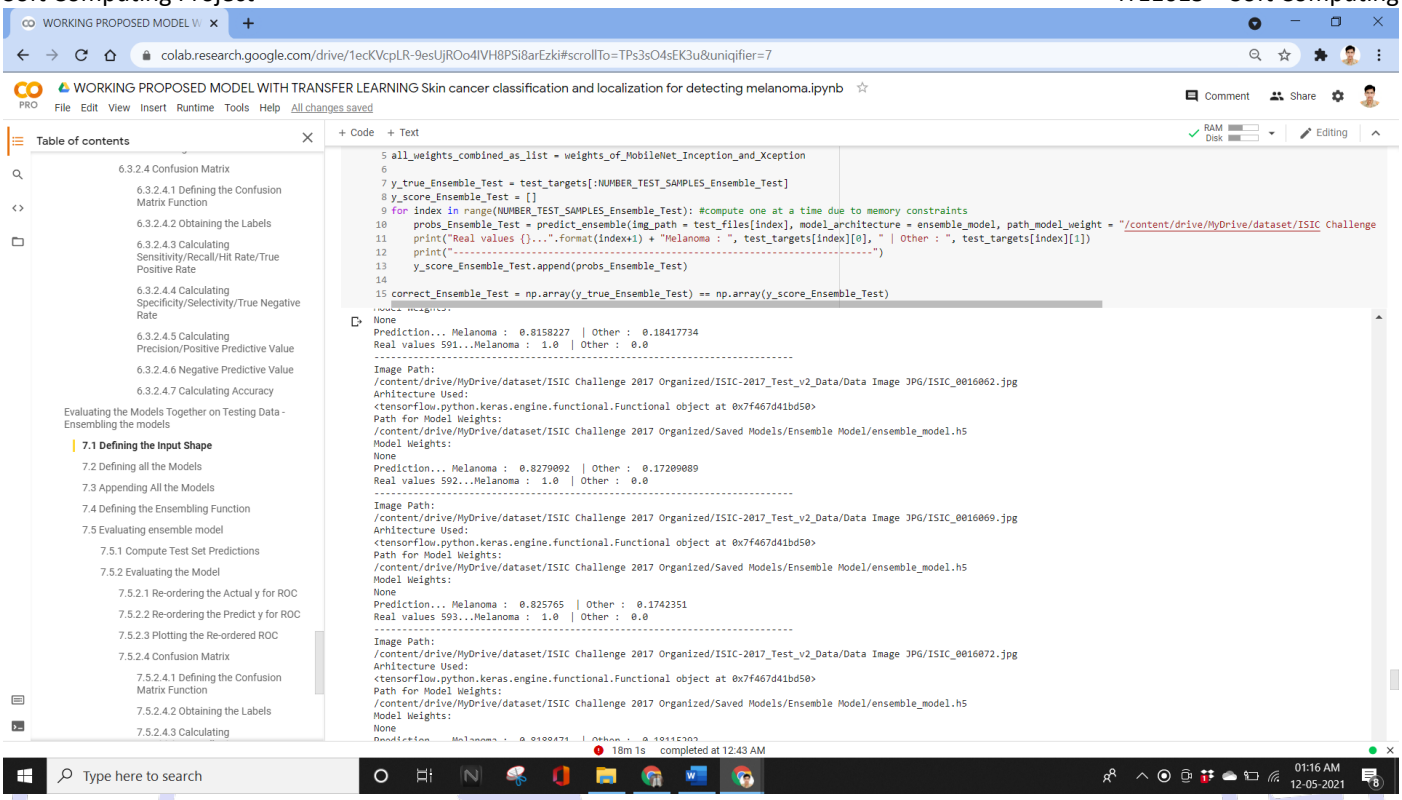

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
return mobilenet_model
1 # Model 1
2 mobilenet_model = mobilenet_architecture()
3 mobilenet_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5")
Model: "model_15"
Layer (type) Output Shape Param #
input_11 (InputLayer) [(None, 512, 512, 3)] 0
conv1 (Conv2D) (None, 256, 256, 32) 864
conv1_bn (BatchNormalization (None, 256, 256, 32) 128
conv1_relu (ReLU) (None, 256, 256, 32) 0
conv_dw_1 (DepthwiseConv2D) (None, 256, 256, 32) 288
conv_dw_1_bn (BatchNormaliza (None, 256, 256, 32) 128
conv_dw_1_relu (ReLU) (None, 256, 256, 32) 0
conv_pw_1 (Conv2D) (None, 256, 256, 64) 2048
conv_pw_1_bn (BatchNormaliza (None, 256, 256, 64) 256
conv_pw_1_relu (ReLU) (None, 256, 256, 64) 0
conv_pad_2 (ZeroPadding2D) (None, 257, 257, 64) 0
conv_dw_2 (DepthwiseConv2D) (None, 128, 128, 64) 576
conv_dw_2_bn (BatchNormaliza (None, 128, 128, 64) 256
conv_dw_2_relu (ReLU) (None, 128, 128, 64) 0
conv_pw_2 (Conv2D) (None, 128, 128, 128) 8192
conv_pw_2_bn (BatchNormaliza (None, 128, 128, 128) 512
conv_pw_2_relu (ReLU) (None, 128, 128, 128) 0
conv_dw_3 (DepthwiseConv2D) (None, 128, 128, 128) 1152
conv_dw_3_bn (BatchNormaliza (None, 128, 128, 128) 512
18m 1s completed at 12:43 AM
Type here to search
01:15 AM
12-05-2021

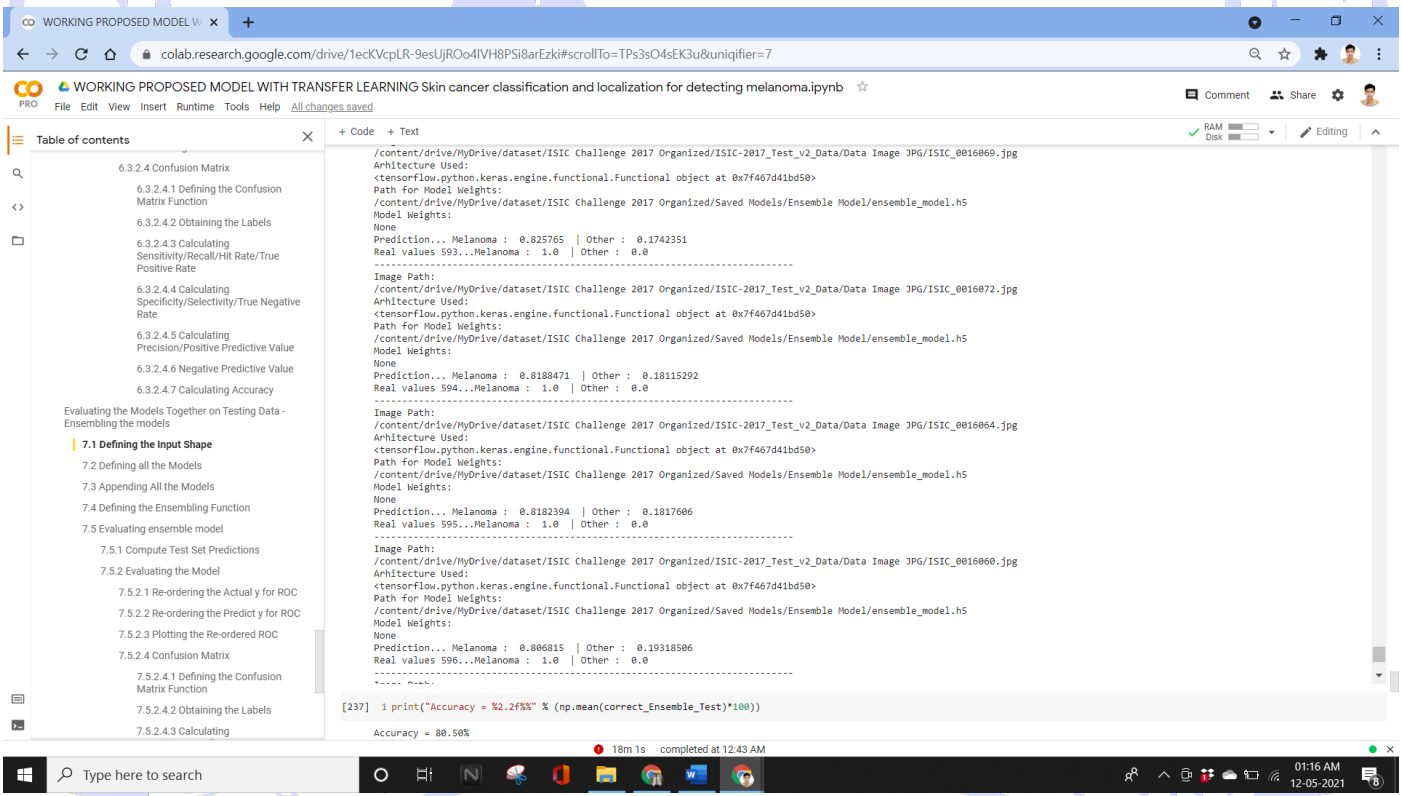

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
conv_dw_4 (DepthwiseConv2D) (None, 64, 64, 128) 1152
conv_dw_4_bn (BatchNormaliza (None, 64, 64, 128) 512
conv_dw_4_relu (ReLU) (None, 64, 64, 128) 0
conv_pw_4 (Conv2D) (None, 64, 64, 256) 32768
def inception_architecture():
"""
Pre-build architecture of inception for our dataset.
"""
# Imprting the model
from keras.applications.inception_v3 import InceptionV3
# Pre-build model
base_model = InceptionV3(include_top = False, weights = None, input_tensor = model_input)
# Adding output layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
output = Dense(units = 2, activation = 'softmax')(x)
# Creating the whole model
inception_model = Model(base_model.input, output)
# Summary of the model
inception_model.summary()
# Compiling the model
inception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
loss = 'categorical_crossentropy',
metrics = ['accuracy'])
return inception_model
[230]
# Model 2
inception_model = inception_architecture()
inception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5")
max_pooling2d_20 (MaxPooling2D) (None, 126, 126, 64) 0 activation_472[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

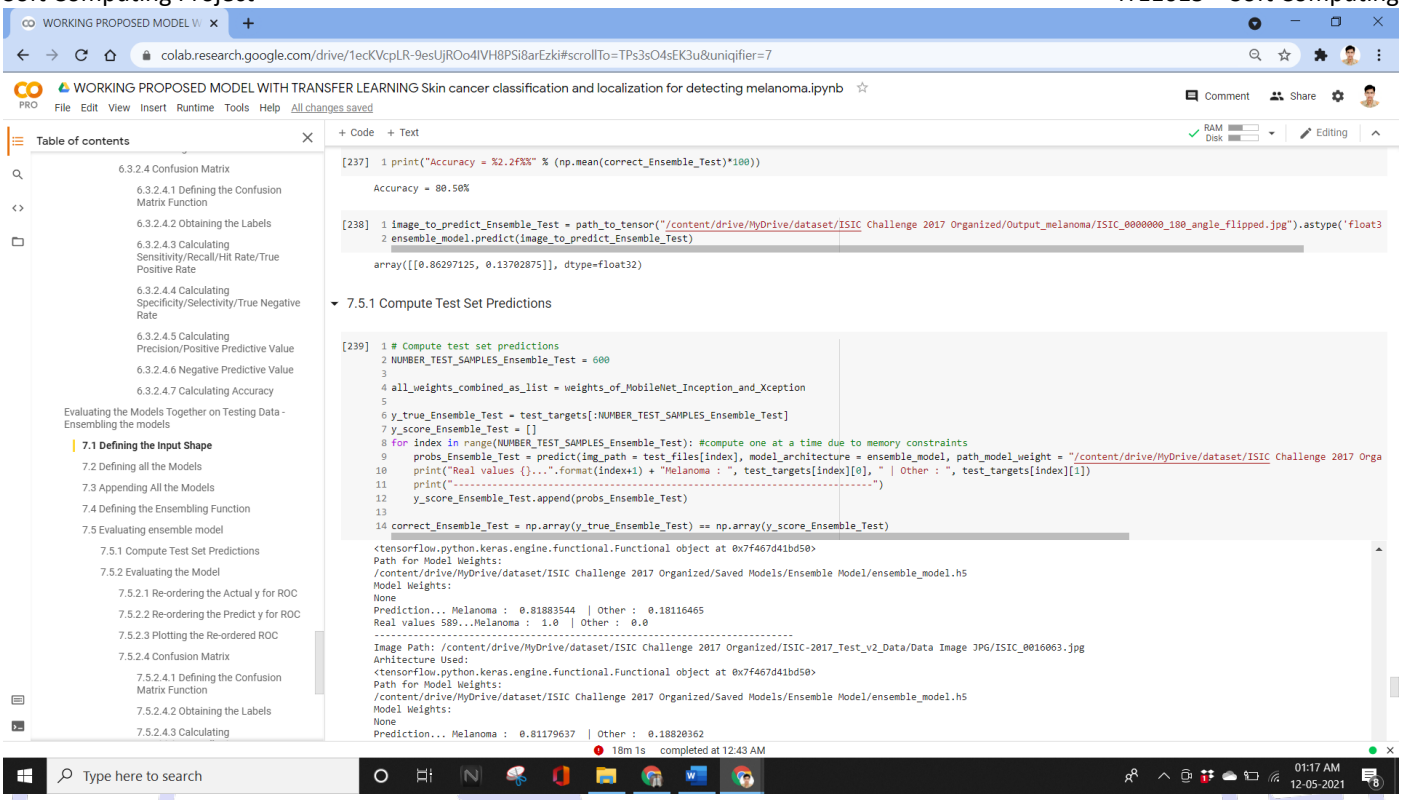

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
max_pooling2d_20 (MaxPooling2D) (None, 126, 126, 64) 0 activation_472[0][0]
conv2d_493 (Conv2D) (None, 126, 126, 80) 5120 max_pooling2d_20[0][0]
batch_normalization_493 (BatchN (None, 126, 126, 80) 240 conv2d_493[0][0]
activation_473 (Activation) (None, 126, 126, 80) 0 batch_normalization_493[0][0]
conv2d_494 (Conv2D) (None, 124, 124, 192 138240 activation_473[0][0]
batch_normalization_494 (BatchN (None, 124, 124, 192 576 conv2d_494[0][0]
activation_474 (Activation) (None, 124, 124, 192 0 batch_normalization_494[0][0]
max_pooling2d_21 (MaxPooling2D) (None, 61, 61, 192) 0 activation_474[0][0]
conv2d_498 (Conv2D) (None, 61, 61, 64) 12288 max_pooling2d_21[0][0]
batch_normalization_498 (BatchN (None, 61, 61, 64) 192 conv2d_498[0][0]
activation_478 (Activation) (None, 61, 61, 64) 0 batch_normalization_498[0][0]
conv2d_496 (Conv2D) (None, 61, 61, 48) 9216 max_pooling2d_21[0][0]
conv2d_499 (Conv2D) (None, 61, 61, 96) 55296 activation_478[0][0]
batch_normalization_496 (BatchN (None, 61, 61, 48) 144 conv2d_496[0][0]
batch_normalization_499 (BatchN (None, 61, 61, 96) 288 conv2d_499[0][0]
activation_476 (Activation) (None, 61, 61, 48) 0 batch_normalization_496[0][0]
activation_479 (Activation) (None, 61, 61, 96) 0 batch_normalization_499[0][0]
average_pooling2d_45 (AveragePo (None, 61, 61, 192) 0 max_pooling2d_21[0][0]
conv2d_495 (Conv2D) (None, 61, 61, 64) 12288 max_pooling2d_21[0][0]
conv2d_497 (Conv2D) (None, 61, 61, 64) 76800 activation_476[0][0]
conv2d_500 (Conv2D) (None, 61, 61, 96) 82944 activation_479[0][0]
conv2d_501 (Conv2D) (None, 61, 61, 32) 6144 average_pooling2d_45[0][0]
batch_normalization_495 (BatchN (None, 61, 61, 64) 192 conv2d_495[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

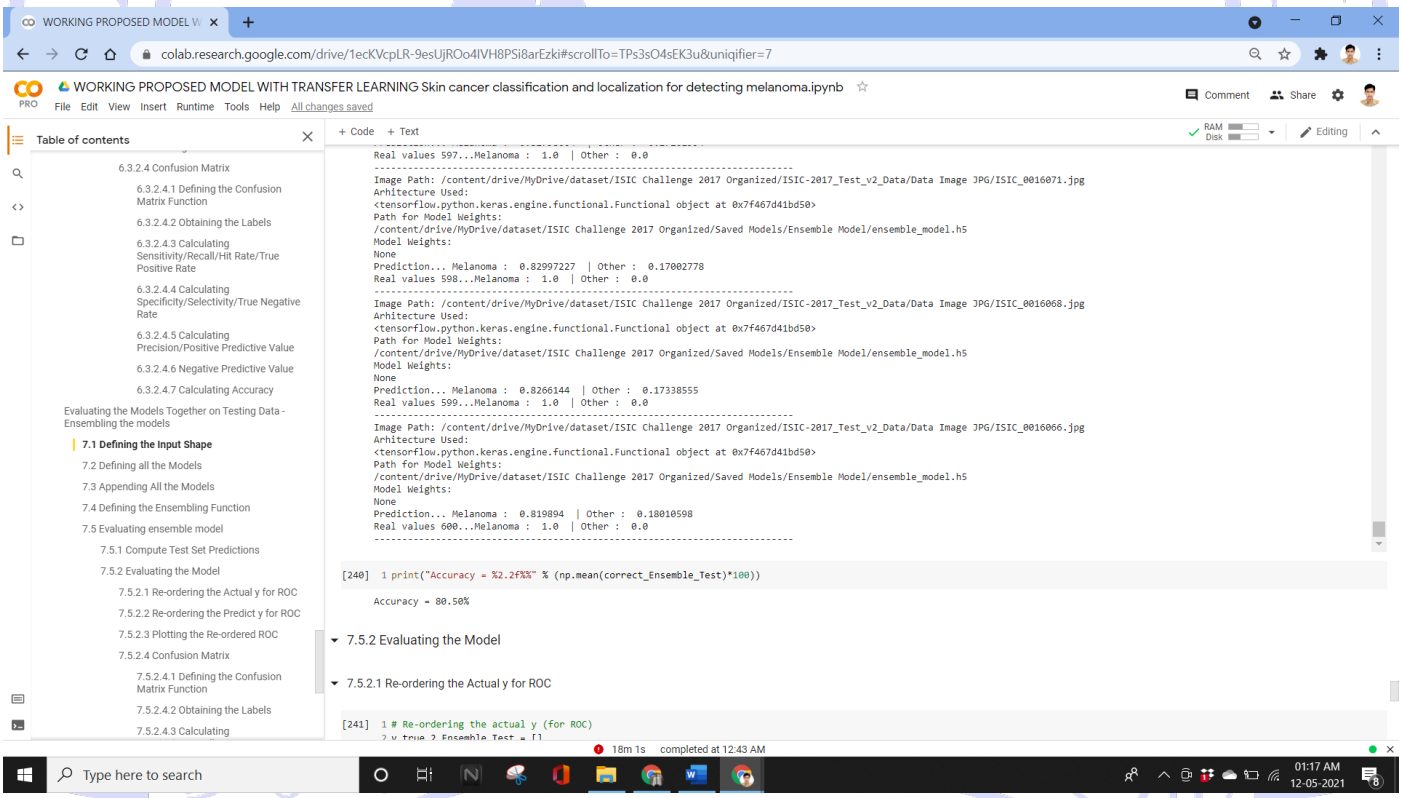

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
def xception_architecture():
"""
Pre-build architecture of inception for our dataset.
"""
# Imprting the model
from keras.applications.xception import Xception
# Pre-build model
base_model = Xception(include_top = False, weights = None, input_tensor = model_input)
# Adding output layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
output = Dense(units = 2, activation = 'softmax')(x)
# Creating the whole model
xception_model = Model(base_model.input, output)
# Summary of the model
xception_model.summary()
# Compiling the model
xception_model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
loss = 'categorical_crossentropy',
metrics = ['accuracy'])
return xception_model
[232] # Model 3
xception_model = xception_architecture()
xception_model.load_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5")
Model: "model_17"
Layer (type) Output Shape Param # Connected to
input_11 (InputLayer) [(None, 512, 512, 3) 0
block1_conv1 (Conv2D) (None, 255, 255, 32) 864 input_11[0][0]
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

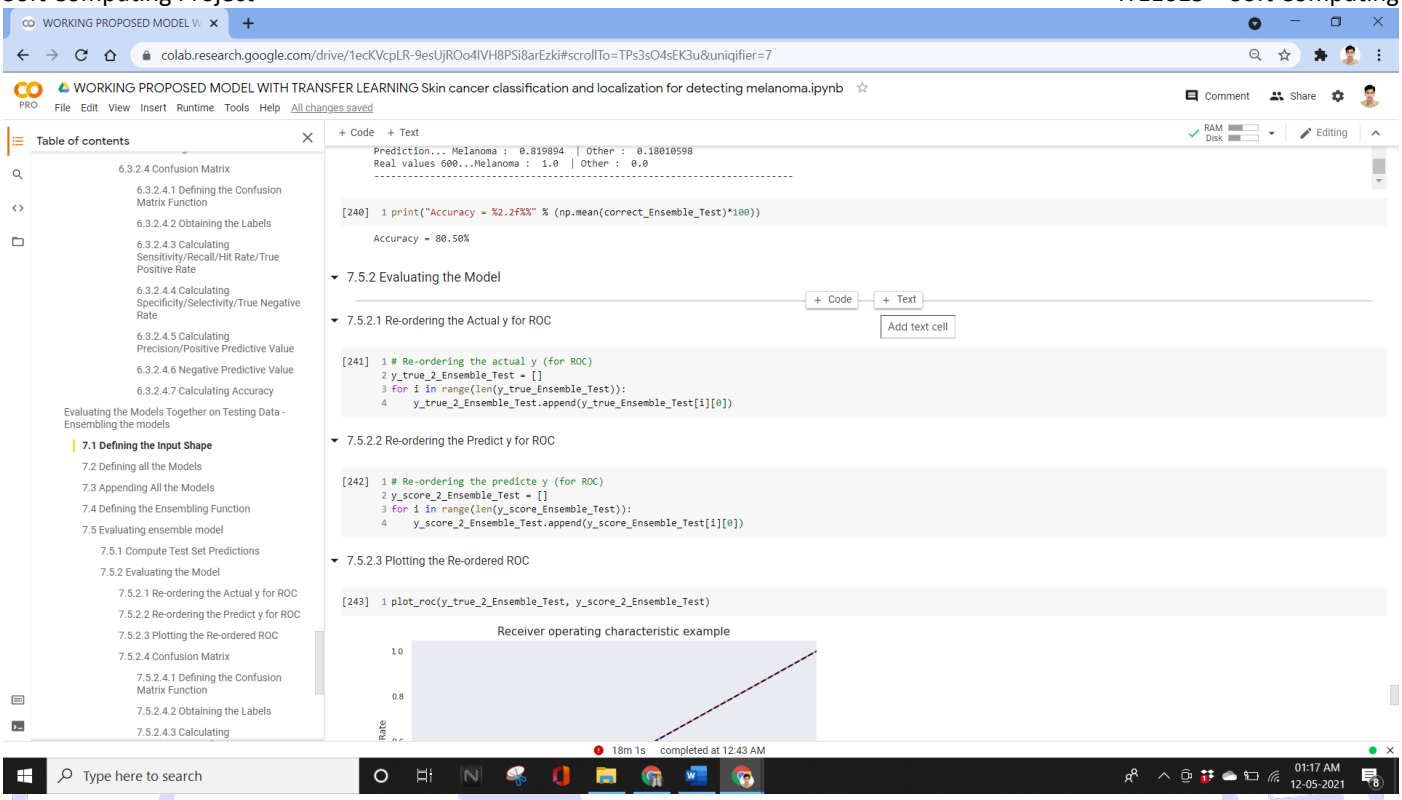

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
block14_sepconv2_act (Activatio (None, 16, 16, 2048) 0 block14_sepconv2_bn[0][0]
global_average_pooling2d_17 (Gl (None, 2048) 0 block14_sepconv2_act[0][0]
dense_17 (Dense) (None, 2) 4098 global_average_pooling2d_17[0][0]
Total params: 20,865,578
Trainable params: 20,811,050
Non-trainable params: 54,528
7.3 Appending All the Models
[233] # Appending all models
models = [mobilenet_model, inception_model, xception_model]
7.4 Defining the Ensembling Function
[234] def ensemble(models, model_input):
outputs = [model.outputs[0] for model in models]
y = keras.layers.Average()(outputs)
model = Model(model_input, y, name='ensemble')
return model
[235] # Getting ensemble model
ensemble_model = ensemble(models, model_input)
7.5 Evaluating ensemble model
[236] # Compute test set predictions
#model_architecture,path_model_weight
NUMBER_TEST_SAMPLES_Ensemble_Test = 600
all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception
y_true_Ensemble_Test = test_targets[:NUMBER_TEST_SAMPLES_Ensemble_Test]
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

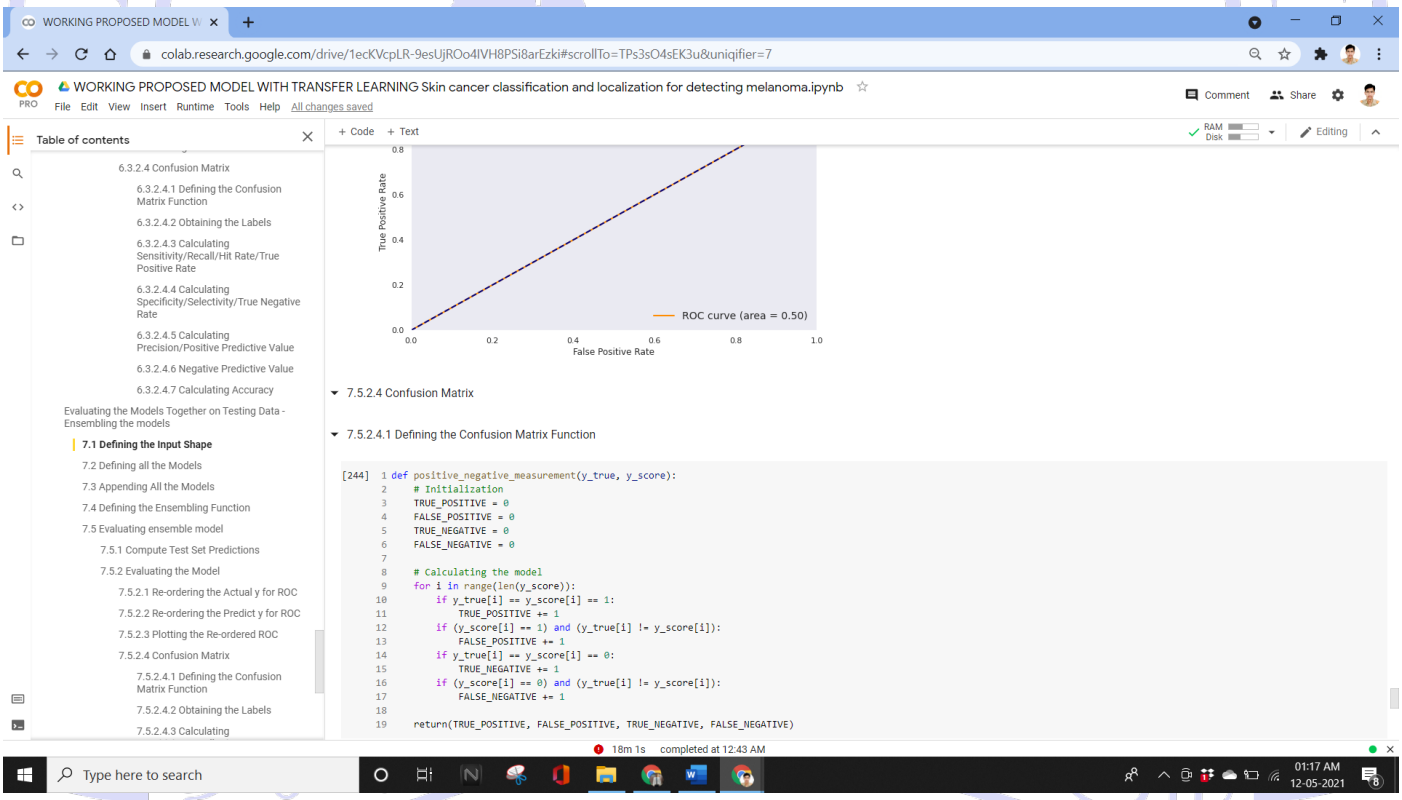

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM
Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception
y_true_Ensemble_Test = test_targets[:NUMBER_TEST_SAMPLES_Ensemble_Test]
y_score_Ensemble_Test = []
for index in range(NUMBER_TEST_SAMPLES_Ensemble_Test): #compute one at a time due to memory constraints
probs_Ensemble_Test = predict_ensemble(img_path = test_files[index], model_architecture = ensemble_model, path_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge
print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0], " | Other : ", test_targets[index][1])
print("----------")
y_score_Ensemble_Test.append(probs_Ensemble_Test)
correct_Ensemble_Test = np.array(y_true_Ensemble_Test) == np.array(y_score_Ensemble_Test)
None
Prediction... Melanoma : 0.8158227 | Other : 0.18417734
Real values 591...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016062.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8279092 | Other : 0.17209089
Real values 592...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016069.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.825765 | Other : 0.1742351
Real values 593...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016072.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

