

## Contents

Introduction .....	2
Purpose .....	2
Scope.....	2
Future Goals.....	2
Overview .....	3
Main Objective.....	3
Background .....	3
Literature Review.....	3
Methodology.....	4
Software Requirements: .....	4
Work performed .....	5
Designing.....	5
IMPLEMENTATION and testing: .....	8
Conclusion.....	9
References .....	9

## **Introduction**

---

This part of the document explains the purpose, scope, future goals and also gives an overview of the whole document.

Image segmentation is an important technology for image processing. There are many applications whether on synthesis of the objects or computer graphics images require precise segmentation.

Nowadays, many programs are among the popular synthesis, and there is no doubt that viewers interest is concentrated on the main lead. The highest demand of image segmentation is for sports scene in term of both visual compression and image handling using extraction.

The colour information and edge extraction are done to achieve the image segmentation. The colour information helps obtain the texture information of the target image while the edge extraction detects the boundary of the target image. By combining these, the target image can be correctly segmented and represent. From this project we are able to extract most part of the target.

### **Purpose**

This purpose of working with image segmentation is to make an image more detailed and easier to analyse therefore it get more meaningful. More precisely, image segmentation is the proses of recognizing the object in an image the number and what object is being placed in the image.

It helps us recognize that pixels with the same label share certain characteristics.

### **Scope**

The scope of the project is helpful but will get very innovative in future because this project helps us get all the information within an image. The tags, objects, colours and many more to be detected. The performing multiple operation on any type of image you input. But specifically, we choose butterfly image to be used as an input source.

### **Future Goals**

Following the future goals of our project and improvements will be made:

- The performance requirements of an image processing application is continuously increased by the computing power of implementation platforms, especially when they are executed under real time constraints.
- The real time applications may consist of different image standards, or different algorithms used at different stages of the processing chain.
- The computing paradigm using reconfigurable architecture promises an intermediate-off between flexibility and performance.

## Overview

---

### Main Objective

The main objective of our project is to perform segmentation on the outlined of butterflies in outdoor images. We are considering only those images in which the wings are opened and ignoring the butterfly's feelers.

### Background

Image segmentation is, essentially, a classification task in which we **classify each pixel** as belonging to one of the target classes. So when you pass an image through a segmentation model, it will give one label to each of the pixels that present in the image.

If we then color each pixel based on the class that pixel belongs to, we'll be able to easily locate objects and their boundaries. Each pixel belonging to a particular target class is a different color. In this case, pixels belonging to houses are red, and pixels belonging to the non-road ground is blue. This way, we can get the location of objects and all the pixels belonging to that object in a given image. This gives us a fine-grain understanding of an image.

Broadly, segmentation can be divided into two categories:

- 1) **Instance Segmentation**
- 2) **Semantic Segmentation**

### Instance Segmentation:

This takes semantic segmentation one step further and involves detecting objects within defined categories. For e.g. – In the same street scene, you would individually draw boundaries for each of the category and uniquely label – Humans – (Adult, Kid), Automobiles – (Cars, Bus, Motor Bikes...), and so on.

### Semantic Segmentation:

Semantic segmentation can detect objects within the input image, isolate them from the background and group them based on their class. Instance segmentation takes this process a step further and can detect each individual object within a cluster of similar objects, drawing the boundaries for each of them.

## Literature Review

---

There are many algorithms used for image segmentation, some of them segmented an image based on the object while some can segment automatically. Nowadays, no one can point out which the optimal solution is due to different constraints. A similarity close measure was used to classify the belonging of the pixels, and then used region growing to get the object. Unfortunately, it required a set of markers, and if there is an unknown image, it is hard to differentiate which part should be segmented. Linking the area information and the colour histogram were considered for building video databases based on objects. However, the colour information has to be given first, and it is not useful for the life application. A genetic algorithm adapted the segmentation process to changes in image characteristics caused by

variable environmental conditions but it took time learning. In a two-step approach to image segmentation is reported. It was a fully automated model-based image segmentation, and improved active shape models, line-lanes and live-wires, intelligent scissors, core-atoms, active appearance models. However, there were still two problems left. It is strong dependency on a close-to-target initialization, and necessary for manual redesign of segmentation criteria whenever new segmentation problem is encountered. The authors in proposed a graph-based method, the cut ratio is defined following the idea of NP-hard as the ratio of the corresponding sums of two different weights of edges along the cut boundary and models the mean affinity between the segments separated by the boundary per unit boundary length. It allows efficient iterated region-based segmentation as well as pixel-based segmentation. Moreover, in order to understand an image and recognize the represented objects, it is necessary to locate in the image where the objects are. The homogeneity between two pixels and the distance function are included to measure the segmented results  $DI = |I(x, y) - I(v, w)|$ .

Edge detection is a fundamental concept for segmentation because it is easy not only to understand its principle but also to implement. By edge detection we can simply point out the proposed algorithm that extracts some undesired parts into the segmentation result

## **Methodology**

---

In our algorithms, there are some criteria. First of all, we need to be aware of the target image which we would like to segment out. Second, the background image has to be blurred and the colour of the target image should be different to that of background image as much as possible. Moreover, we expect the appendages of the target image to cross over each other as least as possible.

Our approach is to obtain color information of the target image and boundary extraction separately and simultaneously. We apply the character of HSI to acquire the information of the pixels of the target image.

In the meantime, we use the “edge” and “imfill” command to extract the boundary and fill the image region whose boundaries make a closure. Afterwards, we combine them by getting the union of the two results. Finally, we perform a final modification and remove the noise.

The whole algorithm is as follows:

- 1) Firstly, acquire the color information and the edge information separately.
- 2) Use the hue, saturation and intensity to get color information.
- 3) Use the “edge” command to extract the image boundary.
- 4) Combine the above results by getting the union of (2) and (3).
- 5) Final modification (noise removal).

## **Software Requirements:**

Following are our software requirements:

- Anaconda prompt
- Jupyter notebook
- Butterfly images
- Data set on Jupyter notebook

- Testing performed

## Work performed

The work performed in our projects is based on JUPYTER NOTEBOOK. We are creating three different codes. Also, using 2 types of butterflies for which we used 3 pictures for each. In our data set we are providing the details about the 2 butterflies. These details differ them from each other. We provided different pictures, data and some basic qualities about each. Then we created another notebook where we are calling our function. The program does not take input from the user. We are placing the images we provide the details about in the dataset code.

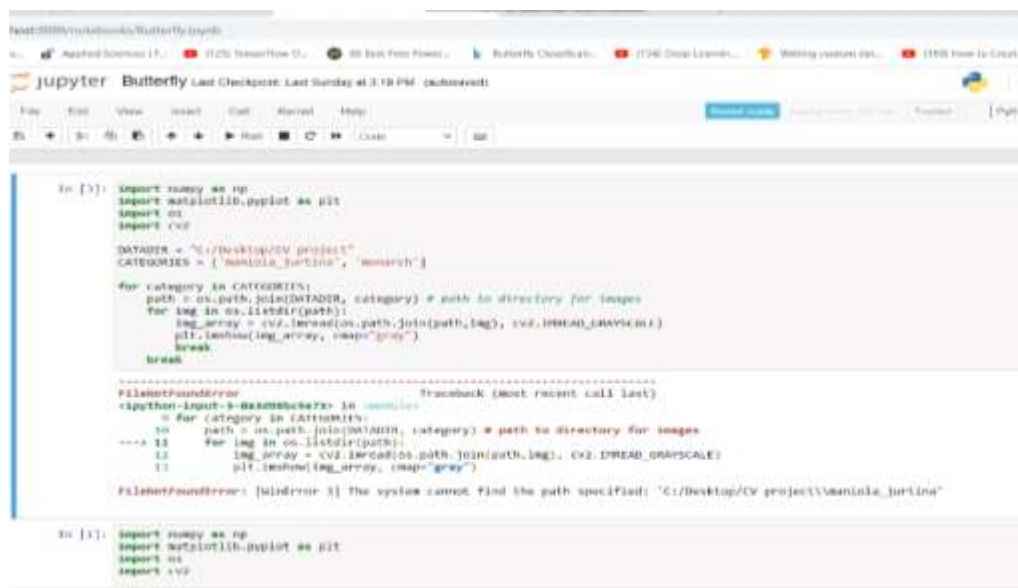
Now, when we run the program the images are already taken as input the program is directly compiled. It starts predicting about the images and gives the answers as output.