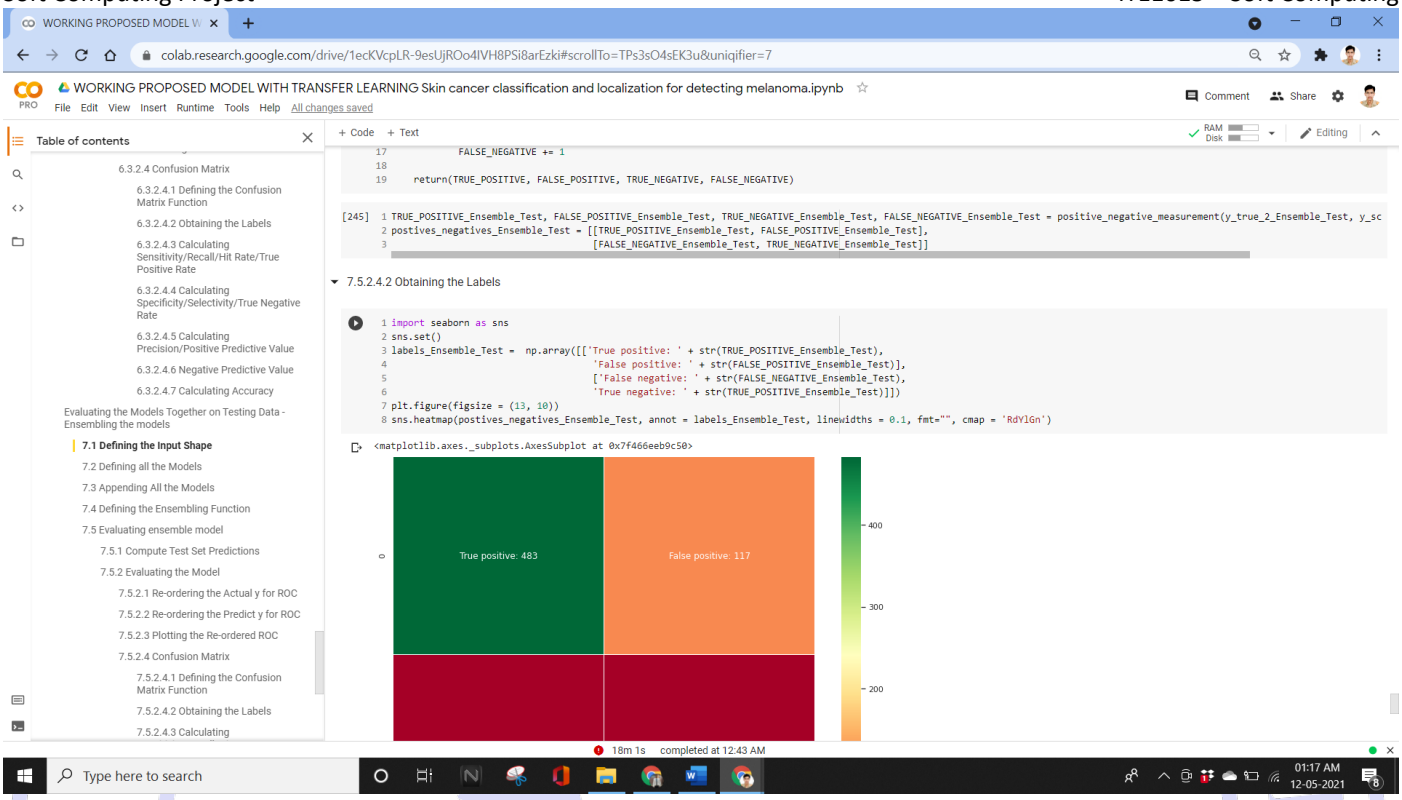

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
+ Code + Text
RAM
Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016069.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.825765 | Other : 0.1742351
Real values 593...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016072.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8188471 | Other : 0.18115292
Real values 594...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016064.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.8182394 | Other : 0.1817606
Real values 595...Melanoma : 1.0 | Other : 0.0
Image Path:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016060.jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.806815 | Other : 0.19318506
Real values 596...Melanoma : 1.0 | Other : 0.0
[237] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))
Accuracy = 80.50%
18m 1s completed at 12:43 AM
Type here to search
01:16 AM
12-05-2021

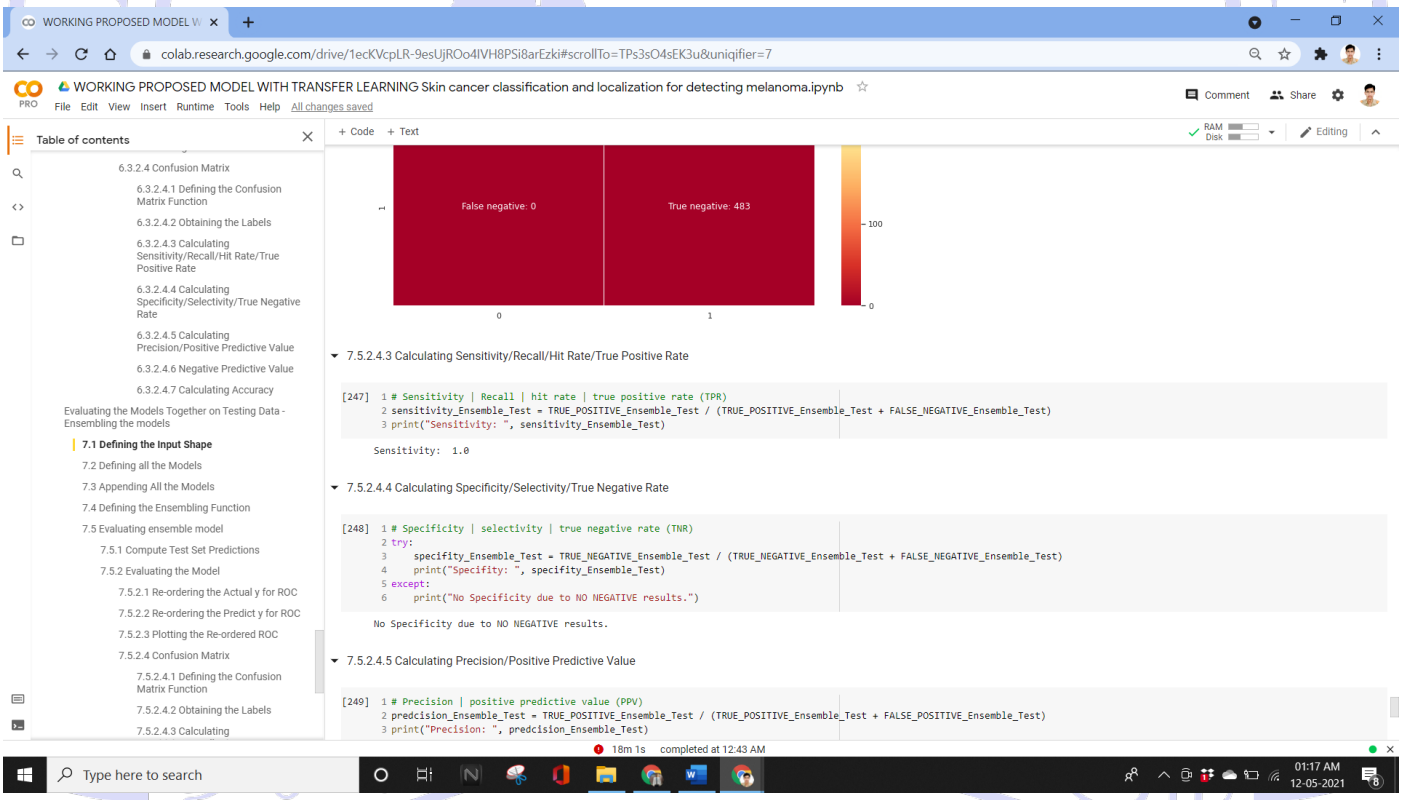

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
[237] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))
Accuracy = 80.50%
[238] 1 image_to_predict_Ensemble_Test = path_to_tensor("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Output_melanoma/ISIC_0000000_180_angle_flipped.jpg").astype('float3
2 ensemble_model.predict(image_to_predict_Ensemble_Test)
array([[0.86297125, 0.13702875]], dtype=float32)
7.5.1 Compute Test Set Predictions
[239] 1 # Compute test set predictions
2 NUMBER_TEST_SAMPLES_Ensemble_Test = 600
3
4 all_weights_combined_as_list = weights_of_MobileNet_Inception_and_Xception
5
6 y_true_Ensemble_Test = test_targets[:NUMBER_TEST_SAMPLES_Ensemble_Test]
7 y_score_Ensemble_Test = []
8 for index in range(NUMBER_TEST_SAMPLES_Ensemble_Test): #compute one at a time due to memory constraints
9 probs_Ensemble_Test = predict(img_path = test_files[index], model_architecture = ensemble_model, path_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Orga
10 print("Real values {}...".format(index+1) + "Melanoma : ", test_targets[index][0], " | Other : ", test_targets[index][1])
11 print("----------")
12 y_score_Ensemble_Test.append(probs_Ensemble_Test)
13
14 correct_Ensemble_Test = np.array(y_true_Ensemble_Test) == np.array(y_score_Ensemble_Test)
<tensorflow.python.keras.engine.functional.Functional object at 0x7f467d41bd50>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.h5
Model Weights:
None
Prediction... Melanoma : 0.81883544 | Other : 0.18116465
Real values 589...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016063.jpg
Ahritecture Used:
Prediction... Melanoma : 0.81179637 | Other : 0.18820362
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021

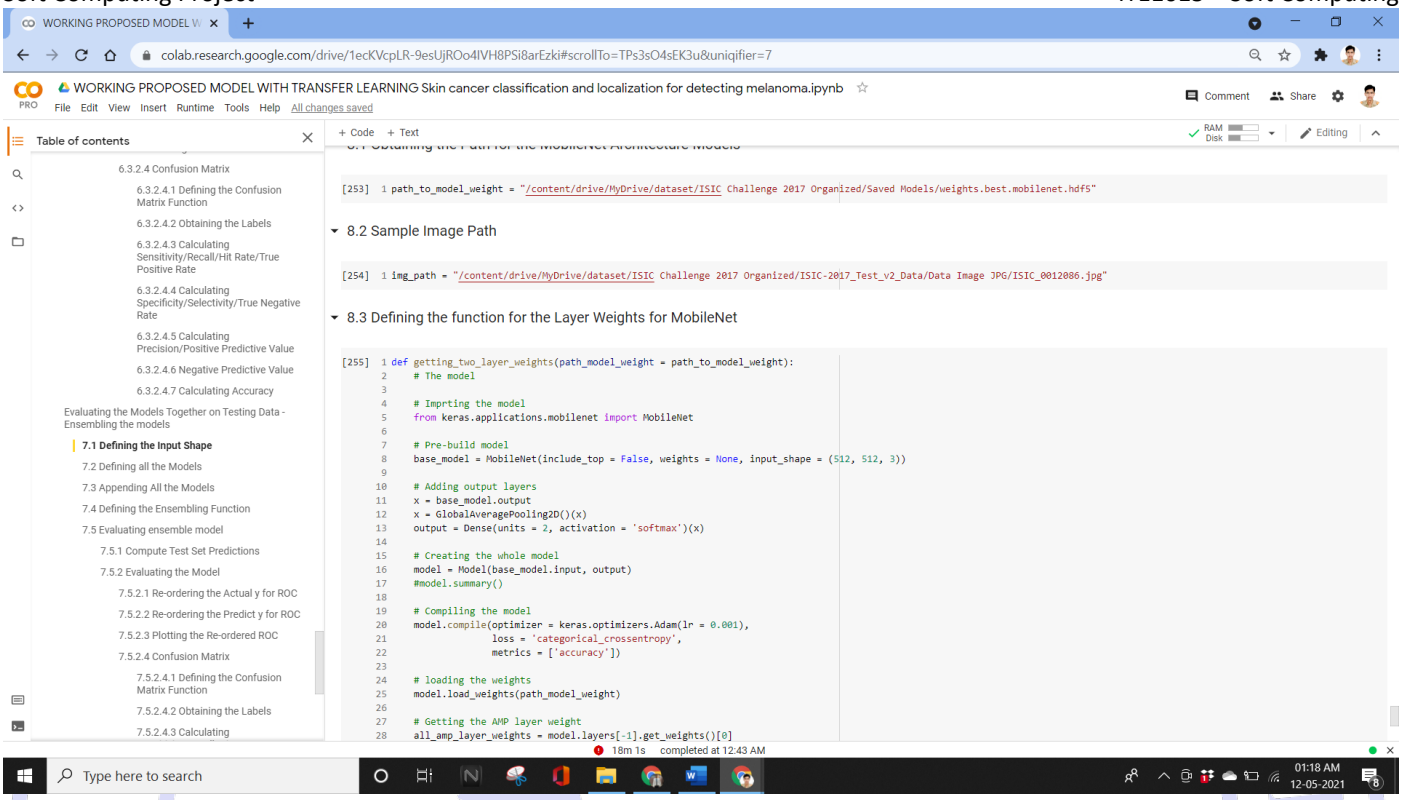

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
Real values 597...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016071.jpg
Ahritecture Used:
Prediction... Melanoma : 0.82997227 | Other : 0.17002778
Real values 598...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016068.jpg
Ahritecture Used:
Prediction... Melanoma : 0.8266144 | Other : 0.17338555
Real values 599...Melanoma : 1.0 | Other : 0.0
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0016066.jpg
Ahritecture Used:
Prediction... Melanoma : 0.819894 | Other : 0.18010598
Real values 600...Melanoma : 1.0 | Other : 0.0
[240] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))
Accuracy = 80.50%
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
[241] 1 # Re-ordering the actual y (for ROC)
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021

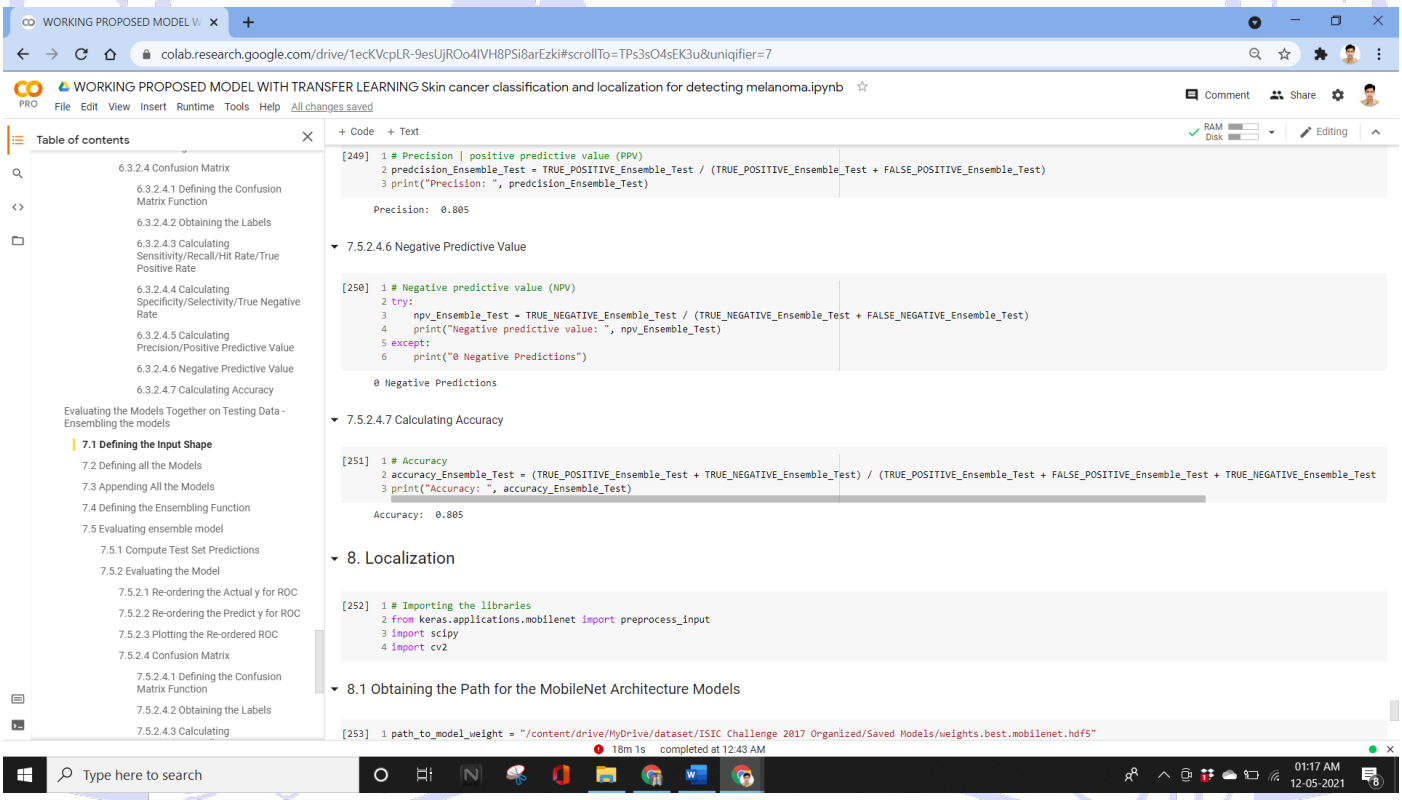

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
Prediction... Melanoma : 0.819894 | Other : 0.18010598
Real values 600...Melanoma : 1.0 | Other : 0.0
[240] 1 print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))
Accuracy = 80.50%
7.5.2 Evaluating the Model
+ Code + Text
Add text cell
7.5.2.1 Re-ordering the Actual y for ROC
[241] 1 # Re-ordering the actual y (for ROC)
2 y_true_2_Ensemble_Test = []
3 for i in range(len(y_true_Ensemble_Test)):
4 y_true_2_Ensemble_Test.append(y_true_Ensemble_Test[i][0])
7.5.2.2 Re-ordering the Predict y for ROC
[242] 1 # Re-ordering the predicte y (for ROC)
2 y_score_2_Ensemble_Test = []
3 for i in range(len(y_score_Ensemble_Test)):
4 y_score_2_Ensemble_Test.append(y_score_Ensemble_Test[i][0])
7.5.2.3 Plotting the Re-ordered ROC
[243] 1 plot_roc(y_true_2_Ensemble_Test, y_score_2_Ensemble_Test)
Receiver operating characteristic example
Rate
1.0
0.8
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021

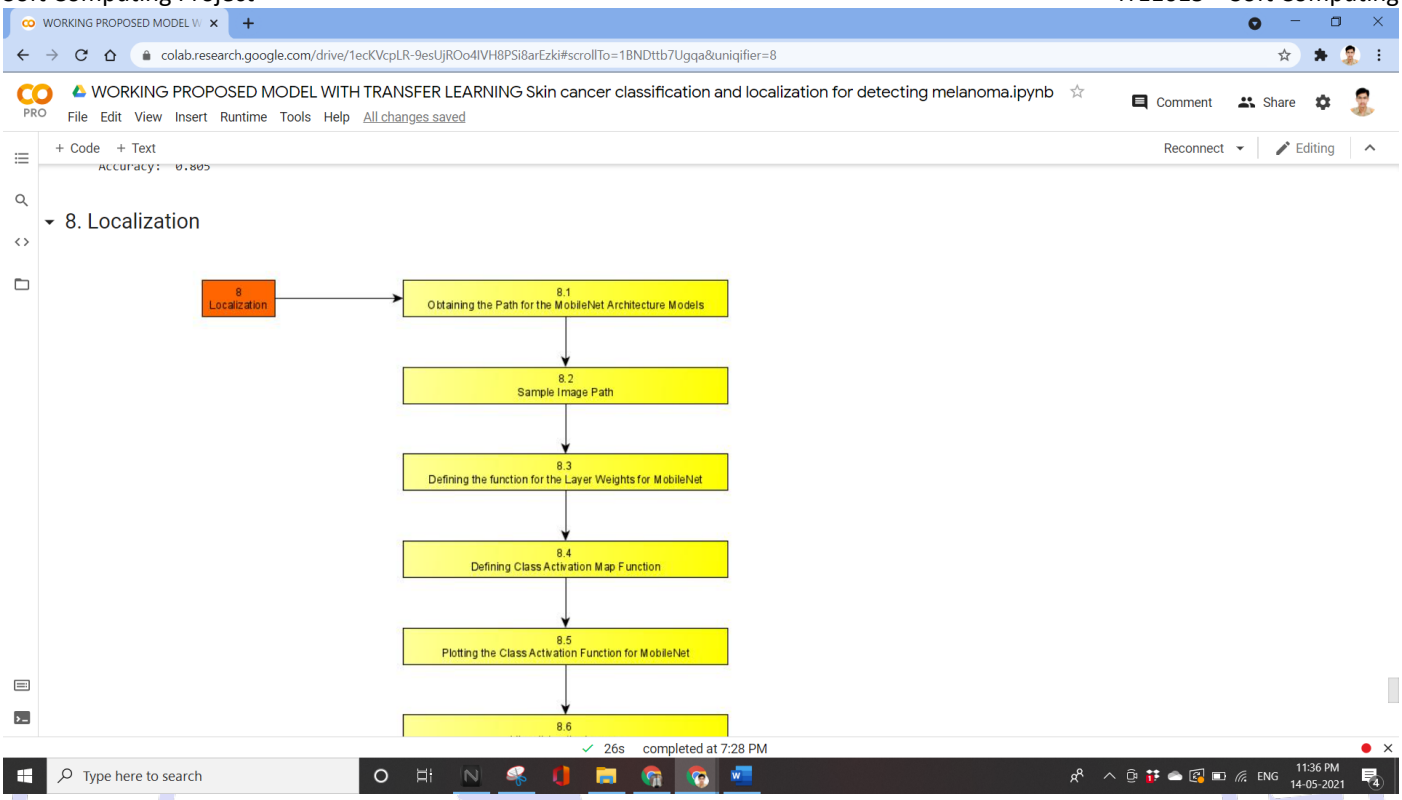

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
True Positive Rate
False Positive Rate
ROC curve (area = 0.50)
0.0
0.2
0.4
0.6
0.8
1.0
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
[244] 1 def positive_negative_measurement(y_true, y_score):
2 # Initialization
3 TRUE_POSITIVE = 0
4 FALSE_POSITIVE = 0
5 TRUE_NEGATIVE = 0
6 FALSE_NEGATIVE = 0
7
8 # Calculating the model
9 for i in range(len(y_score)):
10 if y_true[i] == y_score[i] == 1:
11 TRUE_POSITIVE += 1
12 if (y_score[i] == 1) and (y_true[i] != y_score[i]):
13 FALSE_POSITIVE += 1
14 if y_true[i] == y_score[i] == 0:
15 TRUE_NEGATIVE += 1
16 if (y_score[i] == 0) and (y_true[i] != y_score[i]):
17 FALSE_NEGATIVE += 1
18
19 return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021

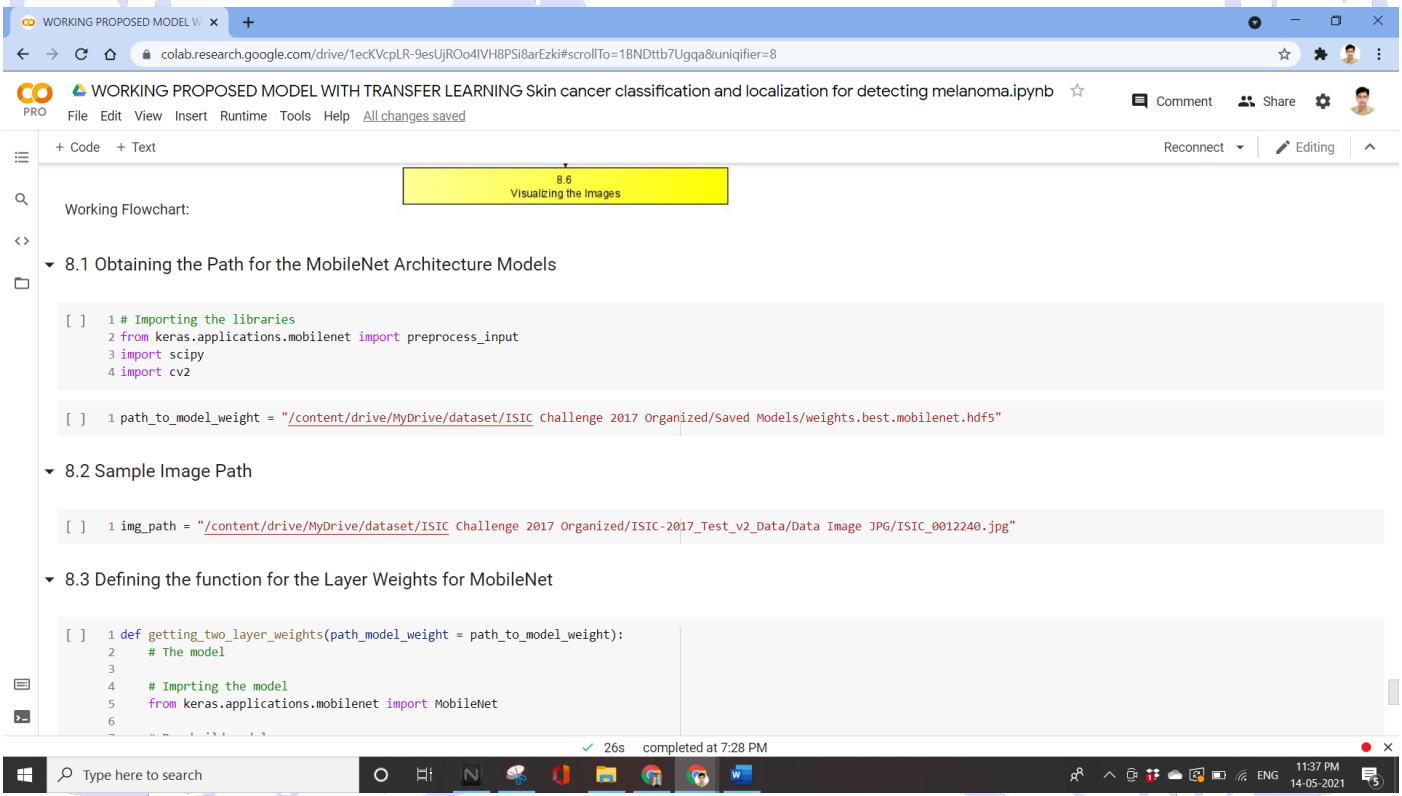

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
FALSE_NEGATIVE += 1
return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
[245] TRUE_POSITIVE_Ensemble_Test, FALSE_POSITIVE_Ensemble_Test, TRUE_NEGATIVE_Ensemble_Test, FALSE_NEGATIVE_Ensemble_Test = positive_negative_measurement(y_true_2_Ensemble_Test, y_sc
postives_negatives_Ensemble_Test = [[TRUE_POSITIVE_Ensemble_Test, FALSE_POSITIVE_Ensemble_Test],
[FALSE_NEGATIVE_Ensemble_Test, TRUE_NEGATIVE_Ensemble_Test]]
7.5.2.4.2 Obtaining the Labels
import seaborn as sns
sns.set()
labels_Ensemble_Test = np.array([['True positive: ' + str(TRUE_POSITIVE_Ensemble_Test),
'False positive: ' + str(FALSE_POSITIVE_Ensemble_Test)],
['False negative: ' + str(FALSE_NEGATIVE_Ensemble_Test),
'True negative: ' + str(TRUE_POSITIVE_Ensemble_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Ensemble_Test, annot = labels_Ensemble_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
<matplotlib.axes._subplots.AxesSubplot at 0x7f466eeb9c50>
True positive: 483
False positive: 117
400
300
200
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021