We are importing **cv2** and **Tensorflow** as **tf** in the main code. The we created the definition where the file path is given and images are called the image array and new array are created. The final model will be working n the data set. The main code is completed here.

Whereas, the data set code is shuffled multiple times. It uses the tensorflow method then imports the layer. X and Y are used to assign specifications. Sample model for each image is created where the details (size, resolution etc.) are provided. We also imported Panda as pd, pickle and time. In the end prediction contains the image according to the provided details.

## Designing

This project is designed to differentiate between two different types of Butterflies. The data was taken from kaggle and changes were made according to the requirements of the project.



```

In [3]: import numpy as np
import matplotlib.pyplot as plt
import os
import cv2

DATADIR = "%Desktop/CV project"
CATEGORIES = ['manila_turtina', 'monarch']

for category in CATEGORIES:
    path = os.path.join(DATADIR, category) # path to directory for images
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap="gray")
        break
    break

FileNotFoundError: [Errno 2] No such file or directory: 'C:/Desktop/CV project/manila_turtina'

In [1]: import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
  
```

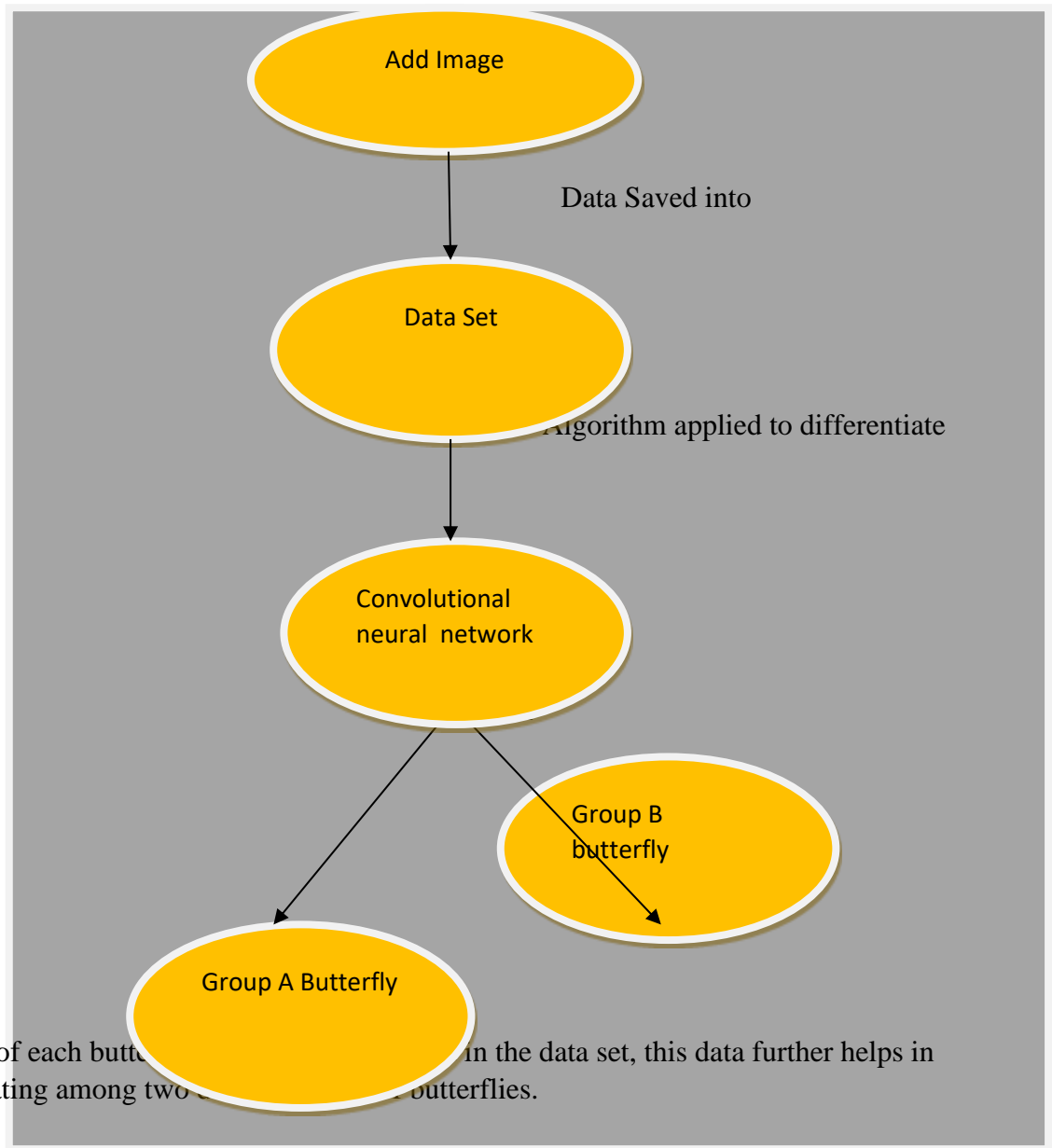
```
in [11]: IMG_SIZE = 70
new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array, cmap = 'gray')
plt.show()

in [x]: training_data = []
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATASET, category) # path to directory for images
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
```