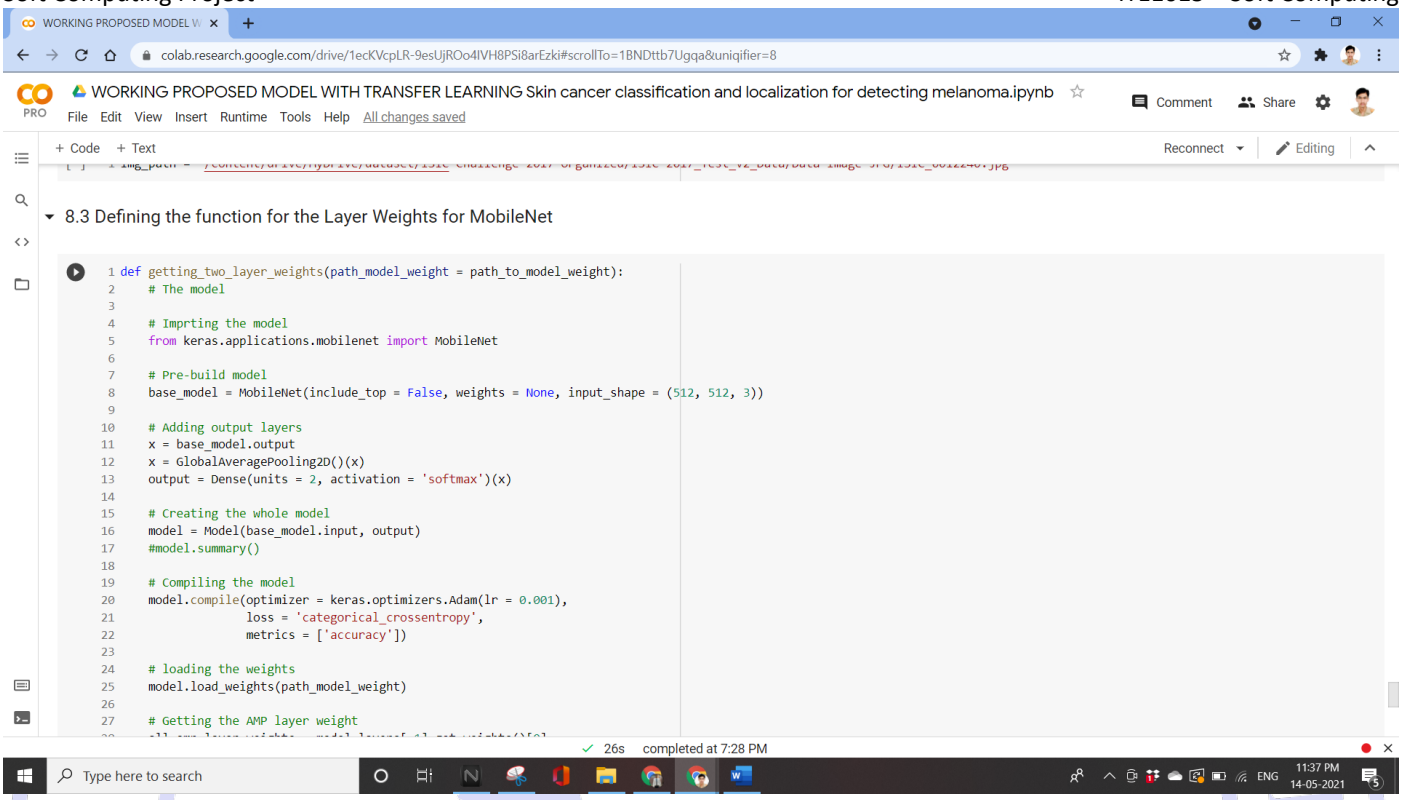

WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
False negative: 0
True negative: 483
100
0
7.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
[247] # Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
print("Sensitivity: ", sensitivity_Ensemble_Test)
Sensitivity: 1.0
7.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
[248] # Specificity | selectivity | true negative rate (TNR)
try:
specifity_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test / (TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
print("Specifity: ", specifity_Ensemble_Test)
except:
print("No Specificity due to NO NEGATIVE results.")
No Specificity due to NO NEGATIVE results.
7.5.2.4.5 Calculating Precision/Positive Predictive Value
[249] # Precision | positive predictive value (PPV)
predcision_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test + FALSE_POSITIVE_Ensemble_Test)
print("Precision: ", predcision_Ensemble_Test)
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
[253] 1 path_to_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
8.2 Sample Image Path
[254] 1 img_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012086.jpg"
8.3 Defining the function for the Layer Weights for MobileNet
[255] 1 def getting_two_layer_weights(path_model_weight = path_to_model_weight):
2 # The model
3
4 # Imprting the model
5 from keras.applications.mobilenet import MobileNet
6
7 # Pre-build model
8 base_model = MobileNet(include_top = False, weights = None, input_shape = (512, 512, 3))
9
10 # Adding output layers
11 x = base_model.output
12 x = GlobalAveragePooling2D()(x)
13 output = Dense(units = 2, activation = 'softmax')(x)
14
15 # Creating the whole model
16 model = Model(base_model.input, output)
17 #model.summary()
18
19 # Compiling the model
20 model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
21 loss = 'categorical_crossentropy',
22 metrics = ['accuracy'])
23
24 # loading the weights
25 model.load_weights(path_model_weight)
26
27 # Getting the AMP layer weight
28 all_amp_layer_weights = model.layers[-1].get_weights()[0]
18m 1s completed at 12:43 AM
Type here to search
01:18 AM
12-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=TPs3sO4sEK3u&uniqifier=7
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Table of contents
+ Code + Text
RAM Disk
Editing
6.3.2.4 Confusion Matrix
6.3.2.4.1 Defining the Confusion Matrix Function
6.3.2.4.2 Obtaining the Labels
6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate
6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate
6.3.2.4.5 Calculating Precision/Positive Predictive Value
6.3.2.4.6 Negative Predictive Value
6.3.2.4.7 Calculating Accuracy
Evaluating the Models Together on Testing Data - Ensembling the models
7.1 Defining the Input Shape
7.2 Defining all the Models
7.3 Appending All the Models
7.4 Defining the Ensembling Function
7.5 Evaluating ensemble model
7.5.1 Compute Test Set Predictions
7.5.2 Evaluating the Model
7.5.2.1 Re-ordering the Actual y for ROC
7.5.2.2 Re-ordering the Predict y for ROC
7.5.2.3 Plotting the Re-ordered ROC
7.5.2.4 Confusion Matrix
7.5.2.4.1 Defining the Confusion Matrix Function
7.5.2.4.2 Obtaining the Labels
7.5.2.4.3 Calculating
[249] 1 # Precision | positive predictive value (PPV)
2 predcision_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test + FALSE_POSITIVE_Ensemble_Test)
3 print("Precision: ", predcision_Ensemble_Test)
Precision: 0.805
7.5.2.4.6 Negative Predictive Value
[250] 1 # Negative predictive value (NPV)
2 try:
3 npv_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test / (TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
4 print("Negative predictive value: ", npv_Ensemble_Test)
5 except:
6 print("0 Negative Predictions")
0 Negative Predictions
7.5.2.4.7 Calculating Accuracy
[251] 1 # Accuracy
2 accuracy_Ensemble_Test = (TRUE_POSITIVE_Ensemble_Test + TRUE_NEGATIVE_Ensemble_Test) / (TRUE_POSITIVE_Ensemble_Test + FALSE_POSITIVE_Ensemble_Test + TRUE_NEGATIVE_Ensemble_Test
3 print("Accuracy: ", accuracy_Ensemble_Test)
Accuracy: 0.805
8. Localization
[252] 1 # Importing the libraries
2 from keras.applications.mobilenet import preprocess_input
3 import scipy
4 import cv2
8.1 Obtaining the Path for the MobileNet Architecture Models
[253] 1 path_to_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
18m 1s completed at 12:43 AM
Type here to search
01:17 AM
12-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
Accuracy: 0.805
8. Localization
8 Localization
8.1 Obtaining the Path for the MobileNet Architecture Models
8.2 Sample Image Path
8.3 Defining the function for the Layer Weights for MobileNet
8.4 Defining Class Activation Map Function
8.5 Plotting the Class Activation Function for MobileNet
8.6
26s completed at 7:28 PM
Type here to search
ENG
11:36 PM
14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
8.6 Visualizing the Images
Working Flowchart:
8.1 Obtaining the Path for the MobileNet Architecture Models
1 # Importing the libraries
2 from keras.applications.mobilenet import preprocess_input
3 import scipy
4 import cv2
1 path_to_model_weight = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5"
8.2 Sample Image Path
1 img_path = "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Test_v2_Data/Data Image JPG/ISIC_0012240.jpg"
8.3 Defining the function for the Layer Weights for MobileNet
1 def getting_two_layer_weights(path_model_weight = path_to_model_weight):
2 # The model
3
4 # Imprting the model
5 from keras.applications.mobilenet import MobileNet
6
26s completed at 7:28 PM
Type here to search
ENG
11:37 PM
14-05-2021



WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

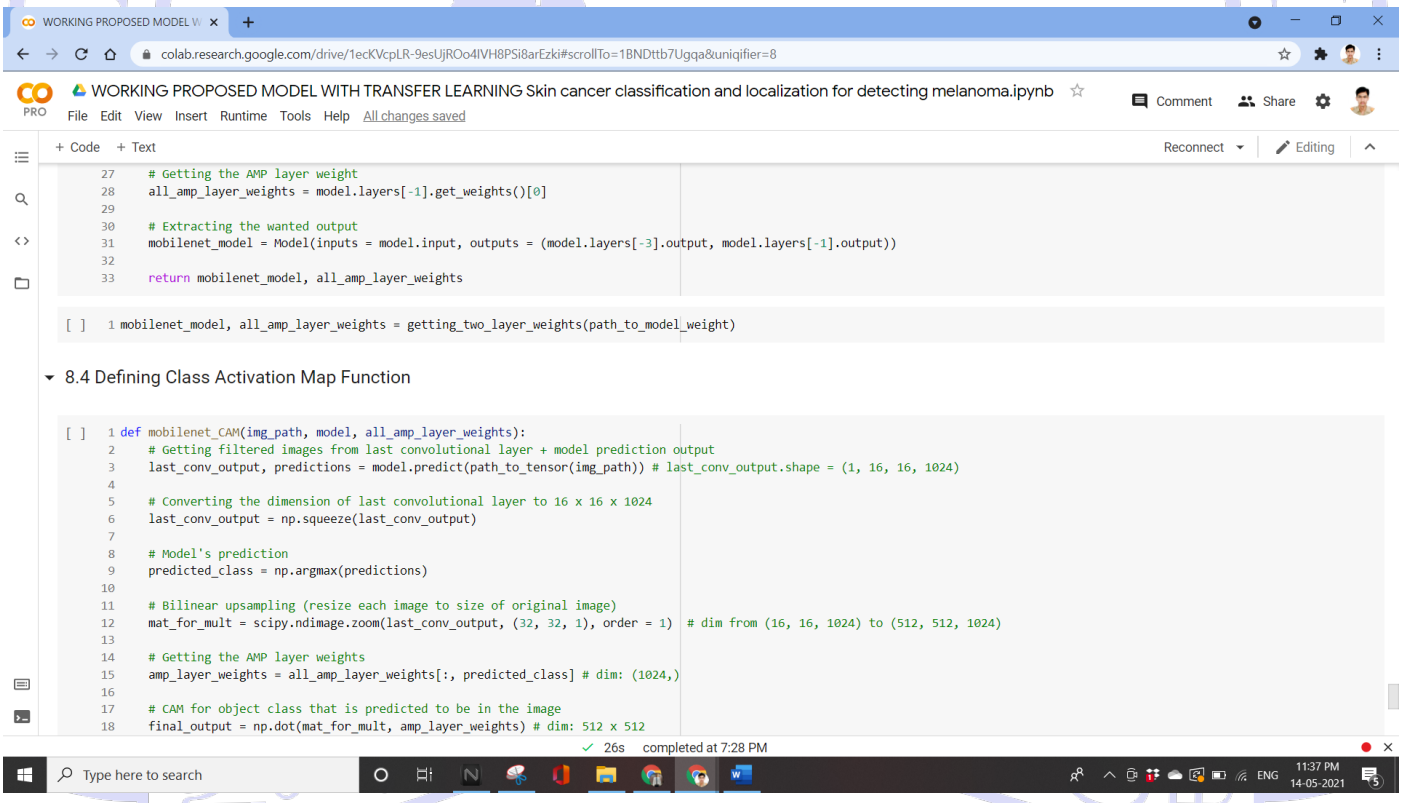
Reconnect Editing

8.3 Defining the function for the Layer Weights for MobileNet

```
def getting_two_layer_weights(path_model_weight = path_to_model_weight):
    # The model

    # Imprting the model
    from keras.applications.mobilenet import MobileNet

    # Pre-build model
    base_model = MobileNet(include_top = False, weights = None, input_shape = (512, 512, 3))

    # Adding output layers
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    output = Dense(units = 2, activation = 'softmax')(x)

    # Creating the whole model
    model = Model(base_model.input, output)
    #model.summary()

    # Compiling the model
    model.compile(optimizer = keras.optimizers.Adam(lr = 0.001),
                  loss = 'categorical_crossentropy',
                  metrics = ['accuracy'])

    # loading the weights
    model.load_weights(path_model_weight)

    # Getting the AMP layer weight
```

26s completed at 7:28 PM

Type here to search ENG 11:37 PM 14-05-2021



WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect Editing

```
    # Getting the AMP layer weight
    all_amp_layer_weights = model.layers[-1].get_weights()[0]

    # Extracting the wanted output
    mobilenet_model = Model(inputs = model.input, outputs = (model.layers[-3].output, model.layers[-1].output))

    return mobilenet_model, all_amp_layer_weights
```

```
[ ] mobilenet_model, all_amp_layer_weights = getting_two_layer_weights(path_to_model_weight)
```

8.4 Defining Class Activation Map Function

```
[ ] def mobilenet_CAM(img_path, model, all_amp_layer_weights):
    # Getting filtered images from last convolutional layer + model prediction output
    last_conv_output, predictions = model.predict(path_to_tensor(img_path)) # last_conv_output.shape = (1, 16, 16, 1024)

    # Converting the dimension of last convolutional layer to 16 x 16 x 1024
    last_conv_output = np.squeeze(last_conv_output)

    # Model's prediction
    predicted_class = np.argmax(predictions)

    # Bilinear upsampling (resize each image to size of original image)
    mat_for_mult = scipy.ndimage.zoom(last_conv_output, (32, 32, 1), order = 1)  # dim from (16, 16, 1024) to (512, 512, 1024)

    # Getting the AMP layer weights
    amp_layer_weights = all_amp_layer_weights[:, predicted_class] # dim: (1024,)

    # CAM for object class that is predicted to be in the image
    final_output = np.dot(mat_for_mult, amp_layer_weights) # dim: 512 x 512
```

26s completed at 7:28 PM

Type here to search ENG 11:37 PM 14-05-2021

WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Reconnect Editing

+ Code + Text

```
    final_output = np.dot(mat_for_mult, amp_layer_weights) # dim: 512 x 512

    # Return class activation map (CAM)
    return final_output, predicted_class
```

```
final_output, predicted_class = mobilenet_CAM(img_path, mobilenet_model, all_amp_layer_weights)
```

8.5 Plotting the Class Activation Function for MobileNet

```
def plot_CAM(img_path, ax, model, all_amp_layer_weights):
    # Loading the image / resizing to 512x512 / Converting BGR to RGB
    #im = cv2.resize(cv2.cvtColor(cv2.imread(img_path), cv2.COLOR_BGR2RGB), (512, 512))
    im = path_to_tensor(img_path).astype("float32")/255.

    # Plotting the image
    ax.imshow(im.squeeze(), vmin=0, vmax=255)

    # Getting the class activation map
    CAM, pred = mobilenet_CAM(img_path, model, all_amp_layer_weights)

    CAM = (CAM - CAM.min()) / (CAM.max() - CAM.min())

    # Plotting the class activation map
    ax.imshow(CAM, cmap = "jet", alpha = 0.5, interpolation='nearest', vmin=0, vmax=1)
```

8.6 Visualizing the Images

26s completed at 7:28 PM

Type here to search ENG 11:37 PM 14-05-2021

WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

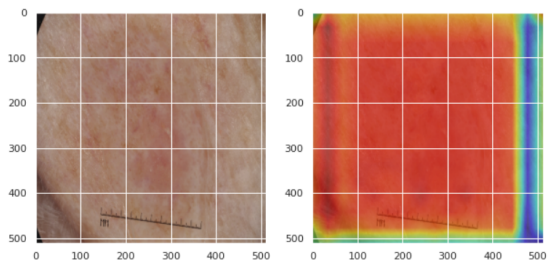
File Edit View Insert Runtime Tools Help All changes saved

Comment Share Reconnect Editing

+ Code + Text

8.6 Visualizing the Images

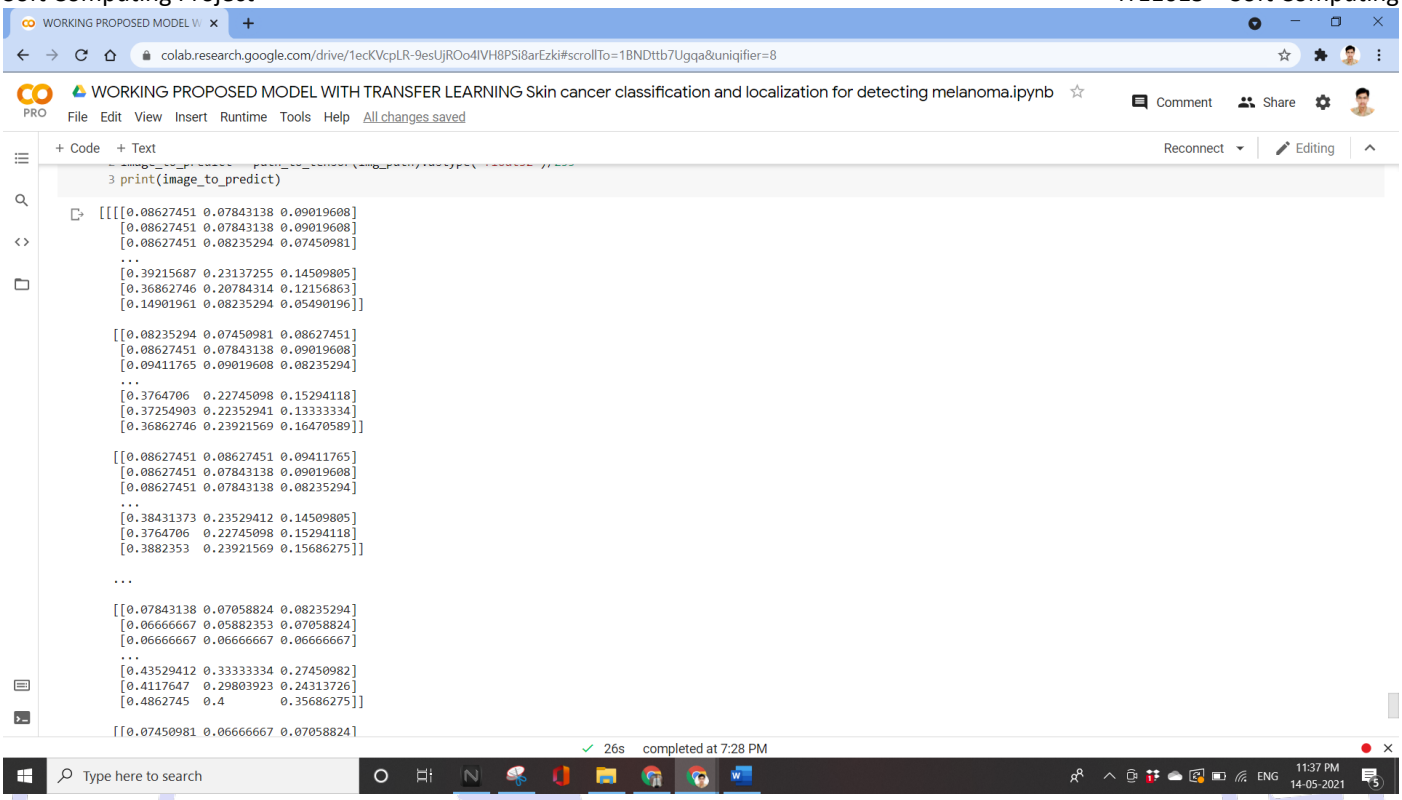
```
# Visualizing images with and without localization
# Canvas
fig, ax = plt.subplots(nrows=1, ncols=2, figsize = (10, 10))
# Image without localization
ax[0].imshow((path_to_tensor(img_path).astype('float32')/255).squeeze())
# Image with localization
CAM = plot_CAM(img_path, ax[1], mobilenet_model, all_amp_layer_weights)
plt.show()
```

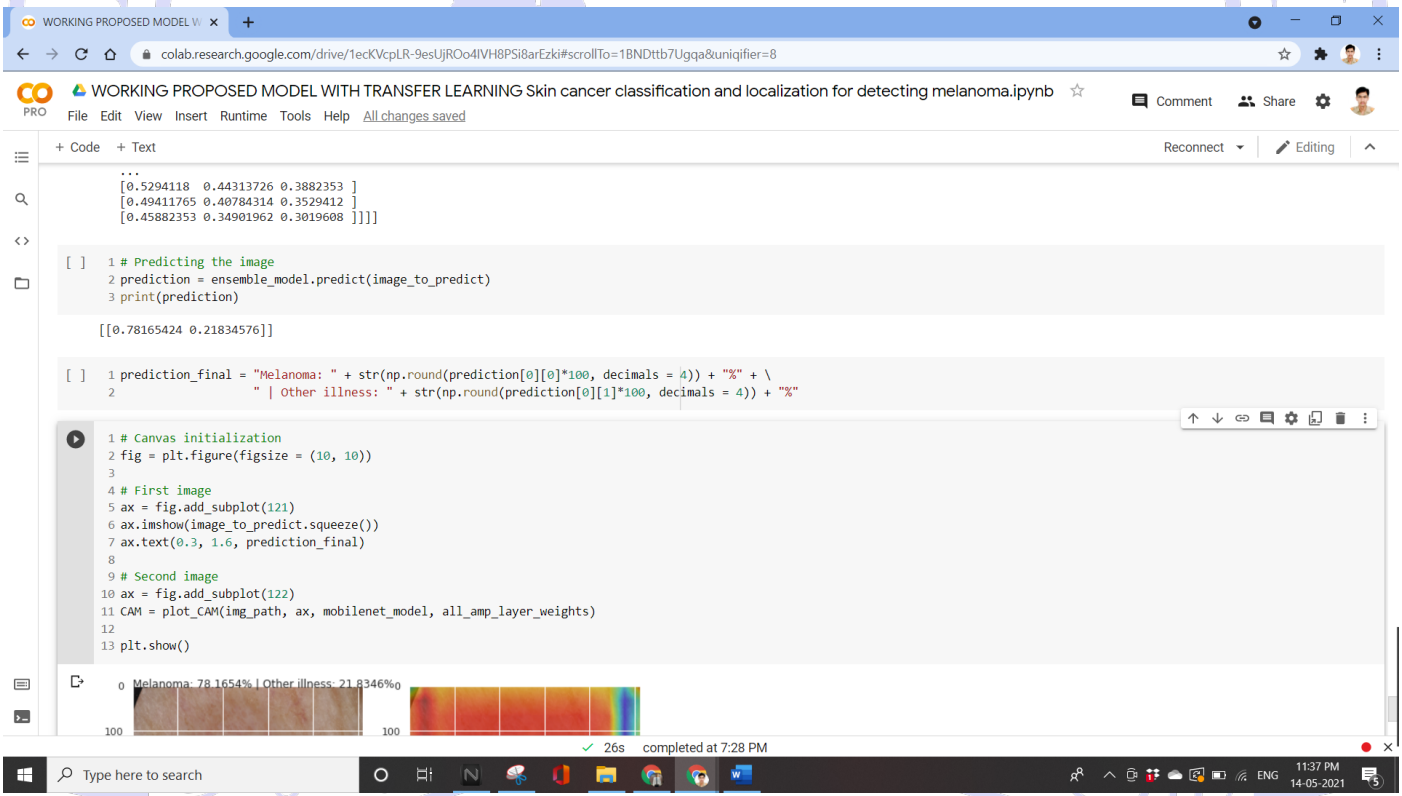



```
# Getting the image tensor
image_to_predict = path_to_tensor(img_path).astype('float32')/255
print(image_to_predict)
```

26s completed at 7:28 PM

Type here to search ENG 11:37 PM 14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
3 print(image_to_predict)
[[[[0.08627451 0.07843138 0.09019608]
[0.08627451 0.07843138 0.09019608]
[0.08627451 0.08235294 0.07450981]
...
[0.39215687 0.23137255 0.14509805]
[0.36862746 0.20784314 0.12156863]
[0.14901961 0.08235294 0.05490196]]
[[0.08235294 0.07450981 0.08627451]
[0.08627451 0.07843138 0.09019608]
[0.09411765 0.09019608 0.08235294]
...
[0.3764706 0.22745098 0.15294118]
[0.37254903 0.22352941 0.13333334]
[0.36862746 0.23921569 0.16470589]]
[[0.08627451 0.08627451 0.09411765]
[0.08627451 0.07843138 0.09019608]
[0.08627451 0.07843138 0.08235294]
...
[0.38431373 0.23529412 0.14509805]
[0.3764706 0.22745098 0.15294118]
[0.3882353 0.23921569 0.15686275]]
...
[[0.07843138 0.07058824 0.08235294]
[0.06666667 0.05882353 0.07058824]
[0.06666667 0.06666667 0.06666667]
...
[0.43529412 0.33333334 0.27450982]
[0.4117647 0.29803923 0.24313726]
[0.4862745 0.4 0.35686275]]
[[0.07450981 0.06666667 0.07058824]
26s completed at 7:28 PM
Type here to search
ENG
11:37 PM
14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
...
[0.5294118 0.44313726 0.3882353]
[0.49411765 0.40784314 0.3529412]
[0.45882353 0.34901962 0.3019608]]]]
[ ] 1 # Predicting the image
2 prediction = ensemble_model.predict(image_to_predict)
3 print(prediction)
[[0.78165424 0.21834576]]
[ ] 1 prediction_final = "Melanoma: " + str(np.round(prediction[0][0]*100, decimals = 4)) + "%" + \
2 " | Other illness: " + str(np.round(prediction[0][1]*100, decimals = 4)) + "%"
1 # Canvas initialization
2 fig = plt.figure(figsize = (10, 10))
3
4 # First image
5 ax = fig.add_subplot(121)
6 ax.imshow(image_to_predict.squeeze())
7 ax.text(0.3, 1.6, prediction_final)
8
9 # Second image
10 ax = fig.add_subplot(122)
11 CAM = plot_CAM(img_path, ax, mobilenet_model, all_amp_layer_weights)
12
13 plt.show()
Melanoma: 78.1654% | Other illness: 21.8346%
0
100
0
100
26s completed at 7:28 PM
Type here to search
ENG
11:37 PM
14-05-2021

WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

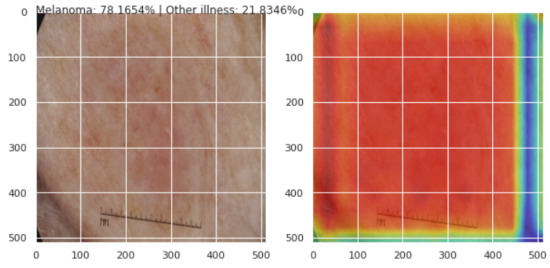
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect Editing

```
# Canvas initialization
fig = plt.figure(figsize = (10, 10))

# First image
ax = fig.add_subplot(121)
ax.imshow(image_to_predict.squeeze())
ax.text(0.3, 1.6, prediction_final)

# Second image
ax = fig.add_subplot(122)
CAM = plot_CAM(img_path, ax, mobilenet_model, all_amp_layer_weights)

plt.show()
```



9. Saving the Complete Model for the Python Interface Application

26s completed at 7:28 PM

Type here to search

ENG 11:38 PM 14-05-2021

WORKING PROPOSED MODEL W

colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=1BNDttb7Ugqa&uniqifier=8

WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb

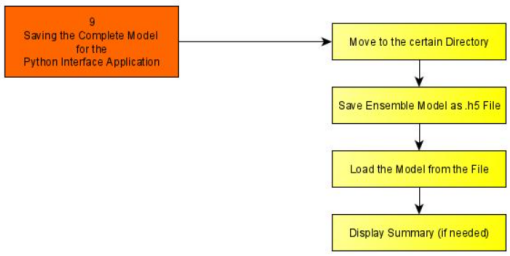
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect Editing

9. Saving the Complete Model for the Python Interface Application



Working Flowchart:

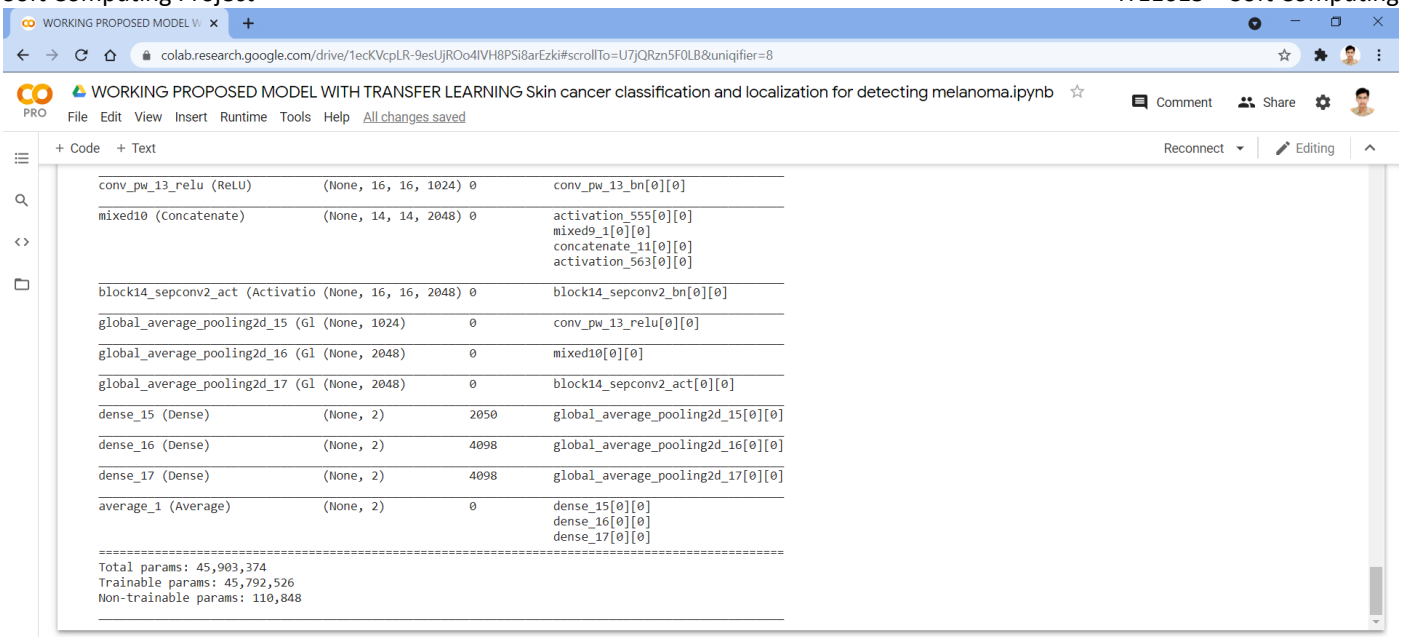
```
%cd "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPUTS/"
ensemble_model.save('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')
h5_saved_ensemble_model = load_model('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')
h5_saved_ensemble_model.summary()
```

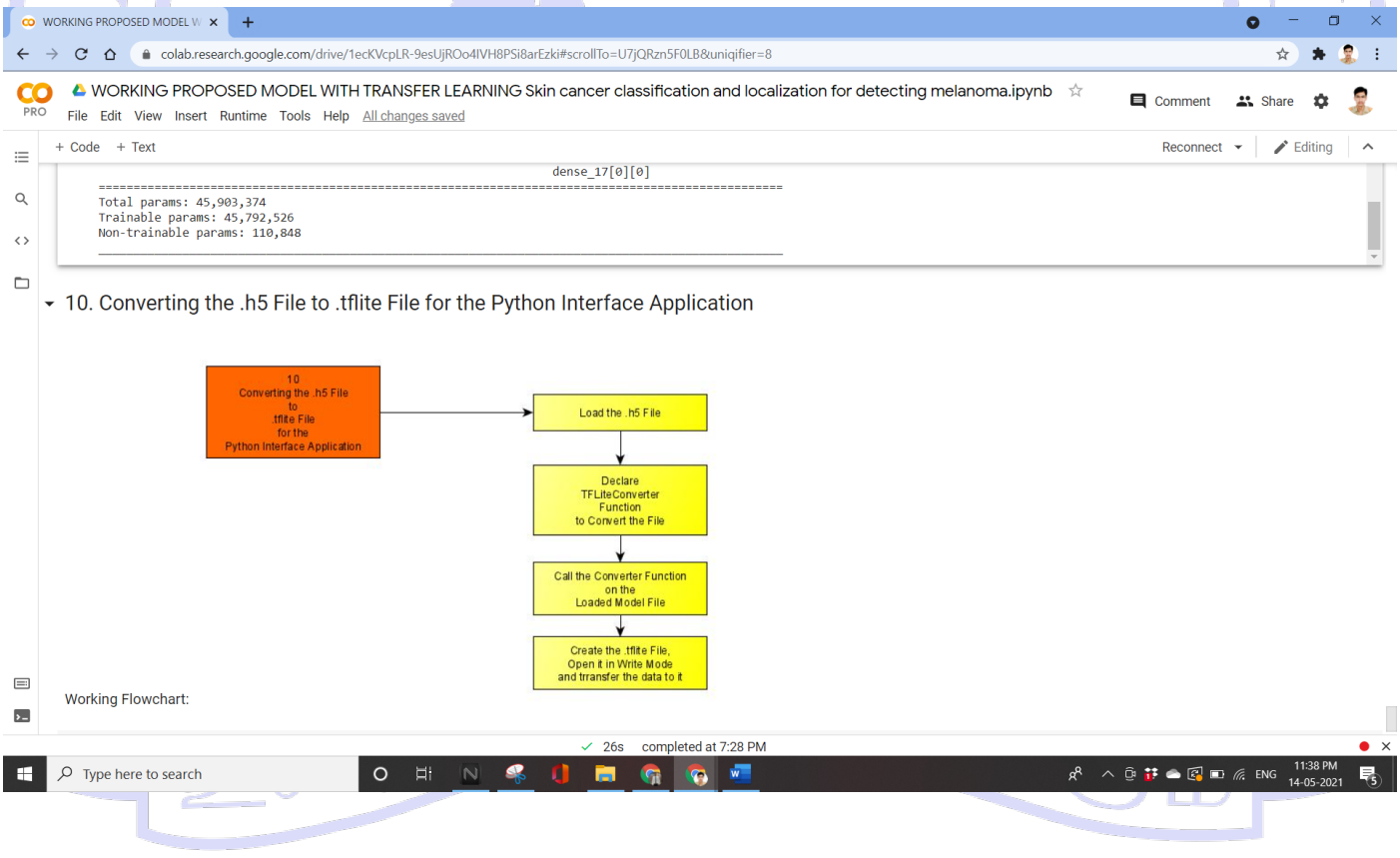
```
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPUTS
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
Model: "ensemble"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_11 (InputLayer)        [(None, 512, 512, 3) 0
```

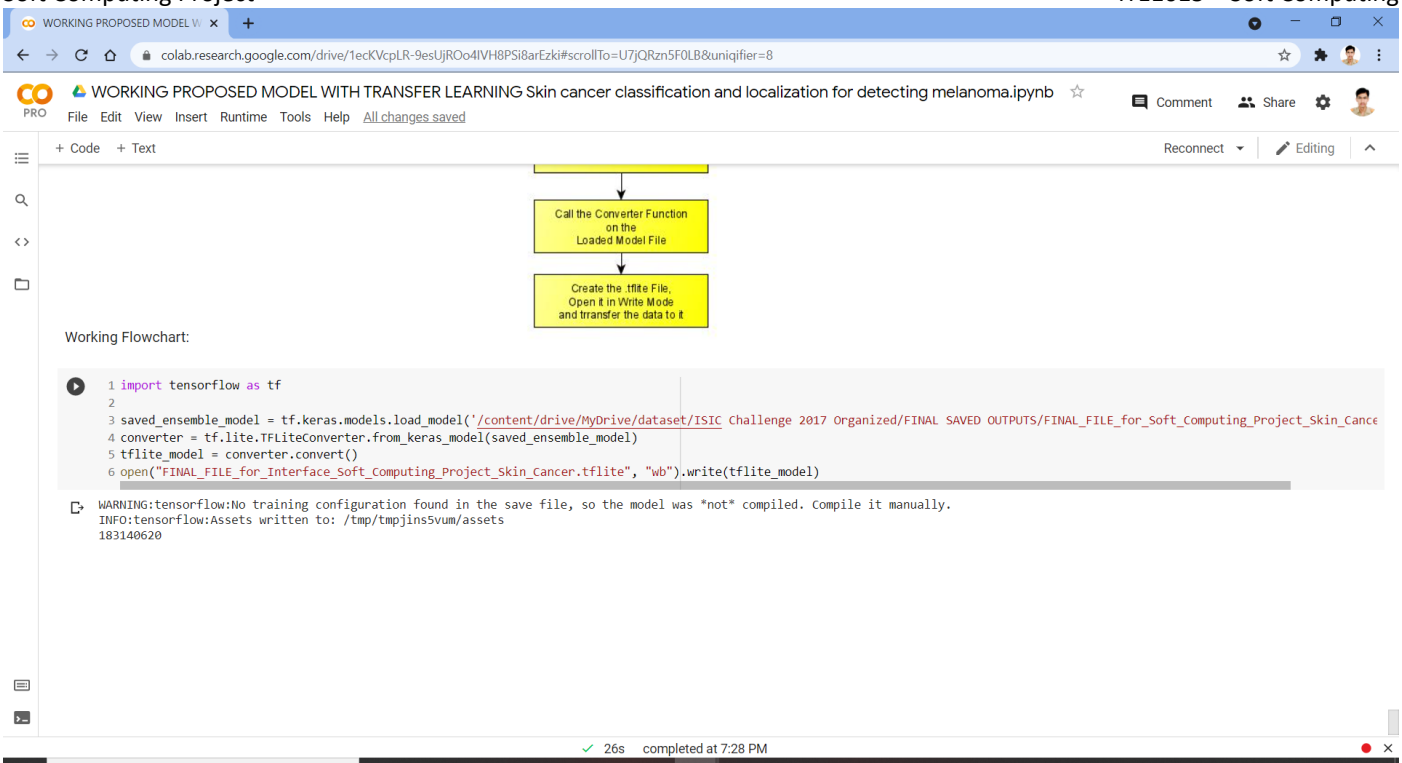
26s completed at 7:28 PM

Type here to search

ENG 11:38 PM 14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=U7jQRzn5F0LB&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
conv_pw_13_relu (ReLU) (None, 16, 16, 1024) 0 conv_pw_13_bn[0][0]
mixed10 (Concatenate) (None, 14, 14, 2048) 0 activation_555[0][0]
mixed9_1[0][0]
concatenate_11[0][0]
activation_563[0][0]
block14_sepconv2_act (Activatio (None, 16, 16, 2048) 0 block14_sepconv2_bn[0][0]
global_average_pooling2d_15 (Gl (None, 1024) 0 conv_pw_13_relu[0][0]
global_average_pooling2d_16 (Gl (None, 2048) 0 mixed10[0][0]
global_average_pooling2d_17 (Gl (None, 2048) 0 block14_sepconv2_act[0][0]
dense_15 (Dense) (None, 2) 2050 global_average_pooling2d_15[0][0]
dense_16 (Dense) (None, 2) 4098 global_average_pooling2d_16[0][0]
dense_17 (Dense) (None, 2) 4098 global_average_pooling2d_17[0][0]
average_1 (Average) (None, 2) 0 dense_15[0][0]
dense_16[0][0]
dense_17[0][0]
Total params: 45,903,374
Trainable params: 45,792,526
Non-trainable params: 110,848
10. Converting the .h5 File to .tflite File for the Python Interface Application
26s completed at 7:28 PM
Type here to search
ENG
11:38 PM
14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=U7jQRzn5F0LB&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
dense_17[0][0]
Total params: 45,903,374
Trainable params: 45,792,526
Non-trainable params: 110,848
10. Converting the .h5 File to .tflite File for the Python Interface Application
10
Converting the .h5 File
to
.tflite File
for the
Python Interface Application
Load the .h5 File
Declare
TFLiteConverter
Function
to Convert the File
Call the Converter Function
on the
Loaded Model File
Create the .tflite File,
Open it in Write Mode
and transfer the data to it
Working Flowchart:
26s completed at 7:28 PM
Type here to search
ENG
11:38 PM
14-05-2021


WORKING PROPOSED MODEL W
colab.research.google.com/drive/1ecKVcpLR-9esUjROo4IVH8PSi8arEzki#scrollTo=U7jQRzn5F0LB&uniqifier=8
WORKING PROPOSED MODEL WITH TRANSFER LEARNING Skin cancer classification and localization for detecting melanoma.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
Call the Converter Function on the Loaded Model File
Create the .tflite File, Open it in Write Mode and trransfer the data to it
Working Flowchart:
1 import tensorflow as tf
2
3 saved_ensemble_model = tf.keras.models.load_model('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPUTS/FINAL_FILE_for_Soft_Computing_Project_Skin_Cance
4 converter = tf.lite.TFLiteConverter.from_keras_model(saved_ensemble_model)
5 tflite_model = converter.convert()
6 open("FINAL_FILE_for_Interface_Soft_Computing_Project_Skin_Cancer.tflite", "wb").write(tflite_model)
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
INFO:tensorflow:Assets written to: /tmp/tmpjins5vum/assets
183140620
26s completed at 7:28 PM
Type here to search
ENG
11:38 PM
14-05-2021

PROPOSED ARCHITECTURE

PRE-PROCESSING

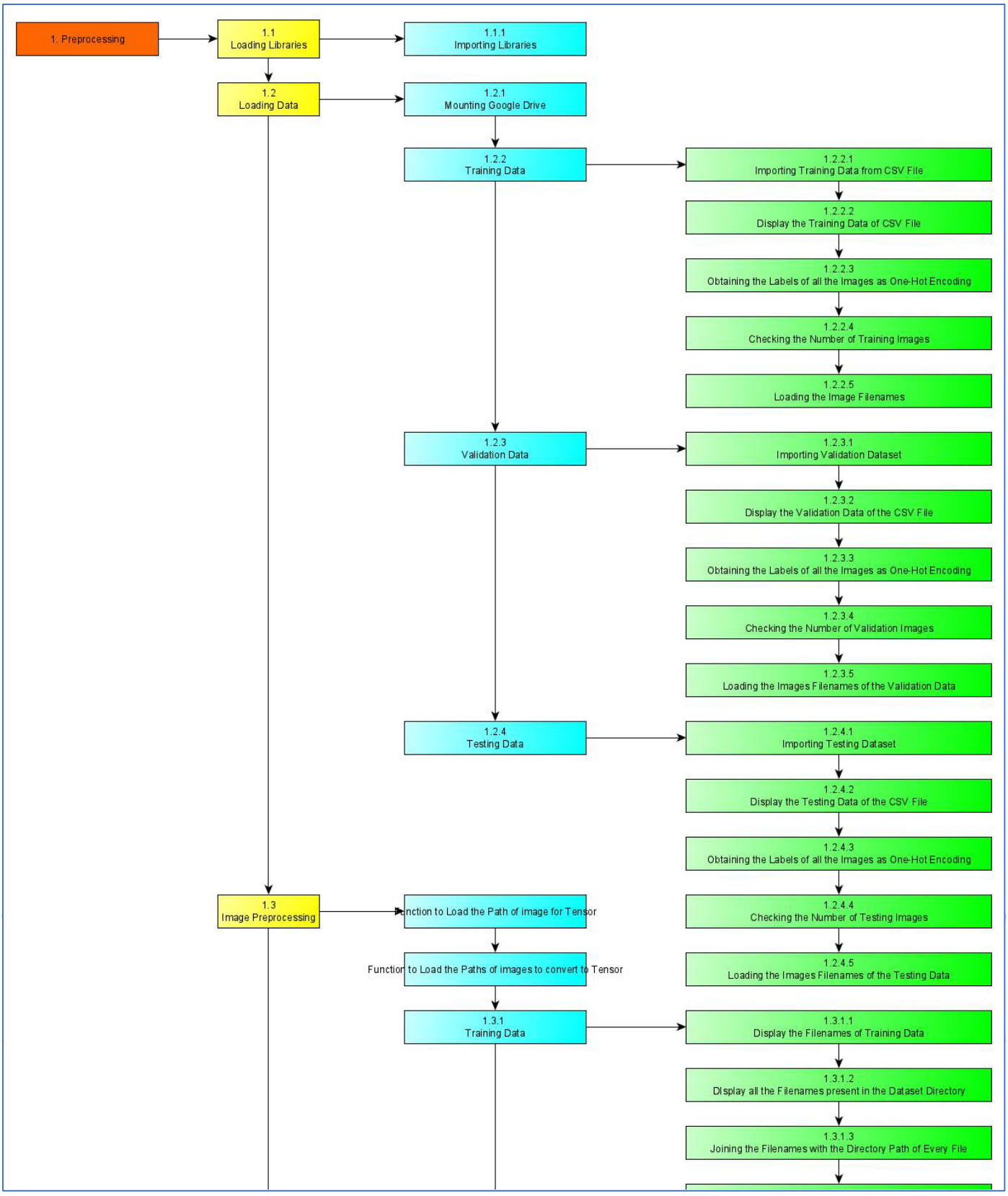


Figure 4: Proposed Architecture - Pre-processing(i)

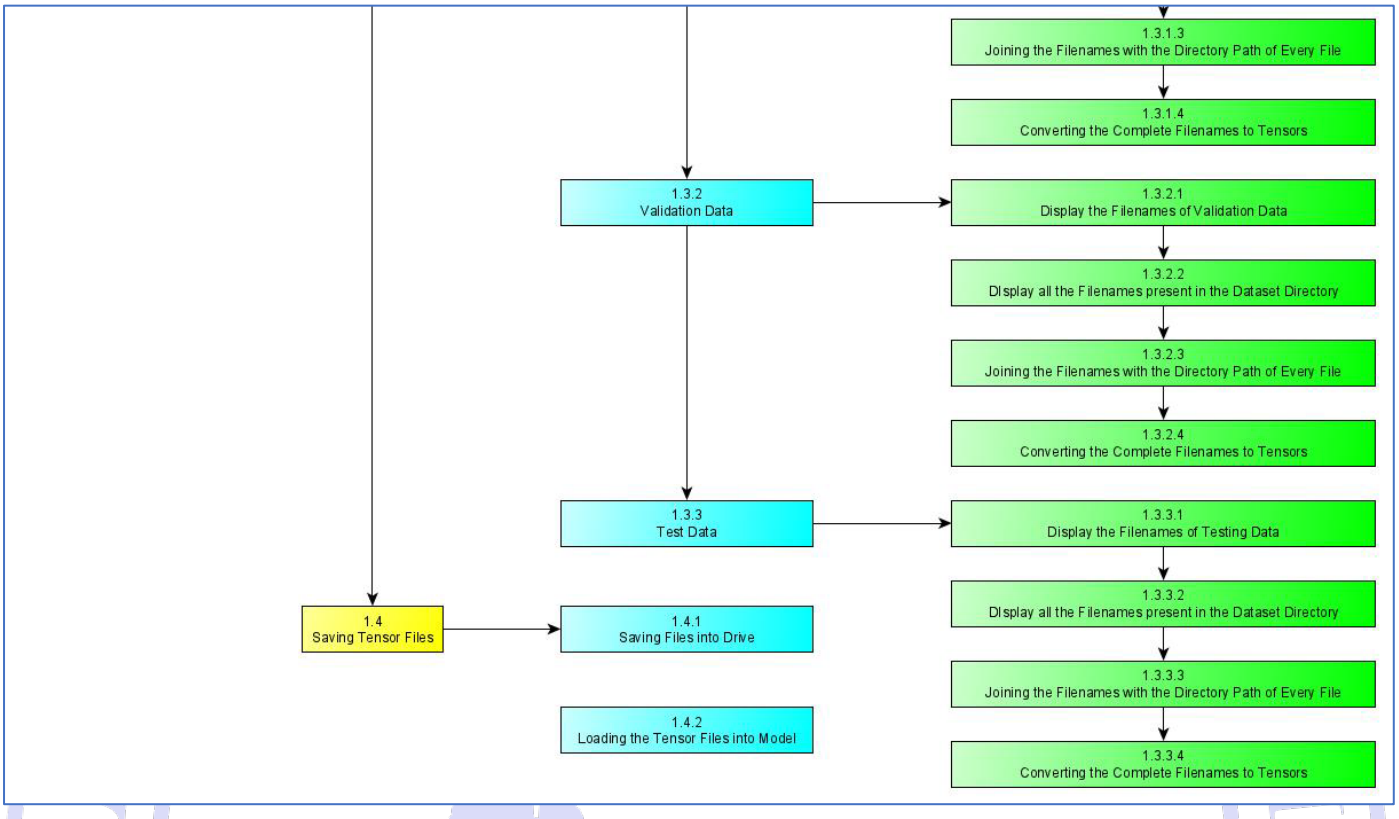


Figure 5: Proposed Architecture - Pre-processing(ii)

All the required libraries are first loaded into the model and then the dataset is also loaded into the model using the Google Drive as the Dataset has been uploaded to the Google Drive it can be accessed directly through it.

After the Google Drive is connected and authorised for use, the CSV files of the Training, Validation and Testing Data are read one-by-one and the data which they contain is displayed. Further, it is required for the dataset images to have their directory location attached to the filenames because then the images can be accessed directly with it without changing the directory which might interrupt any further operation.

After the directory locations are attached to the filenames, the images are loaded into the model one-by-one and they are converted into tensors. After obtaining the tensors, they are stored in a separate location so that these tensors can be used directly and it eliminates the need for loading the dataset and processing it again and again and instead these tensor files can just be loaded into the model and used for further training and processing. This also saves a lot of time because we do not need to create the tensors again and again as the tensors will be same each time they are being created because it is the same images which might be used during the development of the project.

TRAINING MODEL

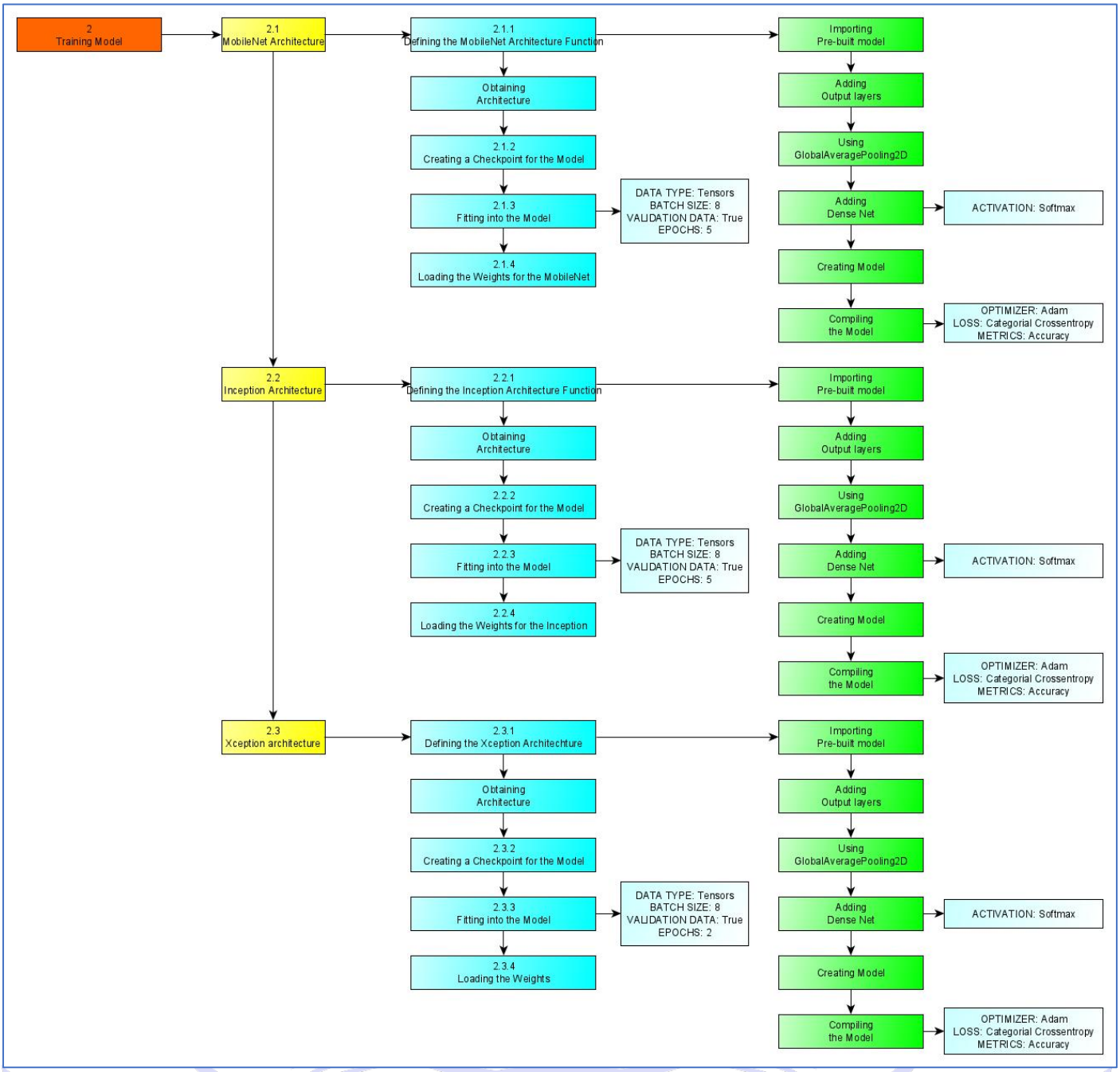


Figure 6: Proposed Architecture - Training Model

For this, we have used Transfer Learning and Ensemble Modelling. For the Transfer Learning, we have used the following pre-trained models:

- MobileNet Architecture
- InceptionV3 Architecture
- Xception Architecture

For the training, the pre-trained models are first loaded in as a function and then a since we have a checkpoint of these models, so we can start training these models from that checkpoint again. We start training it further by fitting it on the data which we have using the tensors.

PREDICTION

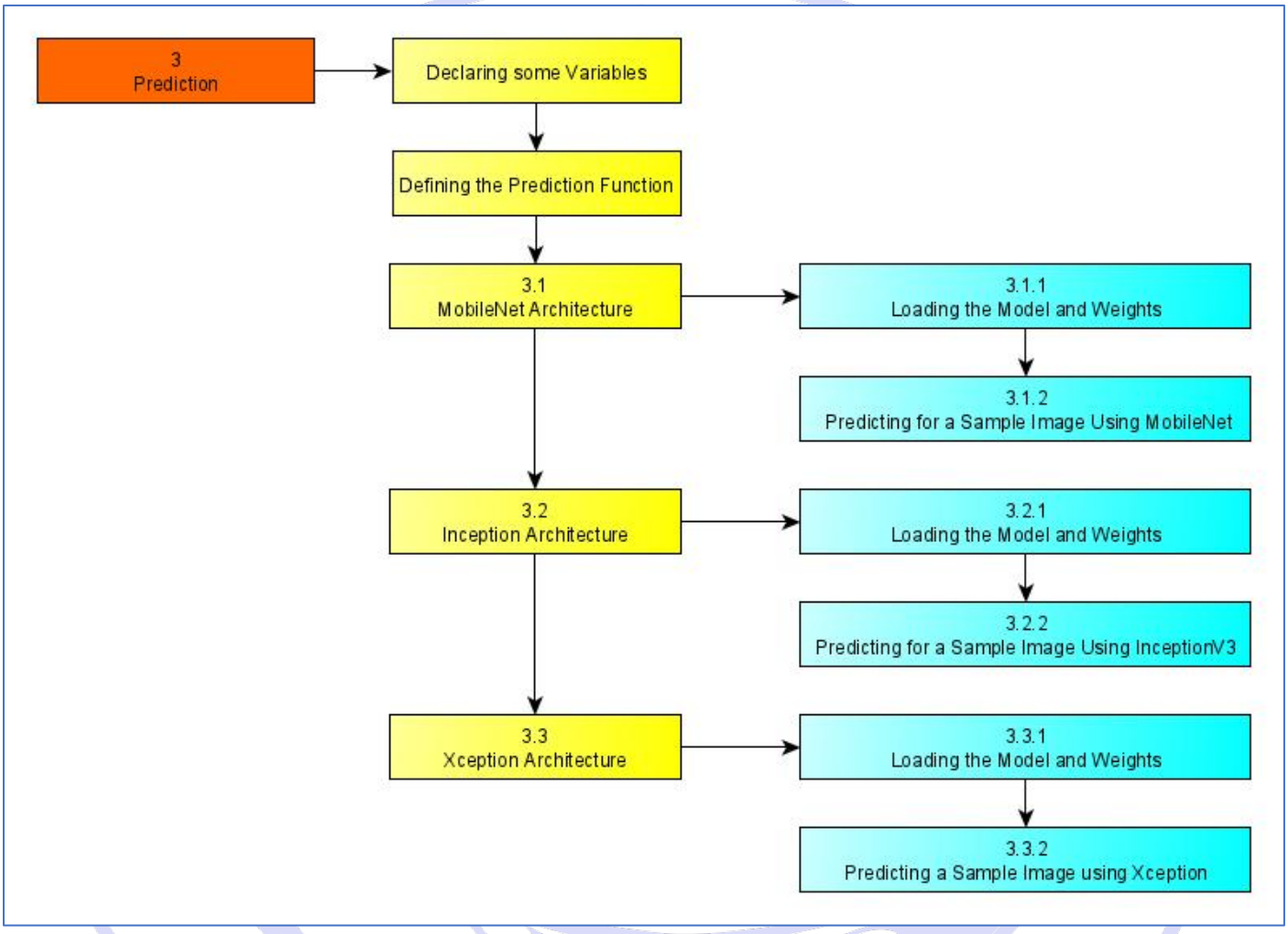


Figure 7: Proposed Architecture - Prediction Function

This prediction is done on a sample image just to check that all the models are working and are predicting. Working of this section denotes that the model can be taken to the further stage of evaluation of the validation data and the testing data.

For this part, we take the sample image and convert that image into a tensor. After that, the Individual Models and their respective weights are loaded so that the prediction can be made using the predictive function which we have defined. This is the function which will also be used for the Interface Application for the prediction on the image which will be uploaded by the user.

EVALUATING THE MODELS INDIVIDUALLY ON VALIDATION DATA

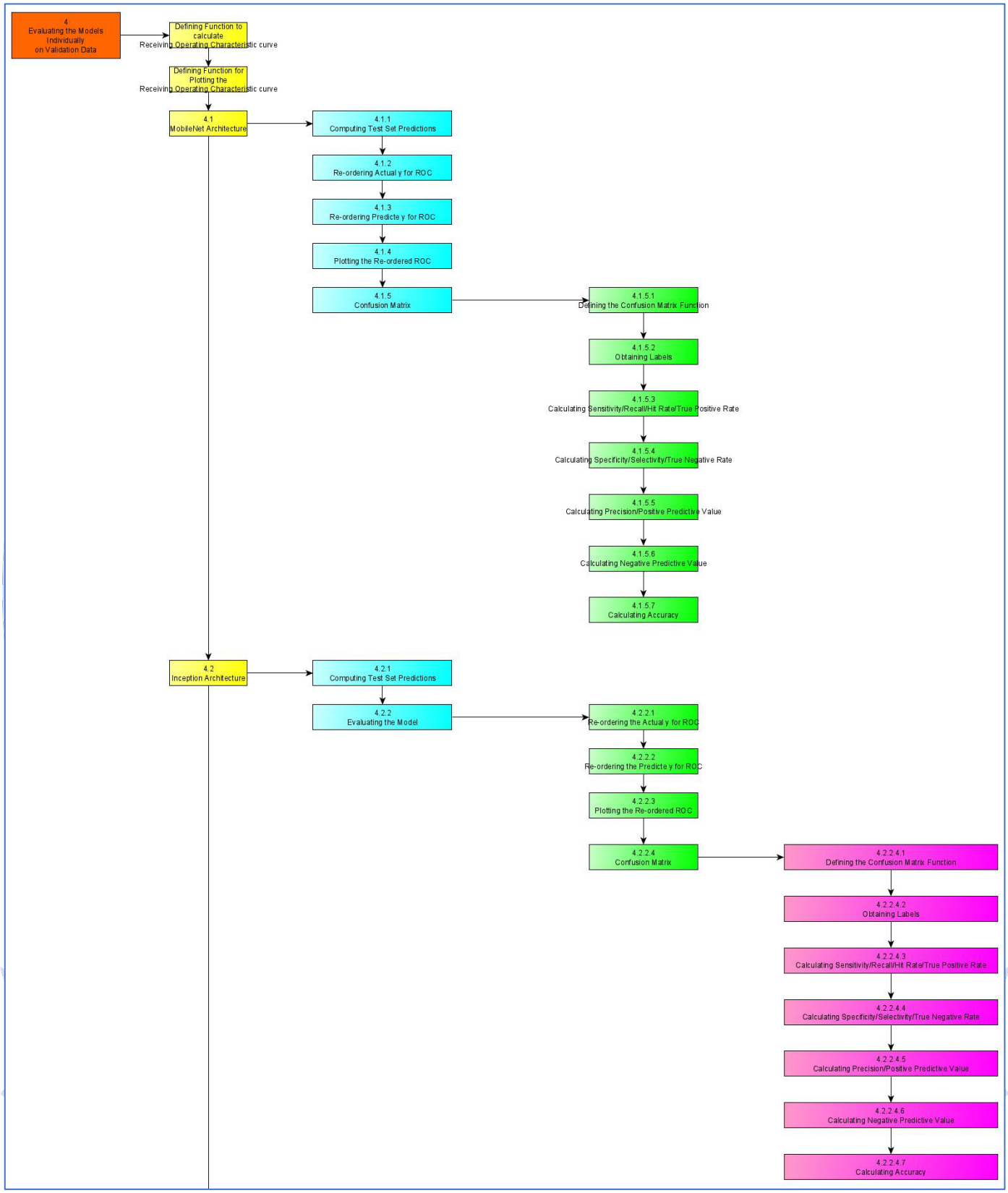


Figure 8: Proposed Architecture - Evaluating Models Individually on Validation Data (i)

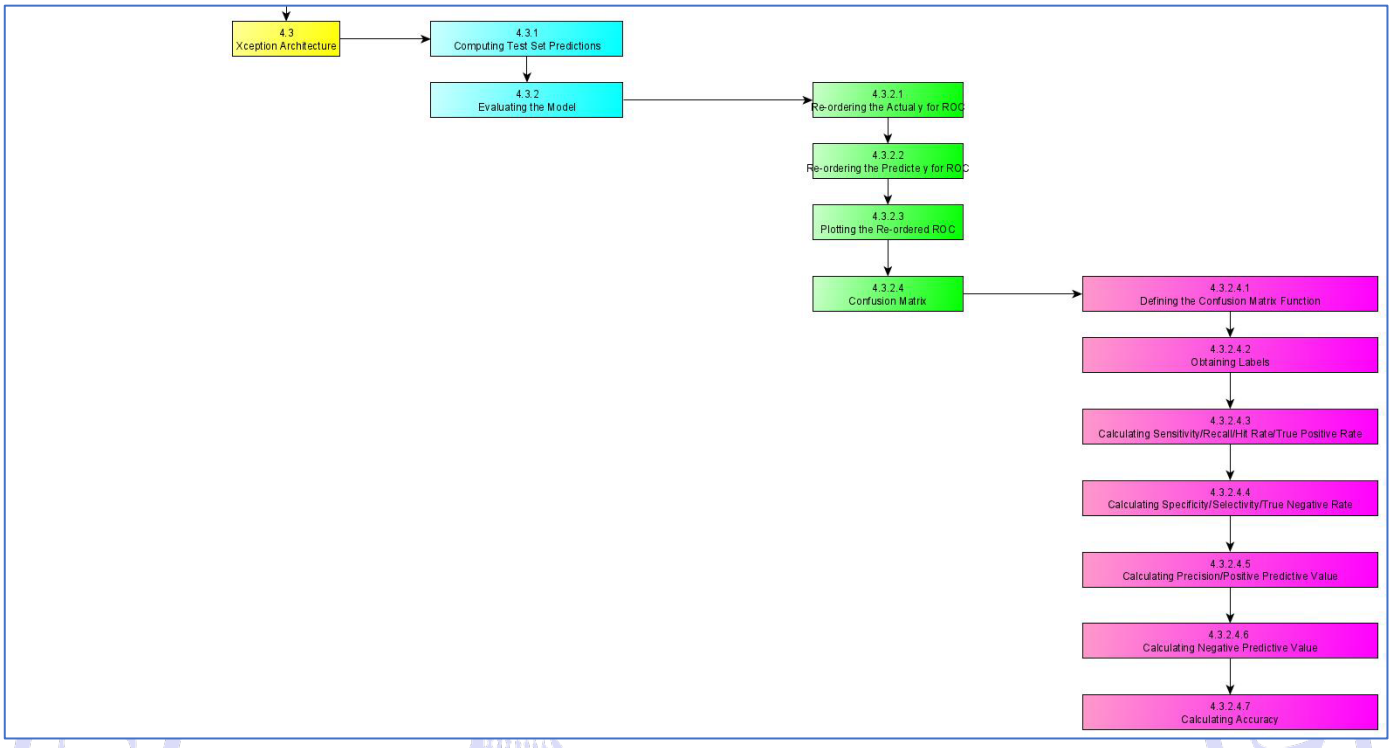


Figure 9: Proposed Architecture - Evaluating the Models Individually on Validation Data (ii)

For the evaluation of the models on the Validation Data, we have used the Receiving Operating Characteristic Curve. A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The method was originally developed for operators of military radar receivers starting in 1941, which led to its name.

For this, the we will be computing the test set predictions and then the model will be evaluated based on the Receiving Operating Characteristic Curve whose values will be obtained using the Confusion Matrix. Then, we will plot the values obtained for this in order to obtain the graph and check for the best possible values which can be used for prediction.

Further, after obtaining all these values, we will be calculating the Precision, Recall, Specificity, Sensitivity, Negative Predictive Value and Accuracy for further evaluation and after this, we can check the parameters, and if possible, we can tune the parameters to obtain better results.

At the end, we can compare the test results for the pre-trained architecture, as we have noted that the loss had reduced to the most efficient values possible and if we will train the model further with more epochs on the architectures, then we might make the model overfit on the data which can decrease the results and might lead to a higher number of wrong predictions, furthermore reducing the accuracy of the model and its efficiency.

EVALUATING THE MODELS TOGETHER ON VALIDATION DATA – ENSEMBLING THE MODELS

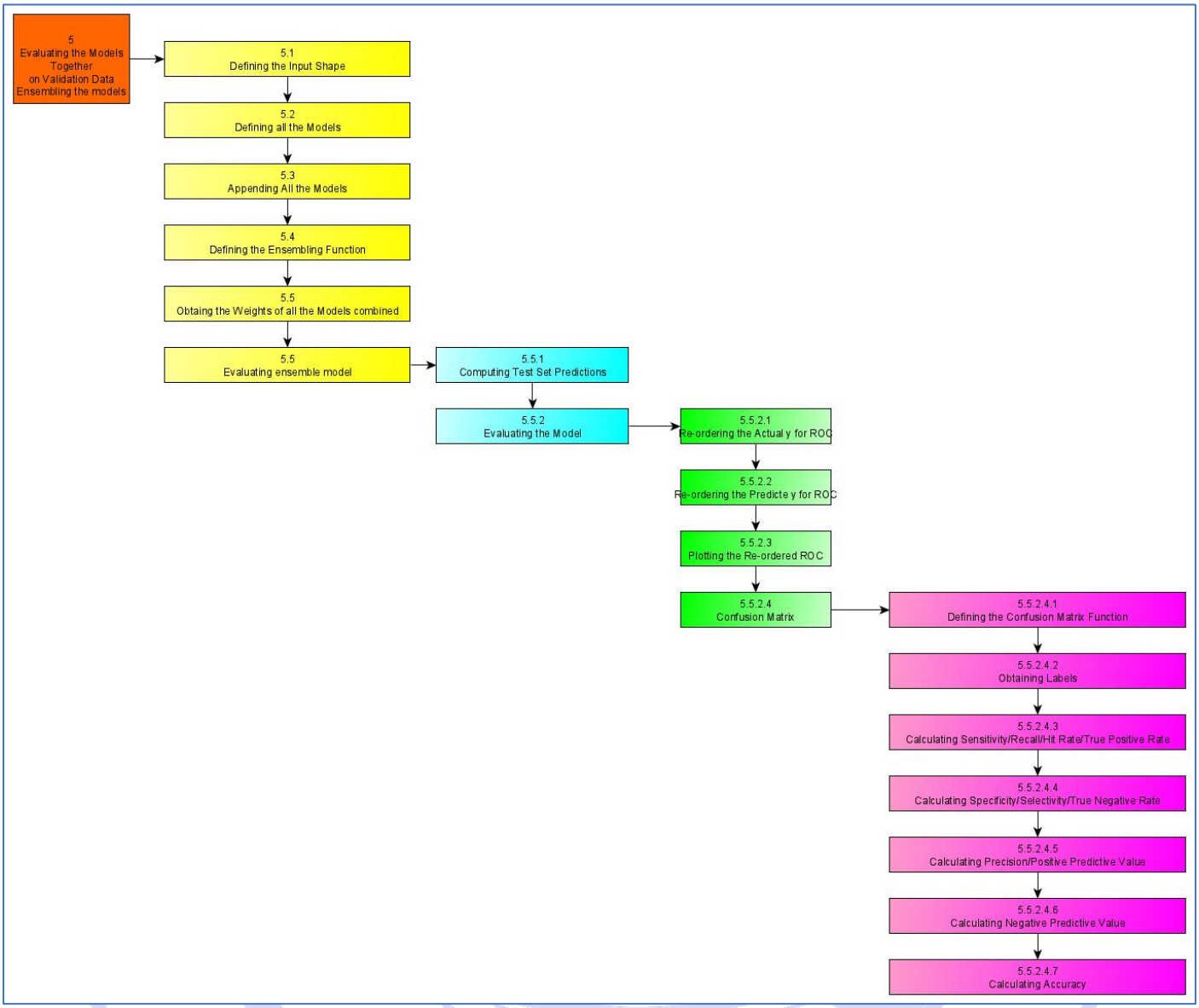


Figure 10: Proposed Architecture - Evaluating the Models Together on Validation Data

Defining a specific input shape for all the input images. The models will be brought together and appended so that the image passes through all the models one-by-one. The ensemble model will be defined which will contain all the models and the then the weights of the model will be combined and a file will be generated so that the weights are together into one and then they can be used together for the for the complete architecture.

For evaluating the models, we will be using the test set predictions and the ROC curve will be used whose computational values will be obtained using the Confusion Matrix. The other evaluation metrics are Precision, Recall, Sensitivity, Specificity, Negative Predictive Value and Accuracy. The results obtained for the Ensemble Model are better than each architecture individually.

EVALUATING THE MODELS INDIVIDUALLY ON TESTING DATA

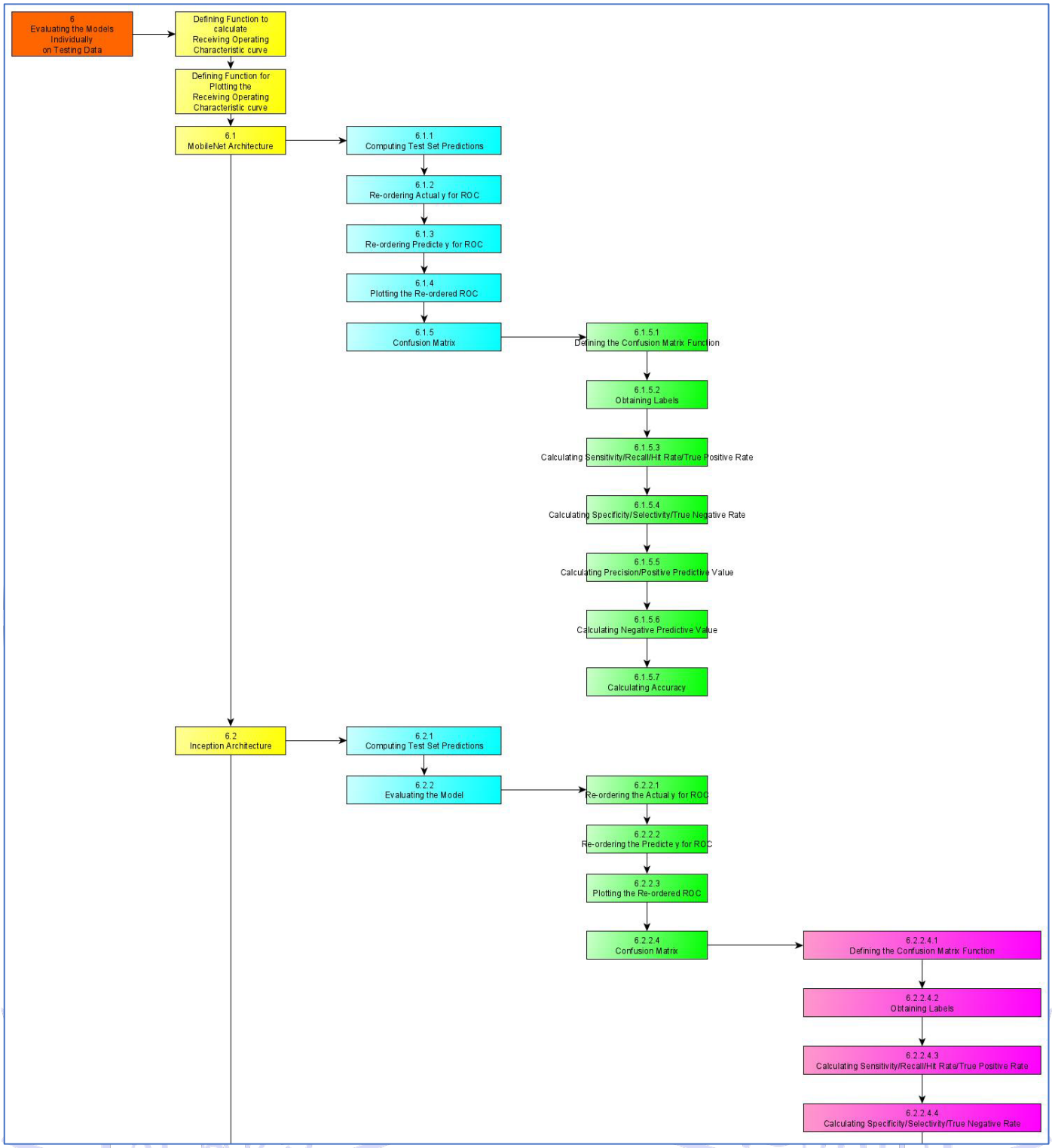


Figure 11: Proposed Architecture - Evaluating the Models Individually on Testing Data (i)

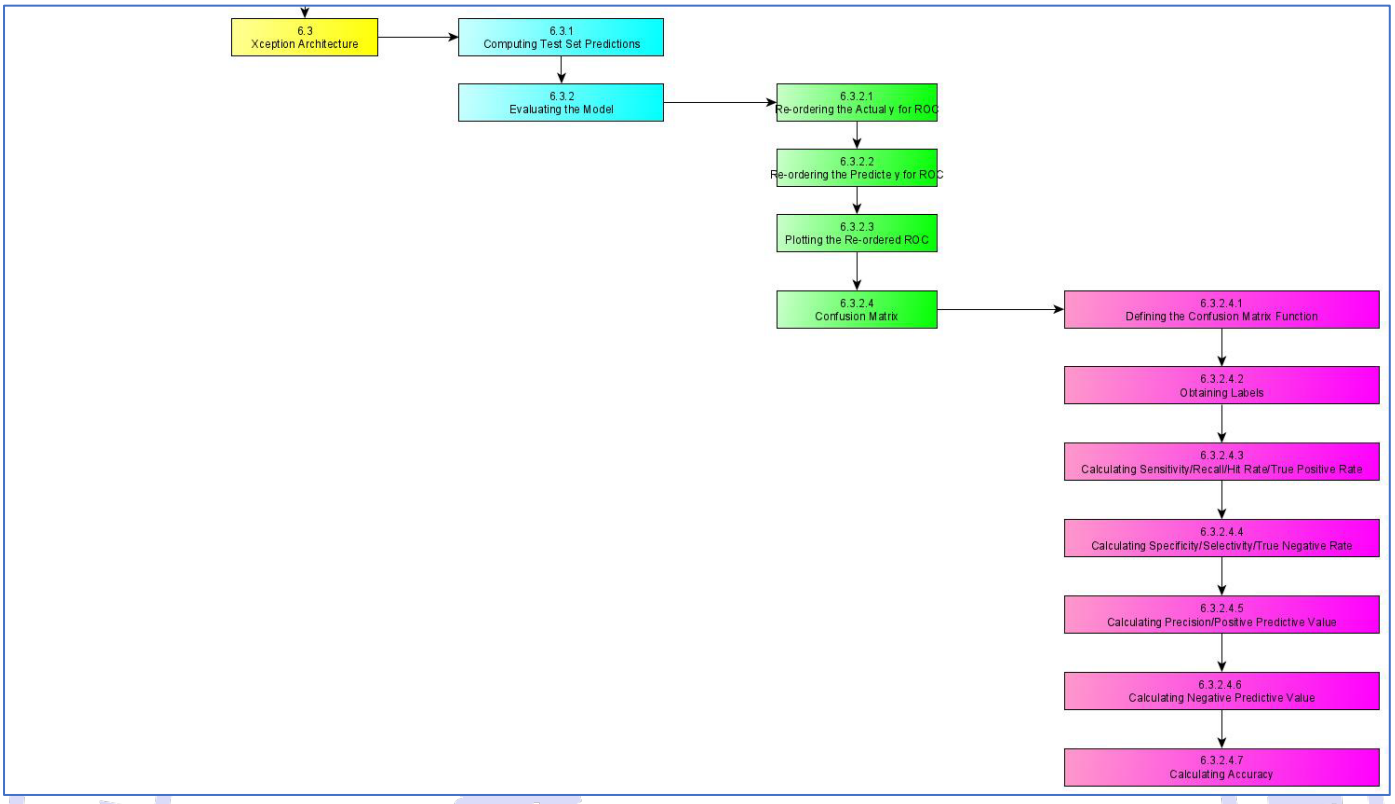


Figure 12: Proposed Architecture - Evaluating the Models Individually on Testing Data (ii)

For the evaluation of the models on the Testing Data, we have used the Receiving Operating Characteristic Curve. A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The method was originally developed for operators of military radar receivers starting in 1941, which led to its name.

For this, the we will be computing the test set predictions and then the model will be evaluated based on the Receiving Operating Characteristic Curve whose values will be obtained using the Confusion Matrix. Then, we will plot the values obtained for this in order to obtain the graph and check for the best possible values which can be used for prediction.

Further, after obtaining all these values, we will be calculating the Precision, Recall, Specificity, Sensitivity, Negative Predictive Value and Accuracy for further evaluation and after this, we can check the parameters, and if possible, we can tune the parameters to obtain better results.

At the end, we can compare the test results for the pre-trained architecture, as we have noted that the loss had reduced to the most efficient values possible and if we will train the model further with more epochs on the architectures, then we might make the model overfit on the data which can decrease the results and might lead to a higher number of wrong predictions, furthermore reducing the accuracy of the model and its efficiency.

The results obtained on the Testing data are very similar to the results obtained on the Validation data.

EVALUATING THE MODELS TOGETHER ON TESTING DATA – ENSEMBLING THE MODELS

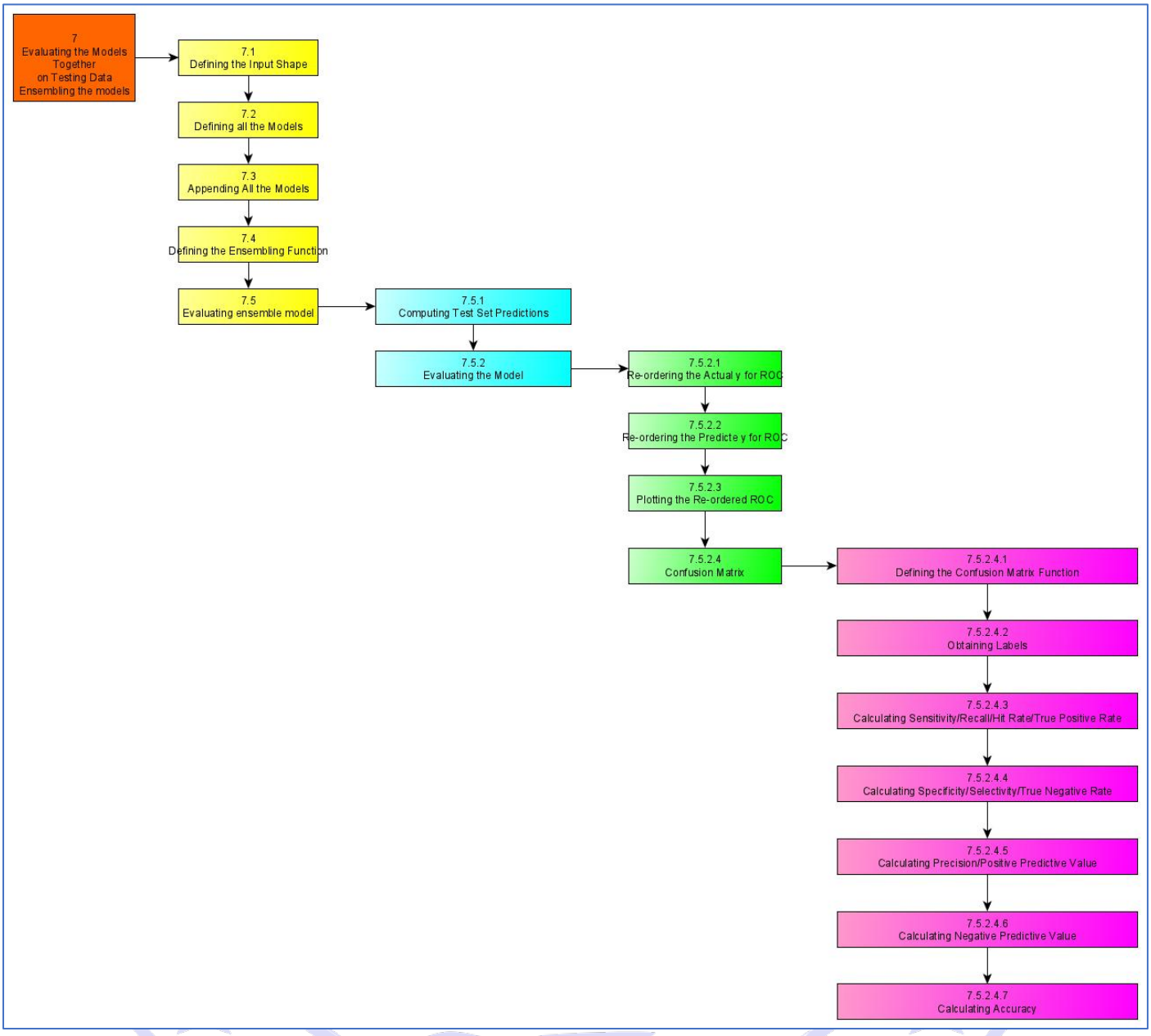


Figure 13: Proposed Model - Evaluating the Models Together on Testing Data

Defining a specific input shape for all the input images. The models will be brought together and appended so that the image passes through all the models one-by-one. The ensemble model will be defined which will contain all the models and the then the weights of the model will be combined and a file will be generated so that the weights are together into one and then they can be used together for the for the complete architecture.

For evaluating the models, we will be using the test set predictions and the ROC curve will be used whose computational values will be obtained using the Confusion Matrix. The other evaluation metrics are Precision, Recall, Sensitivity, Specificity, Negative Predictive Value and Accuracy. The results obtained for the Ensemble Model are better than each architecture individually.

LOCALIZATION

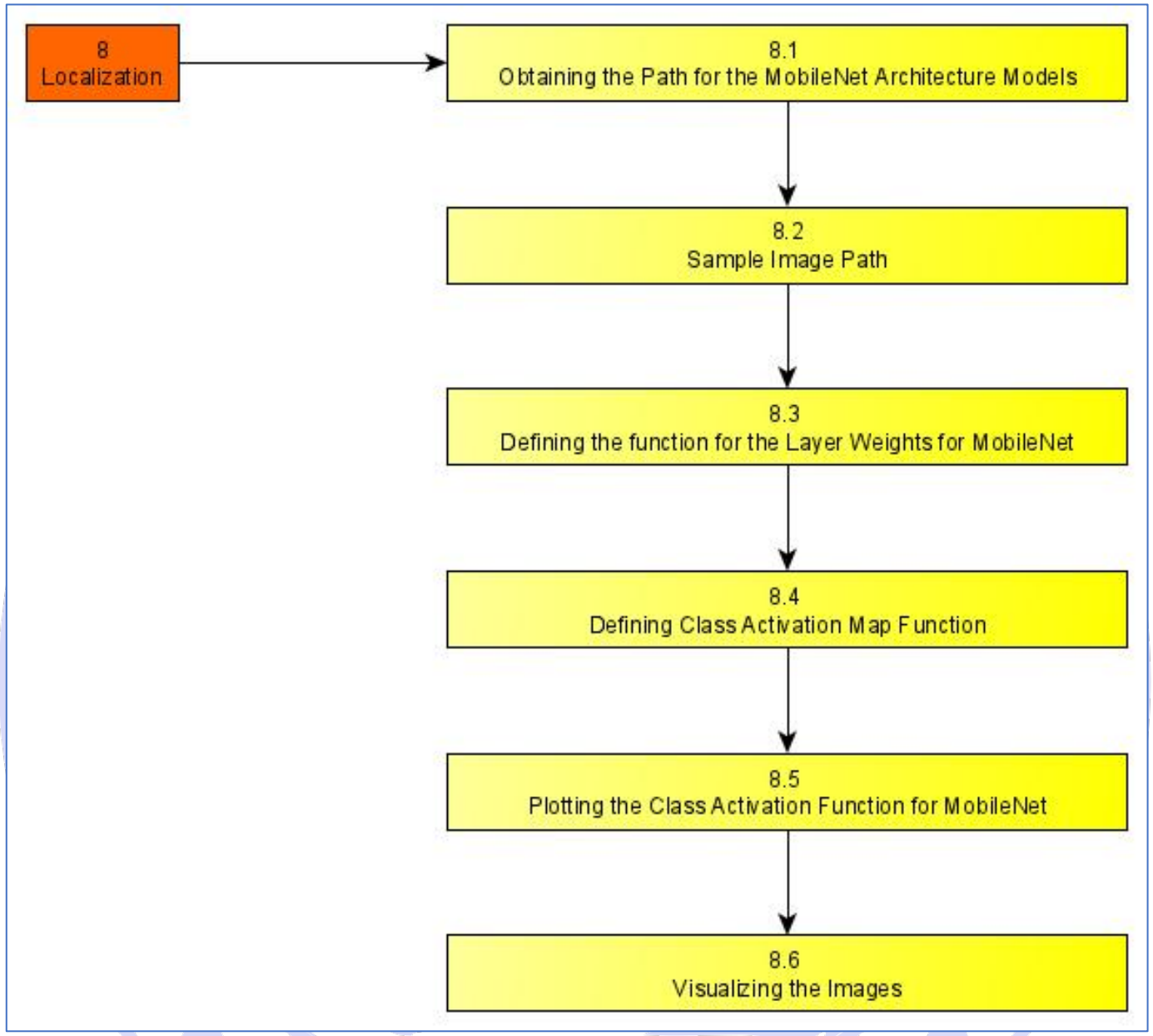


Figure 14: Proposed Architecture - Localization

For this, we define the Class Activation Map and then we create another function using which we can display the results obtained on a particular file, as we can use the address of the that image directly and then plot to even check the results obtained for the model which we have created. The results are displayed along with the plot which is created containing the Image and another heatmap image displayed along.