## REQUIREMENTS:

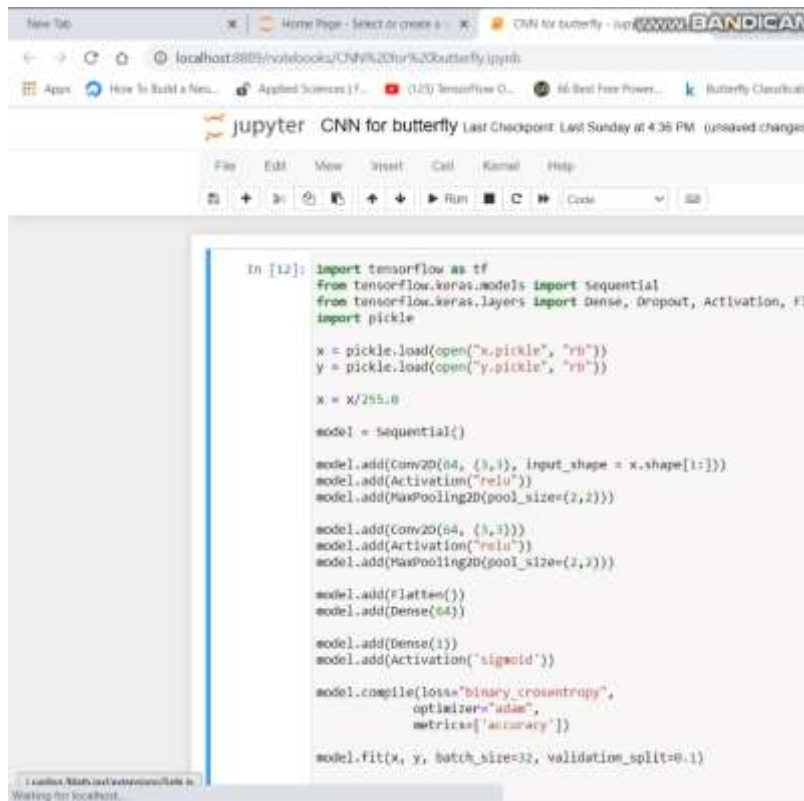
The requirement of this project is that the picture provided should not have the closed wings of the butterfly. As the project examines the wings of the butterfly and makes further calculations.