SAVING THE COMPLETE MODEL FOR THE PYTHON INTERFACE APPLICATION

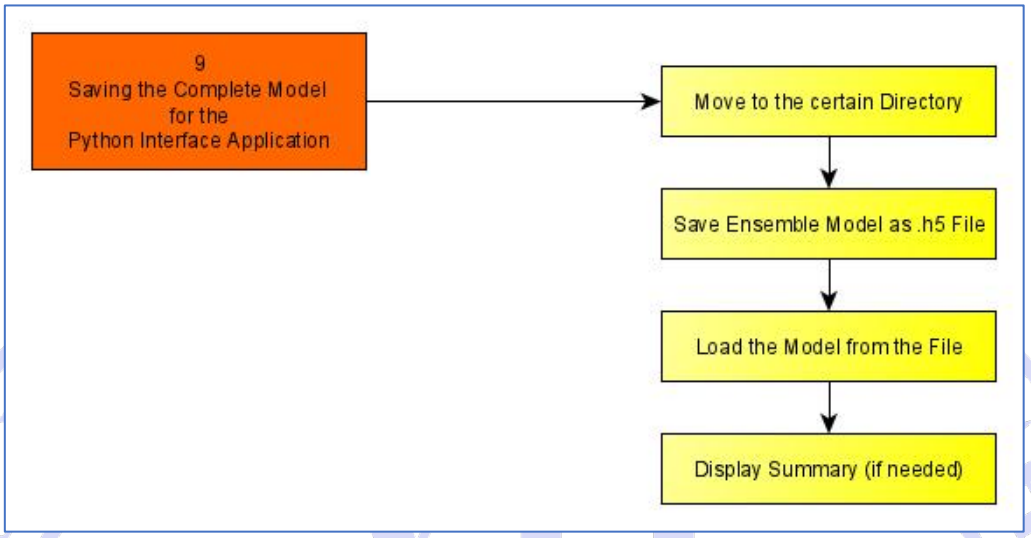


Figure 15: Proposed Architecture - Saving the Complete Model for the Application Interface

First, we will move to the required directory where we can save the file for the model. Then, we will use the complete model and use the save() function.

CONVERTING THE .H5 FILE TO .TFLITE FILE FOR THE PYTHON INTERFACE APPLICATION

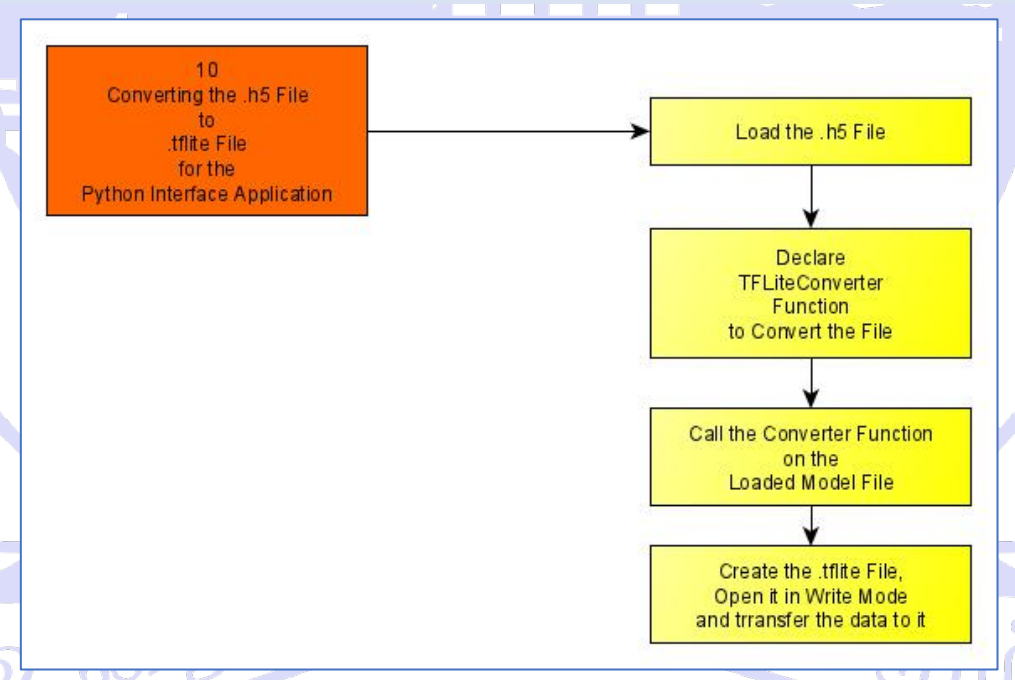


Figure 16: Converting the H5 File to TFLITE File for Application Interface

We will first load the saved model file and then we will use the converter function to convert the file into the into the required format which we can use for the Interface Application.

PROPOSED ARCHITECTURE SPECIFICATIONS

DATA SELECTION

The goal of the challenge is to help participants develop image analysis tools to enable the automated diagnosis of melanoma from dermoscopic images. Image analysis of skin lesions is composed of 3 parts:

- Part 1: Lesion Segmentation
- Part 2: Detection and Localization of Visual Dermoscopic Features/Patterns
- Part 3: Disease Classification

This challenge provides training data (150 images) and blind held-out test dataset (~600 images) will be provided for participants to generate and submit automated results.

BACKGRAOUND

MELANOMA

Skin cancer is a major public health problem, with over 5 million newly diagnosed cases in the United States each year. Melanoma is the deadliest form of skin cancer, responsible for over 9,000 deaths each year.

DERMOSCOPY

As pigmented lesions occurring on the surface of the skin, melanoma is amenable to early detection by expert visual inspection. It is also amenable to automated detection with image analysis. Given the widespread availability of high-resolution cameras, algorithms that can improve our ability to screen and detect troublesome lesions can be of great value. As a result, many centres have begun their own research efforts on automated analysis. However, a centralized, coordinated, and comparative effort across institutions has yet to be implemented.

Dermoscopy is an imaging technique that eliminates the surface reflection of skin. By removing surface reflection, visualization of deeper levels of skin is enhanced. Prior research has shown that when used by expert dermatologists, dermoscopy provides improved diagnostic accuracy, in comparison to standard photography. As inexpensive consumer dermatoscope attachments for smart phones are beginning to reach the market, the opportunity for automated dermoscopic assessment algorithms to positively influence patient care increases.

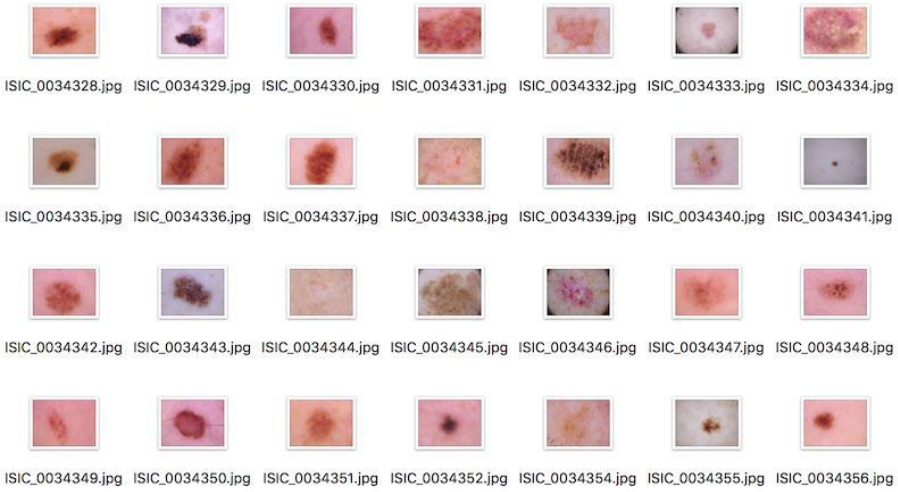


Figure 17: Sample Images of the Dataset

EXPLORATORY DATA ANALYSIS

CHECKING THE TYPES OF DATA

There are four types of EDA in all:

- **Univariate non-graphical**: This is the simplest form of data analysis among the four options. In this type of analysis, the data that is being analyzed consists of just a single variable. The main purpose of this analysis is to describe the data and to find patterns.
- **Univariate graphical**: Unlike the non-graphical method, the graphical method provides the full picture of the data. The three main methods of analysis under this type are histogram, stem and leaf plot, and box plots. The histogram represents the total count of cases for a range of values. Along with the data values, the stem and leaf plot show the shape of the distribution. The box plots graphically depict a summary of minimum, first quartile median, third quartile, and maximum.
- **Multivariate non-graphical**: The multivariate non-graphical type of EDA generally depicts the relationship between multiple variables of data through cross-tabulation or statistics.
- **Multivariate graphical**: This type of EDA displays the relationship between two or more set of data. A bar chart, where each group represents a level of one of the variables and each bar within the group represents levels of other variables.

FINDING THE OUTLIERS

There exist three different options on how to treat non-error outliers:

1. Keep
2. Delete
3. Recode

KEEP

When most of the detected outliers are non-error outliers and rightfully belong to the population of interest.

DELETE

The most straightforward option is to delete any outlying observation. However, this strategy bears a high risk of losing information. Especially if you find many outlying data points, try to avoid this. Also, deleting interesting and influential outliers (points that belong to the population of interest) can falsely impact any output, e.g., prediction or test result, you aim to achieve.

RECODE

Recoding outliers is a good option to treat outliers and keep as much information as possible simultaneously. This option should always be accompanied by sound reasoning and explanation.

DATA VISUALIZATION

- Univariate data analysis
- Bivariate data analysis
- Line plot
- Box plot
- Scatter matrix

DATA PRE-PROCESSING

SPLITTING THE DATA

Divide the dataset into three parts to avoid overfitting or underfitting and model selection bias called -

1. Training set (Has to be the largest set)
2. Cross-Validation set or Development set or Dev set
3. Testing Set

UNDERFITTING:

Underfitting mainly occurs when a machine learning algorithm is not able to capture the lower trend of data which is mainly when data is nor well fitted inside the model.

OVERFITTING?

When machine learning algorithm is trained on very well data and very closely on a dataset which can lead to a negative impact on the performance of the system leading to the wrong system and prediction model.

CHECKING FOR MISSING VALUES

There are three categories of missing data:

- **MCAR** (Missing Completely At Random) where the pattern of missingness is statistically independent of the data record. Example: you have a data set on a piece of paper and you spill coffee on the paper destroying part of the data.

- **MAR** (Missing At Random) where the probability distribution of the pattern of missingness is functionally dependent upon the observable component in the record. MCAR is a special case of MAR. Example: if a child does not attend an educational assessment because the child is (genuinely) ill, this might be predictable from other data we have about the child's health, but it would not be related to what we would have measured had the child not been ill.
- **MNAR** (Missing Not at Random) which is defined as the case which is NOT MAR, or when the missingness is specifically related to what is missing. Example: a person does not attend a drug test because the person took drugs the night before.

CHECKING CATEGORICAL FEATURES

Two major types of categorical features are

- **Nominal** – These are variables which are not related to each other in any order such as colour (black, blue, green).
- **Ordinal** – These are variables where a certain order can be found between them such as student grades (A, B, C, D, Fail).

NORMALIZING DATASET

The goal of normalization is to change values to a common scale without distorting the difference between the range of values.

Using this technique, we are going to have a mean of 0 and a standard deviation of 1 in our dataset. We can either do it normally by combining different functions in numpy, i.e.,

$$z = \frac{x.values - np.mean(x.values)}{np.std(x.values)}$$

where *x* is a data frame with all numerical indices. If we want to retain the values in a data frame, then we can simply remove. values in front of it.

FEATURE TRANSFORMATION

WHY DO WE NEED FEATURE TRANSFORMATION AND SCALING?

Oftentimes, we have datasets in which different columns have different units – like one column can be in kilograms, while another column can be in centimetres. Furthermore, we can have columns like income which can range from 20,000 to 100,000, and even more; while a benign column which can range from 0 to 100(at the most). Thus, Malignant is about 1,000 times larger than age. When we feed these features to the model as is, there is every chance that the Malignant values will influence the result more due to its larger value. But this doesn't necessarily mean it is more important as a predictor. So, to give importance to both Benign, and Malignant, we need feature scaling.

FEATURE TRANSFORMATIONS USED IN THE MODELS

MODEL SELECTION

Tensors often offer more natural representations of data, e.g., consider video, which consists of obviously correlated images over time. You *can* turn this into a matrix, but it's just not natural or intuitive (what does a factorization of some matrix-representation of video mean?).

Tensors are trending for several reasons:

- our understanding of multilinear algebra is improving rapidly, specifically in various types of factorizations, which in turn helps us to identify new potential applications (e.g., multiway component analysis)
- software tools are emerging (e.g., Tensorlab) and are being welcomed
- Big Data applications can often be solved using tensors, for example recommender systems, and Big Data itself is hot
- increases in computational power, as some tensor operations can be hefty (this is also one of the major reasons why deep learning is so popular now)

MODEL TRAINING

Most machines cannot learn until and unless they have a proper dataset or at least any kind of **data**. And modern data is often multi-dimensional (having more than one dimensions). Tensors can play an important role in Machine Learning/Deep Learning by encoding multi-dimensional data. For example (a simple example to explain the advantages of using them), a picture is generally represented by three fields: *width, height* and *depth (color)*. It makes total sense to encode it as a 3D tensor. However, more than often we are dealing with tens of thousands of pictures. Hence this is where the fourth field, *sample size* comes into play. A series of images, such as the famous MNIST dataset, can be easily stored in a 4D tensor in Tensorflow. This representation allows problems involving big data to be solved easily.

(ML Concepts: Tensors, 2018)

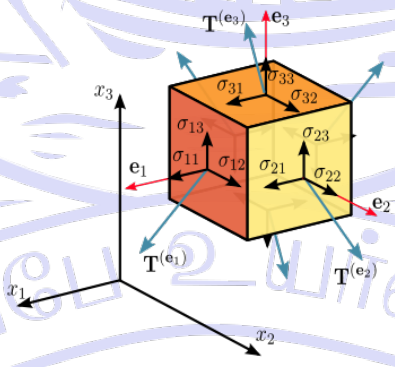


Figure 18: Visualizing a Tensor

A real-life example would be in most recommender systems, model-based Collaborative Filtering approaches such as Matrix Factorization do not the flexibility of integrating context information into the models. By using a **N-dimensional tensor** instead of the traditional 2D User-Item matrix, researchers have developed a more

robust model that provides context-aware recommendations. Increase in computing power in the recent years also allows these heavy tensor operations to be realized.

At this point you may wonder: why do most machine learning algorithms rely on vectors and matrices, while deep learning algorithms and neural networks mostly rely on tensors? A simple answer is that deep learning usually involves hundreds, if not thousands, of dimensions and fields. As we discussed previously, this is best represented by tensors since they can represent anything ranging from *zero* to *N* dimensions. In computer vision problems such as the Merck Molecular Activity Challenge, images can easily be broken down into a few hundred features. Therefore, tensors can be best viewed as containers that wrap and store data features in the context of machine learning and deep learning.

PROJECT DEMONSTRATION

APPLICATION INTERFACE

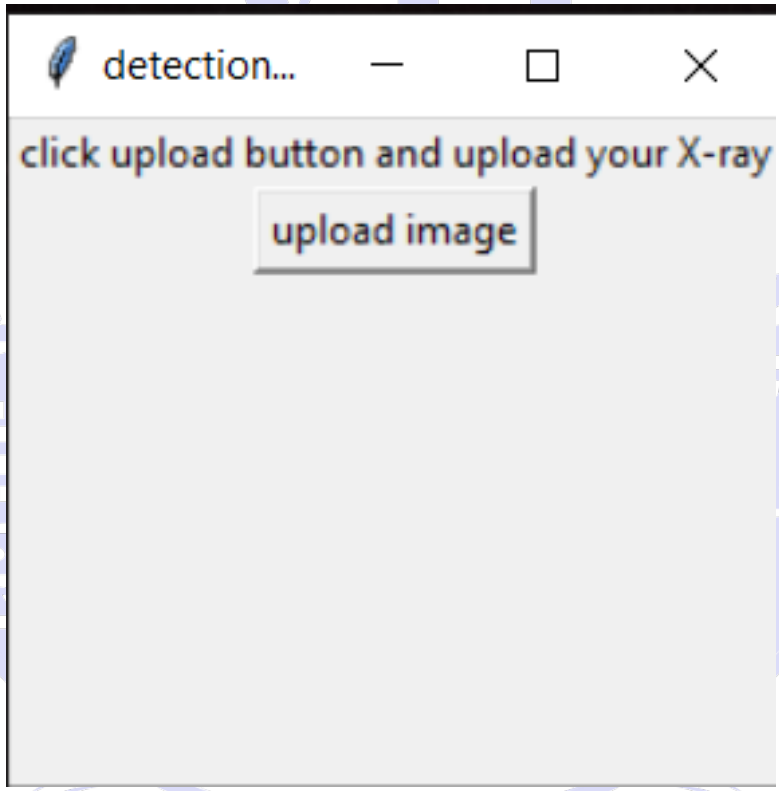


Figure 19: First View of Interface

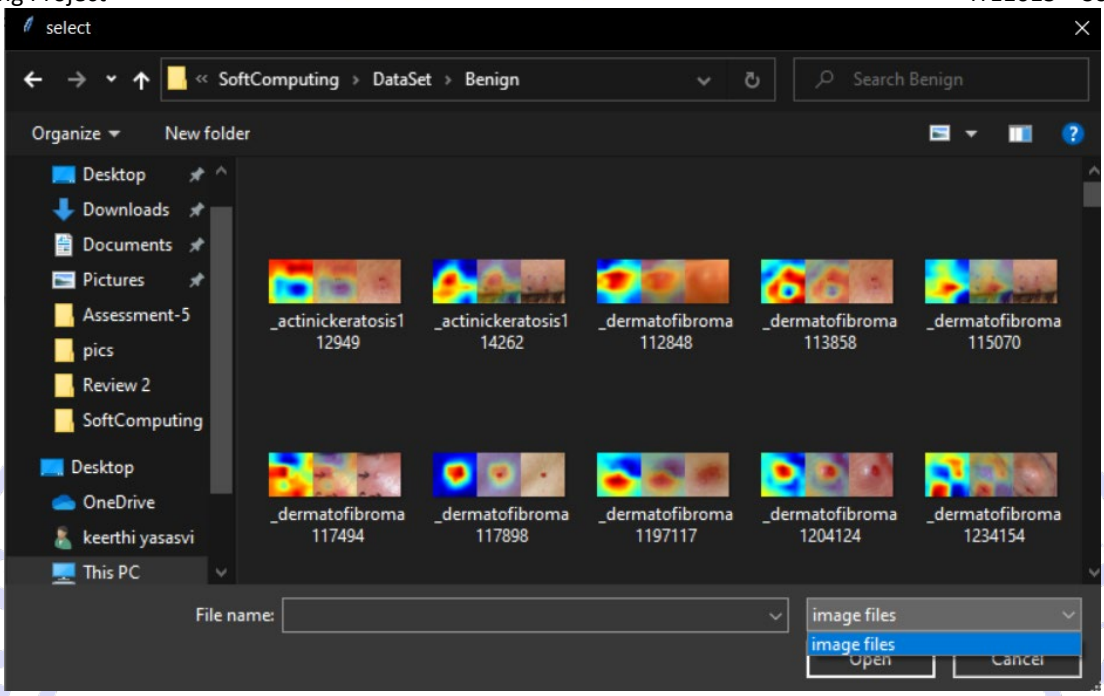


Figure 20: Selecting a File for Uploading

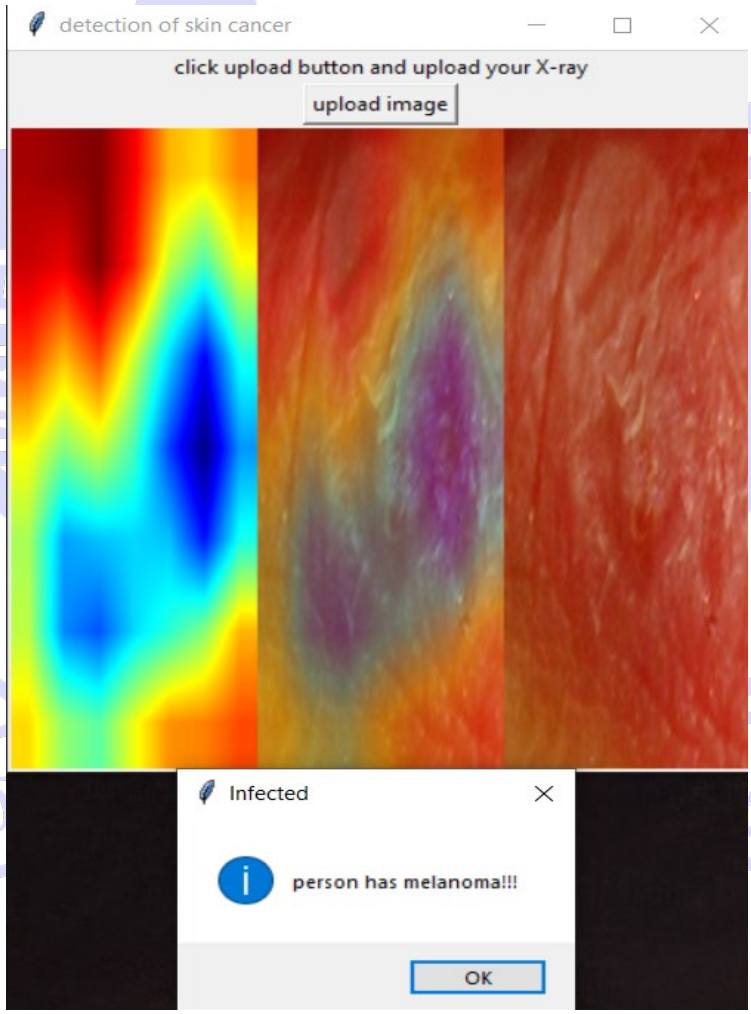


Figure 21: Display of Output

MOUNTING GOOGLE DRIVE

1.2.1. Mounting Google Drive

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

```
Mounted at /content/drive
```

Figure 22: Project Demonstration - Mounting Google Drive for Authorization

LOADING AND DISPLAY THE CSV DATASET

TRAINING CSV FILE

1.2.2.2 Display the Training Data of the CSV File

```
training_data
```

	image_id	melanoma	seborrheic_keratosis
0	ISIC_0000000	0.0	0.0
1	ISIC_0000001	0.0	0.0
2	ISIC_0000002	1.0	0.0
3	ISIC_0000003	0.0	0.0
4	ISIC_0000004	1.0	0.0
...	...	...	...
1995	ISIC_0015220	0.0	1.0
1996	ISIC_0015233	0.0	1.0
1997	ISIC_0015260	0.0	1.0
1998	ISIC_0015284	1.0	0.0
1999	ISIC_0015295	0.0	1.0

2000 rows × 3 columns

Figure 23: Project Demonstration - Data in CSV File for Training Data

VALIDATION CSV FILE

1.2.3.2 Display the Validation Data of the CSV File

```
validation_data
```

	image_id	melanoma	seborrheic_keratosis
0	ISIC_0001769	0.0	0.0
1	ISIC_0001852	0.0	0.0
2	ISIC_0001871	0.0	0.0
3	ISIC_0003462	0.0	0.0
4	ISIC_0003539	0.0	0.0
...	...	...	...
145	ISIC_0015443	0.0	0.0
146	ISIC_0015445	0.0	0.0
147	ISIC_0015483	0.0	0.0
148	ISIC_0015496	0.0	0.0
149	ISIC_0015627	0.0	0.0

150 rows × 3 columns

Figure 24: Project Demonstration - Data in CSV File for Validation Data

TESTING CSV FILE

1.2.4.2. Display the Testing Data of the CSV File

```
testing_data
```

	image_id	melanoma	seborrheic_keratosis
0	ISIC_0012086	0.0	1.0
1	ISIC_0012092	0.0	0.0
2	ISIC_0012095	0.0	0.0
3	ISIC_0012134	0.0	1.0
4	ISIC_0012136	0.0	1.0
...	...	...	...
595	ISIC_0016068	0.0	0.0
596	ISIC_0016069	0.0	0.0
597	ISIC_0016070	0.0	0.0
598	ISIC_0016071	0.0	0.0
599	ISIC_0016072	0.0	0.0

600 rows × 3 columns

Figure 25: Project Demonstration - Data in CSV File for Testing Data

CONVERTING THE DATA INTO TENSORS

TRAINING TENSOR

1.3.1.4. Converting the Complete Filenames to Tensors

```
train_tensors = paths_to_tensor(files).astype('float32')/255
```

```
100%|██████████| 2000/2000 [19:09<00:00,  1.74it/s]
```

Figure 26: Project Demonstration - Converting Training Files to Tensors

VALIDATION TENSOR

1.3.2.4. Converting the Complete Filenames to Tensors

```
valid_tensors = paths_to_tensor(validation_files).astype('float32')/255
```

```
100%|██████████| 150/150 [01:53<00:00,  1.32it/s]
```

Figure 27: Project Demonstration - Converting Validation Files to Tensors

TESTING TENSOR

1.3.3.4. Converting the Complete Filenames to Tensors

```
test_tensors = paths_to_tensor(test_files).astype('float32')/255
```

```
100%|██████████| 600/600 [10:38<00:00,  1.06s/it]
```

Figure 28: Project Demonstration - Converting Testing Files to Tensors

FITTING INTO THE MODELS

2.1.3. Fitting into the Model

```
mobilenet_model.fit(train_tensors,
                    train_targets,
                    batch_size = 8,
                    validation_data = (valid_tensors, valid_targets),
                    epochs = 5,
                    callbacks=[checkpointer],
                    verbose=1)
```

```
Epoch 1/5
250/250 [==============================] - 84s 208ms/step - loss: 0.5928 - accuracy: 0.7956 - val_loss: 1.5225 - val_accuracy:
0.2000

Epoch 00001: val_loss improved from inf to 1.52253, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organize
d/Saved Models/weights.best.mobilenet.hdf5
Epoch 2/5
250/250 [==============================] - 50s 200ms/step - loss: 0.5103 - accuracy: 0.8002 - val_loss: 0.5847 - val_accuracy:
0.8000

Epoch 00002: val_loss improved from 1.52253 to 0.58473, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Orga
nized/Saved Models/weights.best.mobilenet.hdf5
Epoch 3/5
250/250 [==============================] - 50s 201ms/step - loss: 0.5342 - accuracy: 0.7867 - val_loss: 0.6205 - val_accuracy:
0.8000

Epoch 00003: val_loss did not improve from 0.58473
Epoch 4/5
250/250 [==============================] - 50s 200ms/step - loss: 0.5311 - accuracy: 0.7961 - val_loss: 0.5127 - val_accuracy:
0.8000

Epoch 00004: val_loss improved from 0.58473 to 0.51270, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Orga
nized/Saved Models/weights.best.mobilenet.hdf5
Epoch 5/5
250/250 [==============================] - 50s 200ms/step - loss: 0.5047 - accuracy: 0.8116 - val_loss: 0.6225 - val_accuracy:
0.8000

Epoch 00005: val_loss did not improve from 0.51270

<tensorflow.python.keras.callbacks.History at 0x7fa500592d90>
```

Figure 29: Project Demonstration - Training the MobileNet Architecture

2.2.3 Fitting into the Model

```
inception_model.fit(train_tensors,
                    train_targets,
                    batch_size = 8,
                    validation_data = (valid_tensors, valid_targets),
                    epochs = 5,
                    callbacks=[checkpointer],
                    verbose=1)
```

```
Epoch 1/5
250/250 [==============================] - 60s 213ms/step - loss: 0.5970 - accuracy: 0.7834 - val_loss: 0.5039 - val_accuracy:
0.8000

Epoch 00001: val_loss improved from inf to 0.50388, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organize
d/Saved Models/weights.best.InceptionV3.hdf5
Epoch 2/5
250/250 [==============================] - 52s 207ms/step - loss: 0.4849 - accuracy: 0.8230 - val_loss: 0.8663 - val_accuracy:
0.7800

Epoch 00002: val_loss did not improve from 0.50388
Epoch 3/5
250/250 [==============================] - 52s 207ms/step - loss: 0.5306 - accuracy: 0.7890 - val_loss: 0.5700 - val_accuracy:
0.8000

Epoch 00003: val_loss did not improve from 0.50388
Epoch 4/5
250/250 [==============================] - 52s 207ms/step - loss: 0.5063 - accuracy: 0.8151 - val_loss: 0.5268 - val_accuracy:
0.8000

Epoch 00004: val_loss did not improve from 0.50388
Epoch 5/5
250/250 [==============================] - 52s 207ms/step - loss: 0.4957 - accuracy: 0.8166 - val_loss: 0.5712 - val_accuracy:
0.8000

Epoch 00005: val_loss did not improve from 0.50388

<tensorflow.python.keras.callbacks.History at 0x7fa4ac069690>
```

Figure 30: Project Demonstration - Training the Inception Architecture

2.3.3 Fitting into the Model

```
xception_model.fit(train_tensors,
                   train_targets,
                   batch_size = 8,
                   validation_data = (valid_tensors, valid_targets),
                   epochs = 2,
                   callbacks=[checkpointer, tensor_board],
                   verbose=1)
```

```
Epoch 1/2
250/250 [==============================] - 145s 565ms/step - loss: 0.7223 - accuracy: 0.7660 - val_loss: 0.5035 - val_accuracy:
0.8000

Epoch 00001: val_loss improved from inf to 0.50352, saving model to /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organize
d/Saved Models/xception_weights.hdf5
Epoch 2/2
250/250 [==============================] - 140s 558ms/step - loss: 0.5142 - accuracy: 0.8096 - val_loss: 8.2784 - val_accuracy:
0.6800

Epoch 00002: val_loss did not improve from 0.50352

<tensorflow.python.keras.callbacks.History at 0x7fa276ed2f10>
```

Figure 31: Project Demonstration - Training the Xception Architecture

PREDICTIONS ON A SAMPLE IMAGE

3.1.3 Predicting for a Sample Image Using MobileNet

```
predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg",
```

```
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.
jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7fa2740562d0>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
Model Weights:
None
Prediction... Melanoma :  0.86639947  | Other :  0.13360052

[1.0, 0.0]
```

Figure 32: Project Demonstration - Predicting on a Sample Image using MobileNet

3.2.2 Predicting for a Sample Image Using InceptionV3

```
predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg",
```

```
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.
jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7fa173ac0950>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
Model Weights:
None
Prediction... Melanoma :  0.8329839  | Other :  0.1670161

[1.0, 0.0]
```

Figure 33: Project Demonstration - Predicting on a Sample Image using Inception

3.3.2 Predicting a Sample Image using Xception

```
predict("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.jpg",
```

```
Image Path: /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/ISIC-2017_Validation_Data/Data Image JPG/ISIC_0001769.
jpg
Arhitecture Used:
<tensorflow.python.keras.engine.functional.Functional object at 0x7fa2741a2650>
Path for Model Weights:
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Model Weights:
None
Prediction... Melanoma :  0.7678717  | Other :  0.23212835

[1.0, 0.0]
```

Figure 34: Project Demonstration - Predicting on a Sample Image using Xception

EVALUATION OF MOBILENET ON VALIDATION DATA

4.1.5.1 Defining the Confusion Matrix Function

```
def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0

    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1

    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
```

```
TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet, TRUE_NEGATIVE_MobileNet, FALSE_NEGATIVE_MobileNet = positive_negative_measurer
postives_negatives_MobileNet = [[TRUE_POSITIVE_MobileNet, FALSE_POSITIVE_MobileNet],
                                [FALSE_NEGATIVE_MobileNet, TRUE_NEGATIVE_MobileNet]]
```

```
postives_negatives_MobileNet
```

```
[[120, 30], [0, 0]]
```

Figure 35: Project Demonstration - Evaluation of MobileNet on Training Data - Confusion Matrix

4.1.5.2 Obtaining Labels

```
import seaborn as sns
sns.set()
labels_MobileNet =  np.array([['True positive: ' + str(TRUE_POSITIVE_MobileNet),
                               'False positive: ' + str(FALSE_POSITIVE_MobileNet)],
                              ['False negative: ' + str(FALSE_NEGATIVE_MobileNet),
                               'True negative: ' + str(TRUE_NEGATIVE_MobileNet)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_MobileNet, annot = labels_MobileNet, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa17161d410>
```

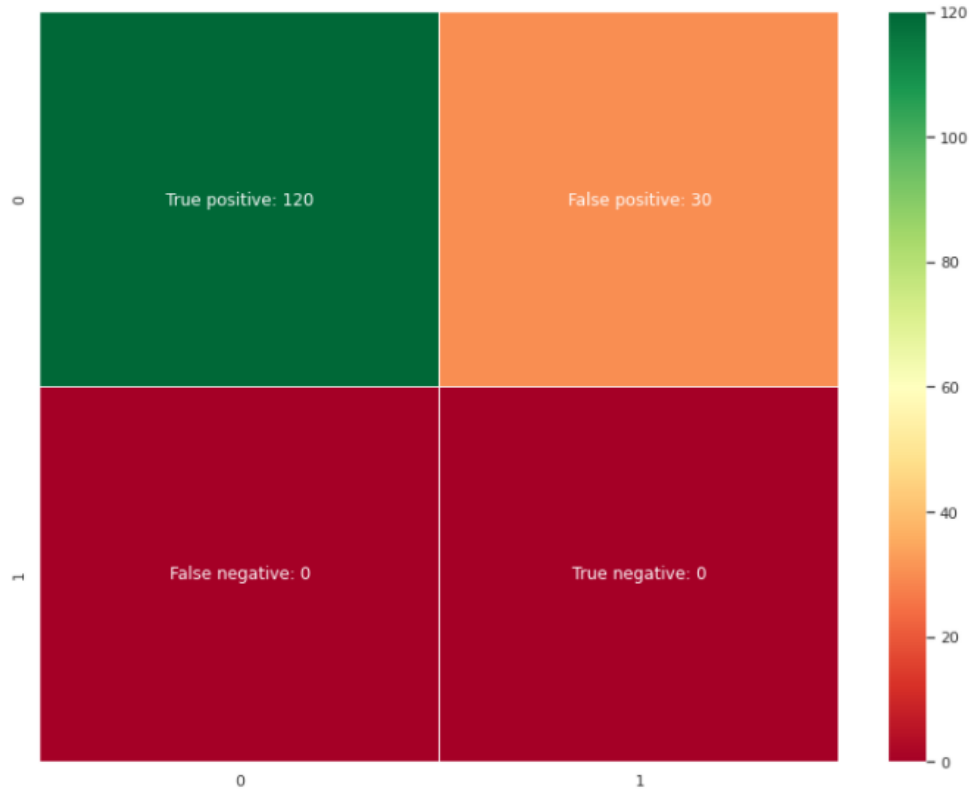


Figure 36: Project Demonstration - Evaluation of MobileNet on Training Data - Obtaining Labels

```
labels_MobileNet
```

```
array([['True positive: 120', 'False positive: 30'],
       ['False negative: 0', 'True negative: 0']], dtype='<U18')
```

4.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
print("Sensitivity: ", sensitivity_MobileNet)
```

```
Sensitivity:  1.0
```

4.1.5.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
    print("Specifity: ", specifity_MobileNet)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

4.1.5.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_MobileNet = TRUE_POSITIVE_MobileNet / (TRUE_POSITIVE_MobileNet + FALSE_POSITIVE_MobileNet)
print("Precision: ", predcision_MobileNet)
```

```
Precision:  0.8
```

Figure 37: Project Demonstration - Evaluation of MobileNet on Training Data(i)

4.1.5.6 Calculating Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_MobileNet = TRUE_NEGATIVE_MobileNet / (TRUE_NEGATIVE_MobileNet + FALSE_NEGATIVE_MobileNet)
    print("Negative predictive value: ", npv_MobileNet)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

4.1.5.7 Calculating Accuracy

```
# Accuracy
accuracy_MobileNet = (TRUE_POSITIVE_MobileNet + TRUE_NEGATIVE_MobileNet) / (TRUE_POSITIVE_MobileNet + FALSE_POSITIVE_MobileNet +
print("Accuracy: ", accuracy_MobileNet)
```

```
Accuracy:  0.8
```

Figure 38: Project Demonstration - Evaluation of MobileNet on Training Data(ii)

EVALUATION OF INCEPTION ON VALIDATION DATA

```
import seaborn as sns
sns.set()
labels_Inception =  np.array([['True positive: ' + str(TRUE_POSITIVE_Inception),
                     'False positive: ' + str(FALSE_POSITIVE_Inception)],
                    ['False negative: ' + str(FALSE_NEGATIVE_Inception),
                     'True negative: ' + str(TRUE_POSITIVE_Inception)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Inception, annot = labels_Inception, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa16660ec10>
```

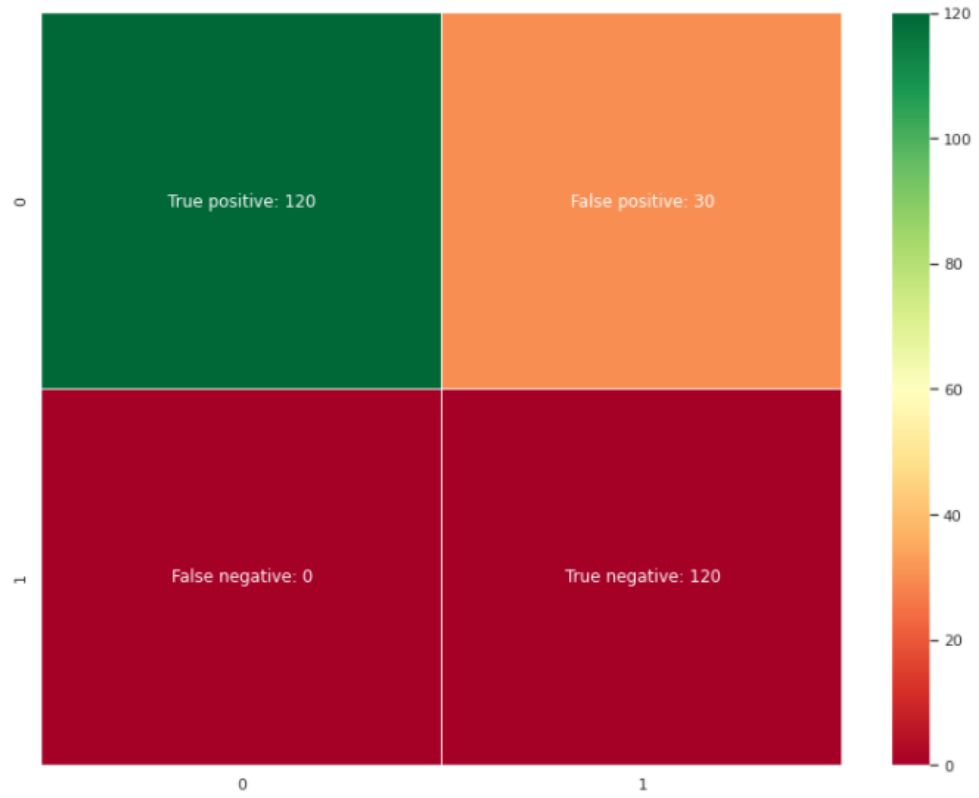


Figure 39: Project Demonstration - Evaluation of Inception on Training Data - Confusion Matrix

4.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception + FALSE_NEGATIVE_Inception)
print("Sensitivity: ", sensitivity_Inception)
```

```
Sensitivity:  1.0
```

4.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception + FALSE_NEGATIVE_Inception)
    print("Specifity: ", specifity_Inception)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

4.2.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Inception = TRUE_POSITIVE_Inception / (TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception)
print("Precision: ", predcision_Inception)
```

```
Precision:  0.8
```

Figure 40: Project Demonstration - Evaluation of Inception on Training Data(i)

4.2.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Inception = TRUE_NEGATIVE_Inception / (TRUE_NEGATIVE_Inception + FALSE_NEGATIVE_Inception)
    print("Negative predictive value: ", npv_Inception)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

4.2.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Inception = (TRUE_POSITIVE_Inception + TRUE_NEGATIVE_Inception) / (TRUE_POSITIVE_Inception + FALSE_POSITIVE_Inception +
print("Accuracy: ", accuracy_Inception)
```

```
Accuracy:  0.8
```

Figure 41: Project Demonstration - Evaluation of Inception on Training Data(ii)

EVALUATION OF XCEPTION ON VALIDATION DATA

4.3.2.4.2 Obtaining the Labels

```
import seaborn as sns
sns.set()
labels_Xception =  np.array([['True positive: ' + str(TRUE_POSITIVE_Xception),
                      'False positive: ' + str(FALSE_POSITIVE_Xception)],
                     ['False negative: ' + str(FALSE_NEGATIVE_Xception),
                      'True negative: ' + str(TRUE_POSITIVE_Xception)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Xception, annot = labels_Xception, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1665d8610>
```

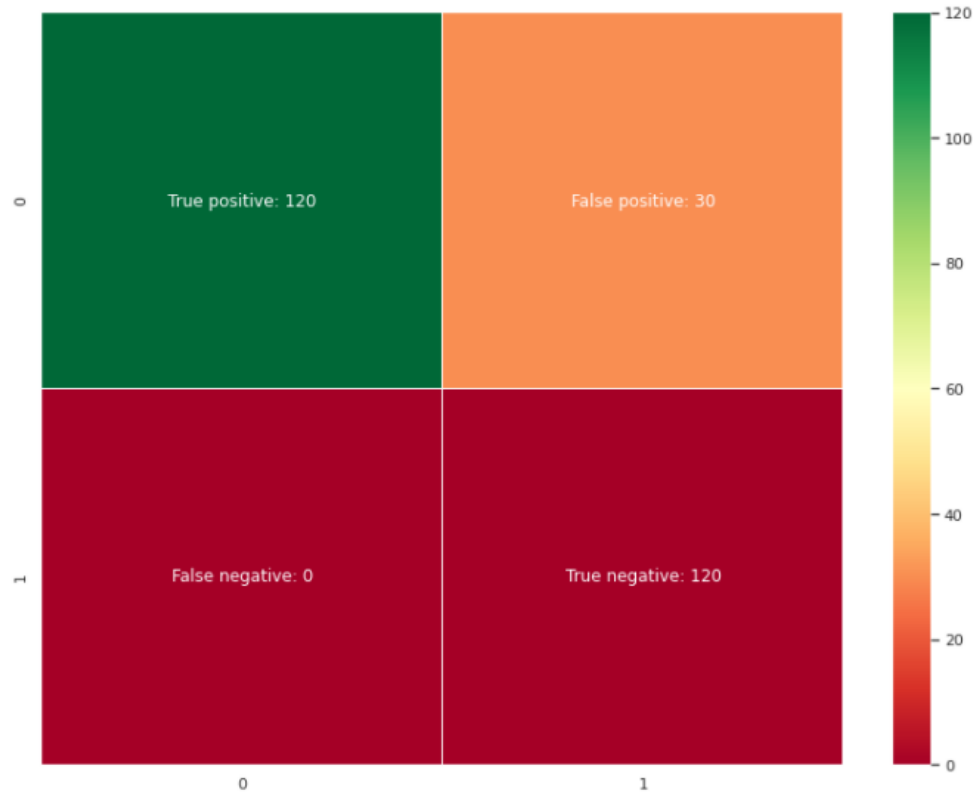


Figure 42: Project Demonstration - Evaluation of Xception on Training Data - Confusion Matrix

4.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception + FALSE_NEGATIVE_Xception)
print("Sensitivity: ", sensitivity_Xception)
```

```
Sensitivity:  1.0
```

4.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception + FALSE_NEGATIVE_Xception)
    print("Specifity: ", specifity_Xception)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

4.3.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Xception = TRUE_POSITIVE_Xception / (TRUE_POSITIVE_Xception + FALSE_POSITIVE_Xception)
print("Precision: ", predcision_Xception)
```

```
Precision:  0.8
```

Figure 43: Project Demonstration - Evaluation of Xception on Training Data(i)

4.3.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Xception = TRUE_NEGATIVE_Xception / (TRUE_NEGATIVE_Xception + FALSE_NEGATIVE_Xception)
    print("Negative predictive value: ", npv_Xception)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

4.3.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Xception = (TRUE_POSITIVE_Xception + TRUE_NEGATIVE_Xception) / (TRUE_POSITIVE_Xception + FALSE_POSITIVE_Xception + TRUE_
print("Accuracy: ", accuracy_Xception)
```

```
Accuracy:  0.8
```

Figure 44: Project Demonstration - Evaluation of Xception on Training Data (ii)

5.4 Defining the Ensembling Function

```
def ensemble(models, model_input):
    outputs = [model.outputs[0] for model in models]
    print("Outputs: ")
    print(outputs)
    y = keras.layers.Average()(outputs)
    print("y: ")
    print(y)
    model = Model(model_input, y, name='ensemble')
    print("Model: ")
    print(model)
    return model
```

```
# Getting ensemble model
ensemble_model = ensemble(models, model_input)
```

```
Outputs:
[<KerasTensor: shape=(None, 2) dtype=float32 (created by layer 'dense_9')>, <KerasTensor: shape=(None, 2) dtype=float32 (create
d by layer 'dense_10')>, <KerasTensor: shape=(None, 2) dtype=float32 (created by layer 'dense_11')>]
y:
KerasTensor(type_spec=TensorSpec(shape=(None, 2), dtype=tf.float32, name=None), name='average/truediv:0', description="created
by layer 'average'")
Model:
<tensorflow.python.keras.engine.functional.Functional object at 0x7fa1660b9910>
```

Figure 45: Project Demonstration - Evaluation of Models Together on Validation Data - Defining Ensemble Function

```
# load all models into memory
members = load_all_models(weight_file_names_with_path)
print('Loaded %d models' % len(members))
```

```
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.mobilenet.hdf5
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/weights.best.InceptionV3.hdf5
>loaded /content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/xception_weights.hdf5
Loaded 3 models
```

Figure 46: Project Demonstration - Evaluation of Models Together on Validation Data - Loading all Model Weights

```
weights_of_MobileNet_Inception_and_Xception = c = np.concatenate((weights_of_MobileNet_and_Inception, weights_of_Xception[0])) #
print(type(weights_of_MobileNet_Inception_and_Xception))
print(weights_of_MobileNet_Inception_and_Xception.shape)
```

```
<class 'numpy.ndarray'>
(9, 3, 3, 32)
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
```

```
# serialize model to JSON
model_weights_for_json = ensemble_model.to_json()
with open("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_model.json", "w") as
    json_file.write(model_weights_for_json)
# serialize weights to HDF5
ensemble_model.save_weights("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Saved Models/Ensemble Model/ensemble_mo
print("Saved model to disk")
```

```
Saved model to disk
```

Figure 47: Project Demonstration - Evaluation of Models Together on Validation Data - Serializing Weights into one

In [ ]:
```
# Compute test set predictions
NUMBER_TEST_SAMPLES_Ensemble = 150

y_true_Ensemble = valid_targets[:NUMBER_TEST_SAMPLES_Ensemble]
y_score_Ensemble = []
for index in range(NUMBER_TEST_SAMPLES_Ensemble): #compute one at a time due to memory constraints
    image_to_predict_Ensemble = path_to_tensor(validation_files[index]).astype("float32")/255.
    probs_Ensemble = ensemble_model.predict(image_to_predict_Ensemble,)
    if np.argmax(probs_Ensemble) == 0:
        y_score_Ensemble.append([1., 0.])
    elif np.argmax(probs_Ensemble) == 1:
        y_score_Ensemble.append([0., 1.])
    print("Predicted value {}... ".format(index+1) + " Melanoma : ", probs_Ensemble[0][0],  " | Other : ", probs_Ensemble[0][1])
    print("Real values {}...".format(index+1) + "      Melanoma : ", valid_targets[index][0], "      | Other : ", valid_targets[i
    print("-------------------------------------------------------------------------------")


correct_Ensemble = np.array(y_true_Ensemble) == np.array(y_score_Ensemble)
```

```
Predicted value 1...  Melanoma :  0.8269514  | Other :  0.17304868
Real values 1...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 2...  Melanoma :  0.8224184  | Other :  0.17758167
Real values 2...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 3...  Melanoma :  0.83412284  | Other :  0.16587715
Real values 3...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 4...  Melanoma :  0.82447284  | Other :  0.17552719
Real values 4...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 5...  Melanoma :  0.8221907  | Other :  0.1778094
Real values 5...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 6...  Melanoma :  0.83770597  | Other :  0.16229409
Real values 6...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
Predicted value 7...  Melanoma :  0.82638013  | Other :  0.17361985
Real values 7...      Melanoma :  1.0       | Other :  0.0
-------------------------------------------------------------------------------
```

In [ ]:
```
print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble)*100))
```

```
Accuracy = 80.00%
```

In [ ]:
```
image_to_predict_Ensemble = path_to_tensor("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Output_melanoma/ISIC_006
ensemble_model.predict(image_to_predict_Ensemble)
```

Out[162]:
```
array([[0.8248186 , 0.17518133]], dtype=float32)
```

Figure 48: Project Demonstration - Evaluation of Models Together on Validation Data - Computing Test Set Predictions

5.5.2.4.2 Obtaining the Labels

```
import seaborn as sns
sns.set()
labels_Ensemble =  np.array([['True positive: ' + str(TRUE_POSITIVE_Ensemble),
                      'False positive: ' + str(FALSE_POSITIVE_Ensemble)],
                     ['False negative: ' + str(FALSE_NEGATIVE_Ensemble),
                      'True negative: ' + str(TRUE_POSITIVE_Ensemble)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Ensemble, annot = labels_Ensemble, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa164688950>
```

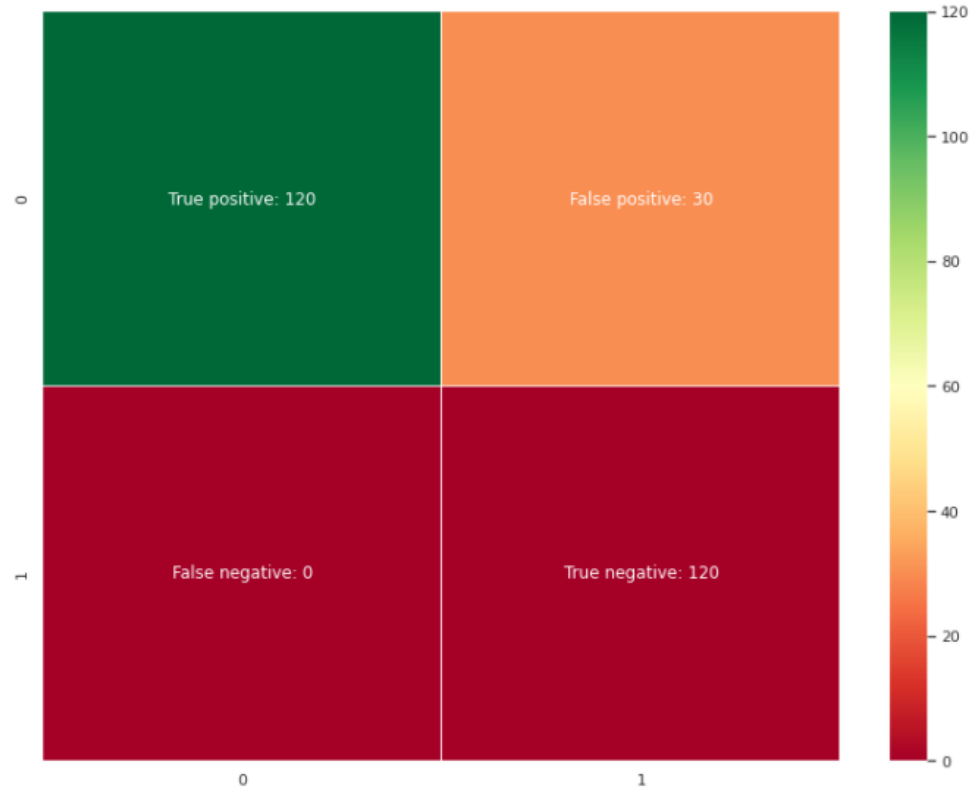


Figure 49: Project Demonstration - Evaluation of Models Together on Validation Data - Obtaining Labels

5.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
print("Sensitivity: ", sensitivity_Ensemble)
```

```
Sensitivity:  1.0
```

5.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
    print("Specifity: ", specifity_Ensemble)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

5.5.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Ensemble = TRUE_POSITIVE_Ensemble / (TRUE_POSITIVE_Ensemble + FALSE_POSITIVE_Ensemble)
print("Precision: ", predcision_Ensemble)
```

```
Precision:  0.8
```

Figure 50: Project Demonstration - Evaluation of Models Together on Validation Data(i)

5.5.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Ensemble = TRUE_NEGATIVE_Ensemble / (TRUE_NEGATIVE_Ensemble + FALSE_NEGATIVE_Ensemble)
    print("Negative predictive value: ", npv_Ensemble)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

5.5.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Ensemble = (TRUE_POSITIVE_Ensemble + TRUE_NEGATIVE_Ensemble) / (TRUE_POSITIVE_Ensemble + FALSE_POSITIVE_Ensemble + TRUE_
print("Accuracy: ", accuracy_Ensemble)
```

```
Accuracy:  0.8
```

Figure 51: Project Demonstration - Evaluation of Models Together on Validation Data(ii)

EVALUATING THE MODELS INDIVIDUALLY ON TESTING DATA

MOBILENET

6.1.5 Confusion Matrix

6.1.5.1 Defining the Confusion Matrix Function

```
def positive_negative_measurement(y_true, y_score):
    # Initialization
    TRUE_POSITIVE = 0
    FALSE_POSITIVE = 0
    TRUE_NEGATIVE = 0
    FALSE_NEGATIVE = 0

    # Calculating the model
    for i in range(len(y_score)):
        if y_true[i] == y_score[i] == 1:
            TRUE_POSITIVE += 1
        if (y_score[i] == 1) and (y_true[i] != y_score[i]):
            FALSE_POSITIVE += 1
        if y_true[i] == y_score[i] == 0:
            TRUE_NEGATIVE += 1
        if (y_score[i] == 0) and (y_true[i] != y_score[i]):
            FALSE_NEGATIVE += 1

    return(TRUE_POSITIVE, FALSE_POSITIVE, TRUE_NEGATIVE, FALSE_NEGATIVE)
```

```
TRUE_POSITIVE_MobileNet_Test, FALSE_POSITIVE_MobileNet_Test, TRUE_NEGATIVE_MobileNet_Test, FALSE_NEGATIVE_MobileNet_Test = positi
postives_negatives_MobileNet_Test = [[TRUE_POSITIVE_MobileNet_Test, FALSE_POSITIVE_MobileNet_Test],
                                     [FALSE_NEGATIVE_MobileNet_Test, TRUE_NEGATIVE_MobileNet_Test]]
```

```
postives_negatives_MobileNet_Test
```

```
[[483, 117], [0, 0]]
```

Figure 52: Project Demonstration - Evaluation of MobileNet Model Individually on Testing Data - Confusion Matrix

6.1.5.2 Obtaining Labels