## FLOW CHART:



## IMPLEMENTATION and testing:

Some of the libraries were used to import pictures. Cv2 and Tensor flow as tf in the main code.



```
In [12]: import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
import pickle

x = pickle.load(open("x.pickle", "rb"))
y = pickle.load(open("y.pickle", "rb"))

x = x/255.0

model = Sequential()

model.add(Conv2D(64, (3,3), input_shape = x.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(64))

model.add(Dense(1))
model.add(Activation("sigmoid"))

model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=['accuracy'])

model.fit(x, y, batch_size=32, validation_split=0.1)
```

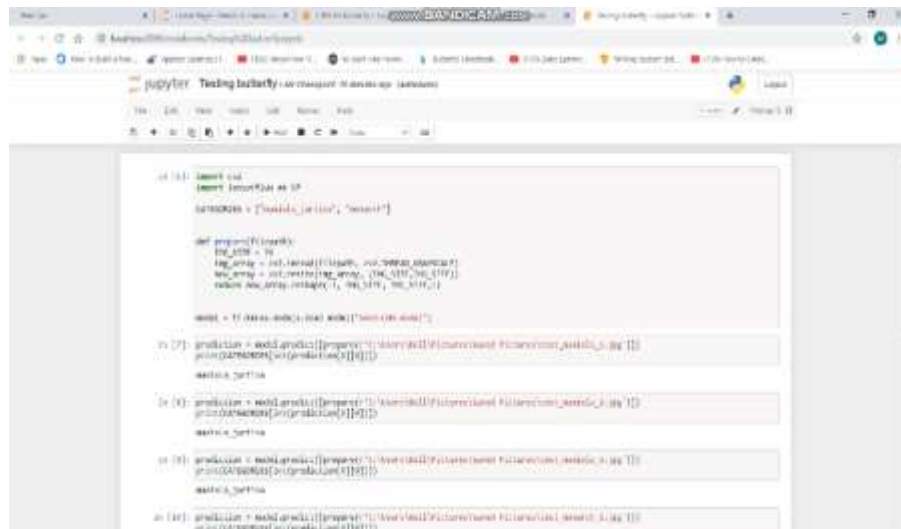
After entering the pictures and its data. The data provided makes a differentiation between 2



different types of butterflies.

When we recall the saved pictures and give it a test by using the command and picture name \_ picture number where it tells about the kind of butterfly it is.





```
def segment(image_path, threshold):  
    image = cv2.imread(image_path)  
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
    _, mask = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)  
    return mask  
  
image_path = 'data/images/1.jpg'  
threshold = 128  
mask = segment(image_path, threshold)  
cv2.imshow('Mask', mask)  
cv2.waitKey(0)  
cv2.destroyAllWindows()
```

## Conclusion

We have developed an old but neglected algorithm by providing a reliable stopping criterion and an acceleration, and applied it to image segmentation.

We used the neural network algorithm in our project. Creating different data sets and running the images through it. Different images can be segmented and predicted once the data set is set accordingly.

## References

[https://www.it.uu.se/edu/course/homepage/projektTDB/ht11/project17/Report\\_ht11\\_17.pdf](https://www.it.uu.se/edu/course/homepage/projektTDB/ht11/project17/Report_ht11_17.pdf)

<https://faculty.ucmerced.edu/mcarreira-perpinan/papers/icml06-poster.pdf>

[https://web.stanford.edu/class/ee368/Project\\_06/Project/ee368\\_reports/ee368group26.pdf](https://web.stanford.edu/class/ee368/Project_06/Project/ee368_reports/ee368group26.pdf)