```
import seaborn as sns
sns.set()
labels_MobileNet_Test =  np.array([['True positive: ' + str(TRUE_POSITIVE_MobileNet_Test),
                                    'False positive: ' + str(FALSE_POSITIVE_MobileNet_Test)],
                                   ['False negative: ' + str(FALSE_NEGATIVE_MobileNet_Test),
                                    'True negative: ' + str(TRUE_NEGATIVE_MobileNet_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_MobileNet_Test, annot = labels_MobileNet_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa2749d8fd0>
```

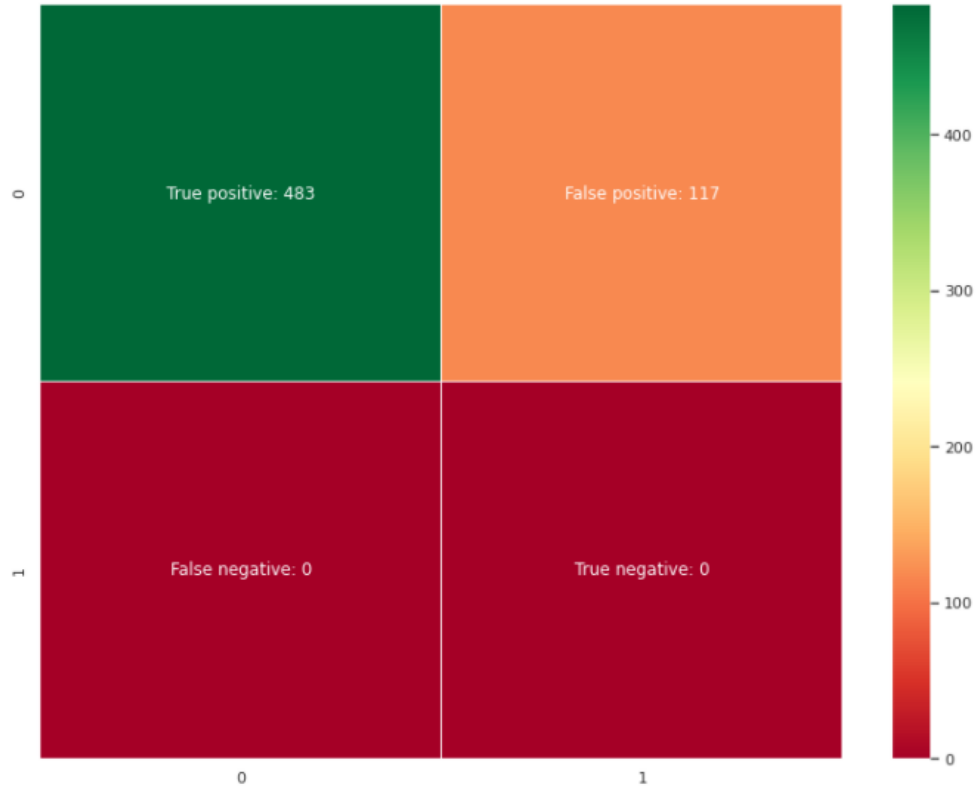


Figure 53: Project Demonstration - Evaluation of MobileNet Model Individually on Testing Data - Obtaining Labels

```
labels_MobileNet_Test
```

```
array([['True positive: 483', 'False positive: 117'],
       ['False negative: 0', 'True negative: 0']], dtype='<U19')
```

6.1.5.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test / (TRUE_POSITIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)
print("Sensitivity: ", sensitivity_MobileNet_Test)
```

```
Sensitivity:  1.0
```

6.1.5.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_MobileNet_Test = TRUE_NEGATIVE_MobileNet_Test / (TRUE_NEGATIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)
    print("Specifity: ", specifity_MobileNet_Test)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

6.1.5.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_MobileNet_Test = TRUE_POSITIVE_MobileNet_Test / (TRUE_POSITIVE_MobileNet_Test + FALSE_POSITIVE_MobileNet_Test)
print("Precision: ", predcision_MobileNet_Test)
```

```
Precision:  0.805
```

Figure 54: Project Demonstration - Evaluation of MobileNet Model Individually on Testing Data(i)

6.1.5.6 Calculating Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_MobileNet_Test = TRUE_NEGATIVE_MobileNet_Test / (TRUE_NEGATIVE_MobileNet_Test + FALSE_NEGATIVE_MobileNet_Test)
    print("Negative predictive value: ", npv_MobileNet_Test)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

6.1.5.7 Calculating Accuracy

```
# Accuracy
accuracy_MobileNet_Test = (TRUE_POSITIVE_MobileNet_Test + TRUE_NEGATIVE_MobileNet_Test) / (TRUE_POSITIVE_MobileNet_Test + FALSE_P
print("Accuracy: ", accuracy_MobileNet_Test)
```

```
Accuracy:  0.805
```

Figure 55: Project Demonstration - Evaluation of MobileNet Model Individually on Testing Data(ii)

INCEPTION

```
import seaborn as sns
sns.set()
labels_Inception_Test =  np.array([['True positive: ' + str(TRUE_POSITIVE_Inception_Test),
                                    'False positive: ' + str(FALSE_POSITIVE_Inception_Test)],
                                   ['False negative: ' + str(FALSE_NEGATIVE_Inception_Test),
                                    'True negative: ' + str(TRUE_POSITIVE_Inception_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Inception_Test, annot = labels_Inception_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa166505390>
```

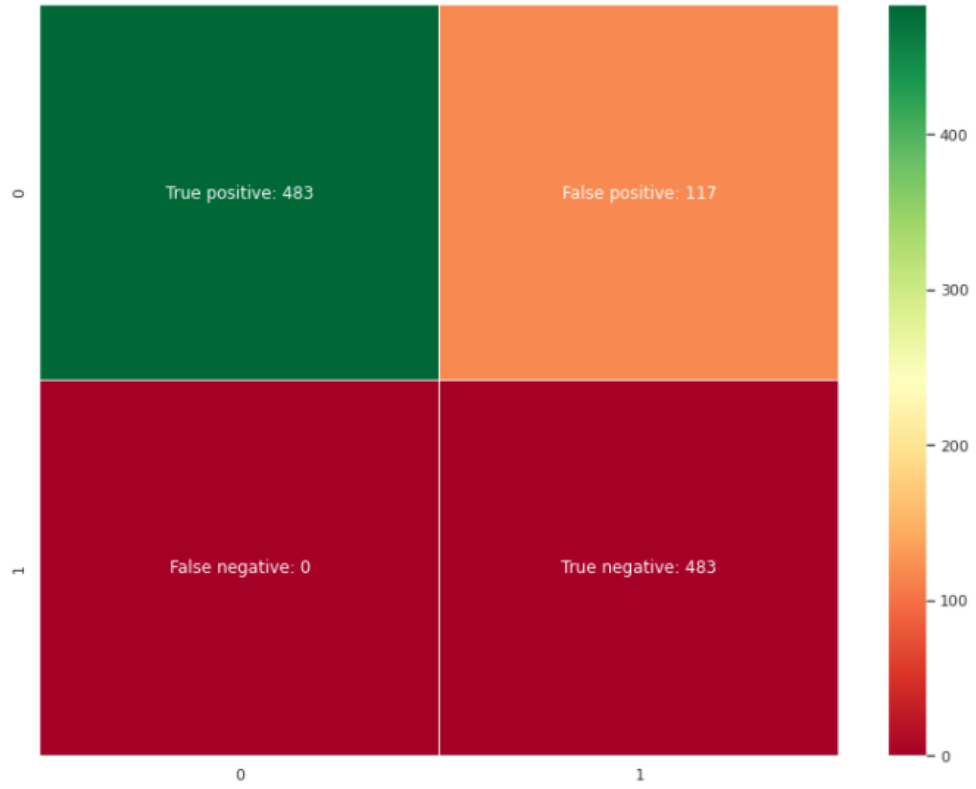


Figure 56: Project Demonstration - Evaluation of Inception Model Individually on Testing Data - Confusion Matrix

6.2.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Inception_Test = TRUE_POSITIVE_Inception_Test / (TRUE_POSITIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
print("Sensitivity: ", sensitivity_Inception_Test)
```

```
Sensitivity:  1.0
```

6.2.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Inception_Test = TRUE_NEGATIVE_Inception_Test / (TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
    print("Specifity: ", specifity_Inception_Test)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

6.2.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Inception_Test = TRUE_POSITIVE_Inception_Test / (TRUE_POSITIVE_Inception_Test + FALSE_POSITIVE_Inception_Test)
print("Precision: ", predcision_Inception_Test)
```

```
Precision:  0.805
```

Figure 57: Project Demonstration - Evaluation of Inception Model Individually on Testing Data(i)

6.2.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Inception_Test = TRUE_NEGATIVE_Inception_Test / (TRUE_NEGATIVE_Inception_Test + FALSE_NEGATIVE_Inception_Test)
    print("Negative predictive value: ", npv_Inception_Test)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

6.2.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Inception_Test = (TRUE_POSITIVE_Inception_Test + TRUE_NEGATIVE_Inception_Test) / (TRUE_POSITIVE_Inception_Test + FALSE_F
print("Accuracy: ", accuracy_Inception_Test)
```

```
Accuracy:  0.805
```

Figure 58: Project Demonstration - Evaluation of Inception Model Individually on Testing Data(ii)

XCEPTION

6.3.2.4.2 Obtaining the Labels

```
import seaborn as sns
sns.set()
labels_Xception_Test =  np.array([['True positive: ' + str(TRUE_POSITIVE_Xception_Test),
                                   'False positive: ' + str(FALSE_POSITIVE_Xception_Test)],
                                  ['False negative: ' + str(FALSE_NEGATIVE_Xception_Test),
                                   'True negative: ' + str(TRUE_POSITIVE_Xception_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Xception_Test, annot = labels_Xception_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa172937750>
```

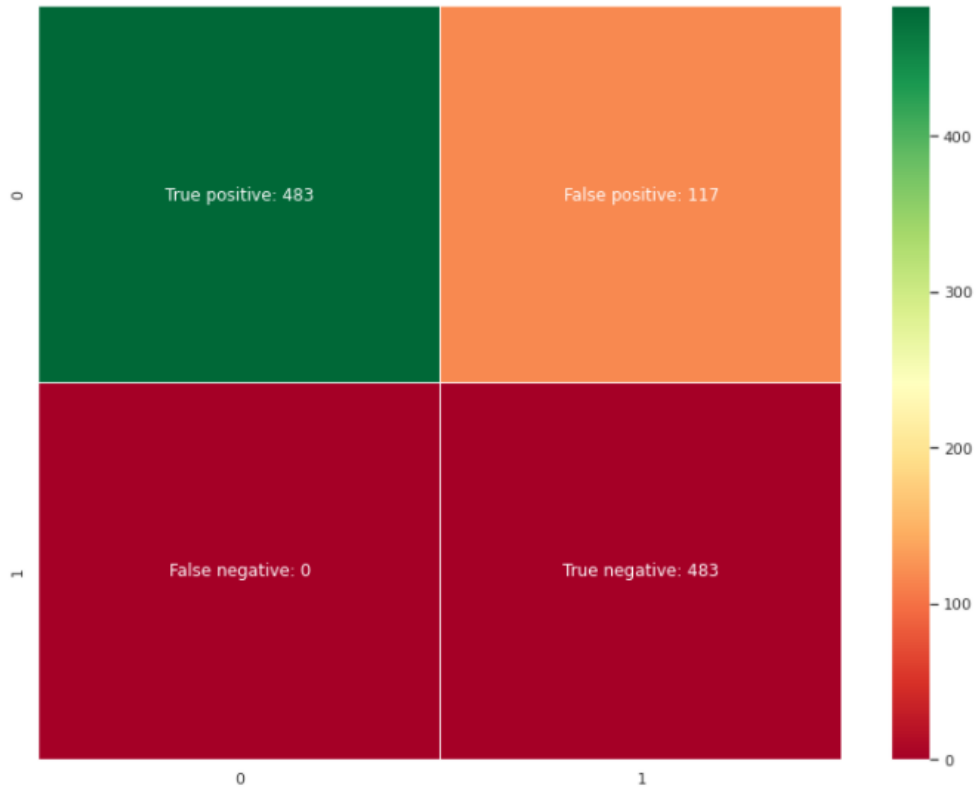


Figure 59: Project Demonstration - Evaluation of Xception Model Individually on Testing Data - Obtaining Labels

6.3.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
print("Sensitivity: ", sensitivity_Xception_Test)
```

```
Sensitivity:  1.0
```

6.3.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Xception_Test = TRUE_NEGATIVE_Xception_Test / (TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
    print("Specifity: ", specifity_Xception_Test)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

6.3.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Xception_Test = TRUE_POSITIVE_Xception_Test / (TRUE_POSITIVE_Xception_Test + FALSE_POSITIVE_Xception_Test)
print("Precision: ", predcision_Xception_Test)
```

```
Precision:  0.805
```

Figure 60: Project Demonstration - Evaluation of Xception Model Individually on Testing Data(i)

6.3.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Xception_Test = TRUE_NEGATIVE_Xception_Test / (TRUE_NEGATIVE_Xception_Test + FALSE_NEGATIVE_Xception_Test)
    print("Negative predictive value: ", npv_Xception_Test)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

6.3.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Xception_Test = (TRUE_POSITIVE_Xception_Test + TRUE_NEGATIVE_Xception_Test) / (TRUE_POSITIVE_Xception_Test + FALSE_POSIT
print("Accuracy: ", accuracy_Xception_Test)
```

```
Accuracy:  0.805
```

Figure 61: Project Demonstration - Evaluation of Xception Model Individually on Testing Data(ii)

EVALUATING THE MODELS TOGETHER ON TESTING DATA - ENSEMBLING THE MODELS

```
print("Accuracy = %2.2f%%" % (np.mean(correct_Ensemble_Test)*100))
```

```
Accuracy = 80.50%
```

```
image_to_predict_Ensemble_Test = path_to_tensor("/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/Output_melanoma/ISI
ensemble_model.predict(image_to_predict_Ensemble_Test)
```

```
array([[0.8248186 , 0.17518133]], dtype=float32)
```

Figure 62: Project Demonstration - Evaluation of Models Together on a Sample Image

7.5.2.4.2 Obtaining the Labels

```
import seaborn as sns
sns.set()
labels_Ensemble_Test =  np.array([['True positive: ' + str(TRUE_POSITIVE_Ensemble_Test),
                                   'False positive: ' + str(FALSE_POSITIVE_Ensemble_Test)],
                                  ['False negative: ' + str(FALSE_NEGATIVE_Ensemble_Test),
                                   'True negative: ' + str(TRUE_POSITIVE_Ensemble_Test)]])
plt.figure(figsize = (13, 10))
sns.heatmap(postives_negatives_Ensemble_Test, annot = labels_Ensemble_Test, linewidths = 0.1, fmt="", cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa162f37dd0>
```

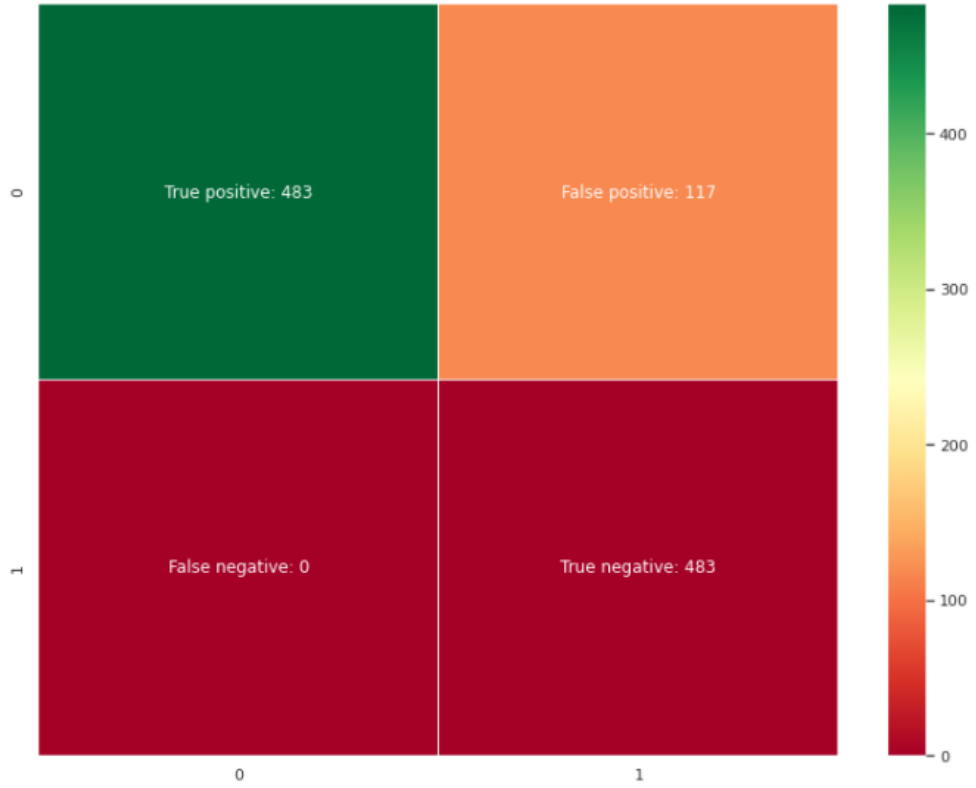


Figure 63: Project Demonstration - Evaluation of Models Together - Obtaining the Labels

7.5.2.4.3 Calculating Sensitivity/Recall/Hit Rate/True Positive Rate

```
# Sensitivity | Recall | hit rate | true positive rate (TPR)
sensitivity_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
print("Sensitivity: ", sensitivity_Ensemble_Test)
```

```
Sensitivity:  1.0
```

7.5.2.4.4 Calculating Specificity/Selectivity/True Negative Rate

```
# Specificity | selectivity | true negative rate (TNR)
try:
    specifity_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test / (TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
    print("Specifity: ", specifity_Ensemble_Test)
except:
    print("No Specificity due to NO NEGATIVE results.")
```

```
No Specificity due to NO NEGATIVE results.
```

7.5.2.4.5 Calculating Precision/Positive Predictive Value

```
# Precision | positive predictive value (PPV)
predcision_Ensemble_Test = TRUE_POSITIVE_Ensemble_Test / (TRUE_POSITIVE_Ensemble_Test + FALSE_POSITIVE_Ensemble_Test)
print("Precision: ", predcision_Ensemble_Test)
```

```
Precision:  0.805
```

Figure 64: Project Demonstration - Evaluation of Models Together on Metrics(i)

7.5.2.4.6 Negative Predictive Value

```
# Negative predictive value (NPV)
try:
    npv_Ensemble_Test = TRUE_NEGATIVE_Ensemble_Test / (TRUE_NEGATIVE_Ensemble_Test + FALSE_NEGATIVE_Ensemble_Test)
    print("Negative predictive value: ", npv_Ensemble_Test)
except:
    print("0 Negative Predictions")
```

```
0 Negative Predictions
```

7.5.2.4.7 Calculating Accuracy

```
# Accuracy
accuracy_Ensemble_Test = (TRUE_POSITIVE_Ensemble_Test + TRUE_NEGATIVE_Ensemble_Test) / (TRUE_POSITIVE_Ensemble_Test + FALSE_POSIT
print("Accuracy: ", accuracy_Ensemble_Test)
```

```
Accuracy:  0.805
```

Figure 65: Project Demonstration - Evaluation of Models Together on Metrics(ii)

LOCALIZATION

```
# Visualizing images with and without localization
# Canvas
fig, ax = plt.subplots(nrows=1, ncols=2, figsize = (10, 10))
# Image without localization
ax[0].imshow((path_to_tensor(img_path).astype('float32')/255).squeeze())
# Image with localization
CAM = plot_CAM(img_path, ax[1], mobilenet_model, all_amp_layer_weights)
plt.show()
```

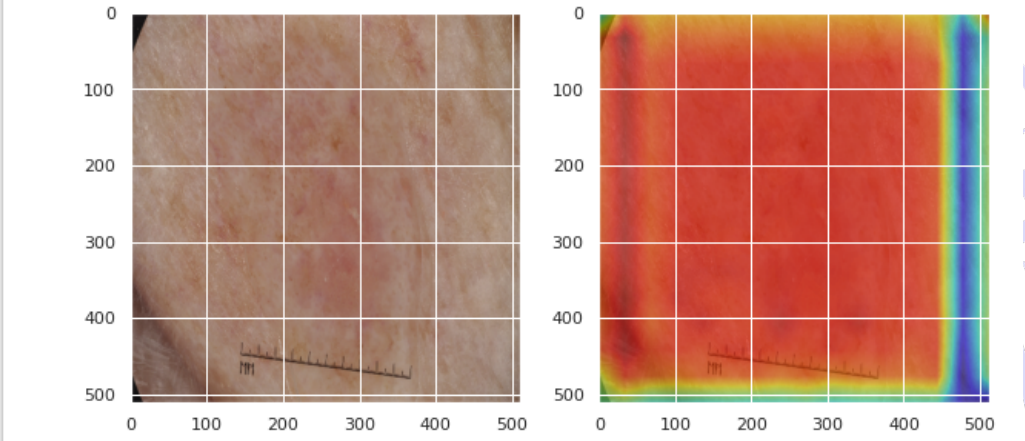


Figure 66: Project Demonstration - Localization - Visualizing Images

```
[ ]
# Canvas initialization
fig = plt.figure(figsize = (10, 10))

# First image
ax = fig.add_subplot(121)
ax.imshow(image_to_predict.squeeze())
ax.text(0.3, 1.6, prediction_final)

# Second image
ax = fig.add_subplot(122)
CAM = plot_CAM(img_path, ax, mobilenet_model, all_amp_layer_weights)

plt.show()
```

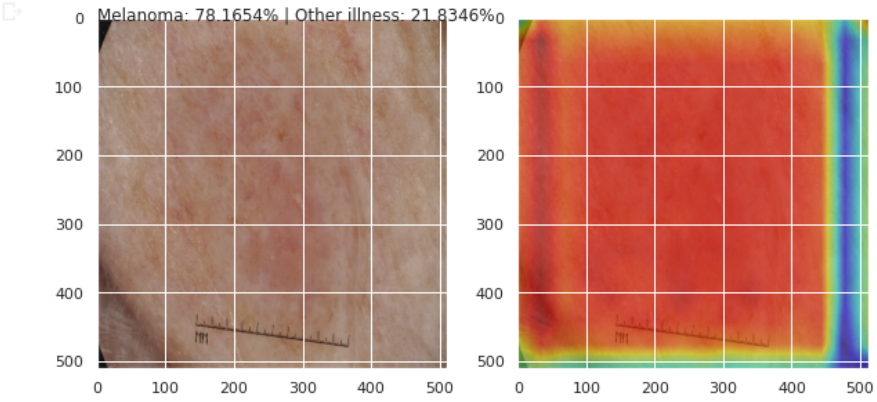


Figure 67: Project Demonstration - Localization - Display of Result

SAVING THE COMPLETE MODEL FOR PYTHON INTERFACE APPLICATION

9. Saving the Complete Model for the Python Interface Application

```
In [ ]: %cd "/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPUTS/"
ensemble_model.save('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')
from tensorflow.keras.models import load_model
h5_saved_ensemble_model = load_model('FINAL_FILE_for_Soft_Computing_Project_Skin_Cancer.h5')
h5_saved_ensemble_model.summary()
```

```
/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPUTS
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
Model: "ensemble"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_11 (InputLayer)           [(None, 512, 512, 3) 0
__________________________________________________________________________________________________
conv2d_490 (Conv2D)             (None, 255, 255, 32) 864         input_11[0][0]
__________________________________________________________________________________________________
batch_normalization_490 (BatchN (None, 255, 255, 32) 96          conv2d_490[0][0]
__________________________________________________________________________________________________
activation_470 (Activation)     (None, 255, 255, 32) 0           batch_normalization_490[0][0]
__________________________________________________________________________________________________
conv2d_491 (Conv2D)             (None, 253, 253, 32) 9216        activation_470[0][0]
__________________________________________________________________________________________________
batch_normalization_491 (BatchN (None, 253, 253, 32) 96          conv2d_491[0][0]
__________________________________________________________________________________________________
activation_471 (Activation)     (None, 253, 253, 32) 0           batch_normalization_491[0][0]
```

Figure 68: Project Demonstration - Saving the Model for Application Interface

CONVERTING THE .H5 FILE TO .TFLITE FILE FOR THE PYTHON INTERFACE APPLICATION

10. Converting the .h5 File to .tflite File for the Python Interface Application

```
import tensorflow as tf

saved_ensemble_model = tf.keras.models.load_model('/content/drive/MyDrive/dataset/ISIC Challenge 2017 Organized/FINAL SAVED OUTPU
converter = tf.lite.TFLiteConverter.from_keras_model(saved_ensemble_model)
tflite_model = converter.convert()
open("FINAL_FILE_for_Interface_Soft_Computing_Project_Skin_Cancer.tflite", "wb").write(tflite_model)
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
INFO:tensorflow:Assets written to: /tmp/tmp_6qz3iyy/assets

183140620
```

Figure 69: Project Demonstration - Converting the H5 File to TFLITE File

COST ANALYSIS, RESULT AND DISCUSSION

The project required to be subscribed to the Pro version of Google Colab which is on a Recurring Billing system and charges with $9.99/month excluding the taxes. The subscription was needed for 3 months which is the time period required for the project to complete.

SUMMARY

With the help of the model created with the help of Transfer Learning and Ensemble Modelling, we were able to achieve an accuracy of 80.5%. we have the results as:

Evaluation Type	Architecture	Confusion Matrix				Sensitivity/ Recall	Specificity	Precision	Negative Predictive Value	Accuracy
		True Positive	False Positive	True Negative	False Negative					
	MobileNet	120	30	0	0	1	0	0.8	0	0.8

Evaluation of Individual Models on Validation Data	Inception	120	30	0	120	1	0	0.8	0	0.8
	Xception	120	30	0	120	1	0	0.8	0	0.8
Evaluation of Ensemble Model on Validation Data	Ensemble Model	120	30	0	120	1	0	0.8	0	0.8
Evaluation of Individual Models on Testing Data	MobileNet	483	117	0	0	1	0	0.805	0	0.805
	Inception	483	117	0	483	1	0	0.805	0	0.805
	Xception	483	117	0	483	1	0	0.805	0	0.805
Evaluation of Ensemble Model on Testing Data	Ensemble Model	483	117	0	483	1	0	0.805	0	0.805

REFERENCES

WEBSITES

- https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/
- https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-3-dcb54abaf7b0
- https://towardsdatascience.com/data-visualization-for-eda-exploratory-data-analysis-f001a1bf0087
- https://analyticsindiamag.com/exploratory-data-analysis-functions-types-tools/
- https://medium.com/mljcunito/the-last-step-in-data-preprocessing-handling-missing-values-ae8f19dae309
- https://www.kdnuggets.com/2020/07/easy-guide-data-preprocessing-python.html

CONCEPTS & INFORMATION

- ResearchGate.com
- ScienceDirect.com
- GeeksforGeeks.com
- TutorialsPoint.com
- cs.stanford.edu
- Springer.com
- towardsdatascience.com
- academictorrents.com
- kaagle.com
- ncbi.nlm.nih.gov
- springml.com

- https://www.researchgate.net/publication/334751850_Skin_Cancer_Detection_Using_Convolutional_Neural_Network
- Peer-review under responsibility of the scientific committee of the 16th International Learning & Technology Conference 2019. 10.1016/j.procs.2019.12.090
- https://thesai.org/Publications/IJACSA
- https://www.thelancet.com/journals/ebiom/article/PIIS2352-3964(19)30294-4/fulltext
- https://www.sciencedirect.com/science/article/pii/S2215016120300832?via%3Dihub
- https://www.sciencedirect.com/science/article/pii/S1532046418301618?via%3Dihub
- https://www.ejcancer.com/article/S0959-8049(19)30381-8/fulltext
- https://www.researchgate.net/publication/335321267_Bio-Inspired_DeepCNN_Pipeline_for_Skin_Cancer_Early_Diagnosis
- https://www.researchgate.net/publication/347927500_Automated_Multiclass_Classification_of_Skin_Lesions_through_Deep_Convolutional_Neural_Network_with_Dermoscopic_Images
- https://www.researchgate.net/publication/339804392_Deep_Learning_Solutions_for_Skin_Cancer_Detection_and_Diagnosis?enrichId=rgreq-cc1ddabe639ba17298c595c27b48021c-XXX&enrichSource=Y292ZXJQYWdlOzMzOTgwNDM5MjtBUzo4ODM0MjE3NjUxNzMyNTFAMTU4NzYzNTU3MDg4Nw%3D%3D&el=1_x_3&_esc=publicationCoverPdf
- https://ieeexplore.ieee.org/document/9062473
- https://www.sciencedirect.com/science/article/pii/S0933365719301460
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3352407
- https://www.researchgate.net/publication/335337495_A_CNN_toolbox_for_skin_cancer_classification?enrichId=rgreq-91f014447a4c29cec1b8c22e6f8930fb-XXX&enrichSource=Y292ZXJQYWdlOzMzNTMzNzQ5NTtBUzo4MDExNDUyMDMyMjQ1NzdAMTU2ODAxOTMwOTE5Nw%3D%3D&el=1_x_3&_esc=publicationCoverPdf
- https://ieeexplore.ieee.org/document/8720210
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6231861/
- https://iopscience.iop.org/article/10.1088/1757-899X/982/1/012005
- https://www.researchgate.net/publication/327539277_Skin_Cancer_Classification_using_Convolutional_Neural_Networks_Systematic_Review_Preprint
- https://www.sciencedirect.com/science/article/pii/S2352914819302047
- https://www.researchgate.net/publication/334123580_Melanoma_Skin_Cancer_Detection_using_Image_Processing_and_Machine_Learning
- https://www.nature.com/articles/nature21056
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9113301
- https://pubmed.ncbi.nlm.nih.gov/28117445/
- https://pubmed.ncbi.nlm.nih.gov/30333097/
- https://ieeexplore.ieee.org/document/9198489