NU 302 - Research and Development

Group 24 - Improvement of Image Segmentation

“Team Aspectus”

Group Members -

Aakash Sinha	U101114FCS214
Samaksh Singhal	U101114FCS124
Shivam Yadav	U101114FCS359
Akash Agarwal	U101114FCS038
Jatin Kakkar	U101114FCS072

Group Mentor -

Mr. Gaurav Sharma

Index :

Abstract :

We will be performing image segmentation on a given image. The machine learning library which we are using is TensorFlow. In the first phase we used Slim wrapper and VGG-16 model to classify an image in over 1000 classes and give significant probabilities using Softmax. In the second phase we have trained FCN-8s net on VGG-16 and it has used PASCAL VOC 2012 model (trained on ImageNet to generate 21 classes) to generate classes. Then we have used CRF as recursive function (in RNN) to generate a heat map of the obtained foreground which have been classified. Then we have used morphological operations on the image to detect contour and masking to retrieve the final output image. Then we have extended our project by making custom stickers for Telegram ChatBox.

Introduction :

Image segmentation is the process of partitioning a digital image into multiple segments having similar characteristics. Segmentation is generally the first stage in any attempt to analyze or interpret an image automatically. The goal of segmentation is to simplify the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that the pixels with the same label share same certain characteristics.

Objectives :

The main objective or goal of Image segmentation is to perfectly segment an given Image into a simpler Image for easy understanding and/or to gather a detailed information on only the particular part of an image. To achieve this goal and to make it work in any scenario several algorithms are devised and extensive research is still going on.

Why Image Segmentation ?

Below points justify the benefits of Image segmentation as it can be very efficient and helpful for these cases. These benefits lead to the practical use of image segmentation in real time scenarios.

- Application of this concept in crime branch.
- To help partially sighted people by highlighting important objects in their glasses.
- To let robots segment objects so that they can grasp them.
- In medical imaging such as measuring tissue volumes or in the diagnosis,study of anatomical structure.
- In recognition task such as Fingerprint or Iris recognition.
- Object detection such as locating objects from satellite images.
- Computer guided surgery.

Evaluation of Pre-Existing Segmentation Techniques :

Thresholding :

Thresholding is the simplest, powerful and most frequently/widely used technique for image segmentation. It is useful in discriminating foreground from the background. Thresholding operation is used to convert a multilevel or gray scale image into a binary image , which reduces the complexity of data and simplifies recognition and classification. Thresholds are either global or local means they can be constant throughout the image or spatially varying.

Pros :

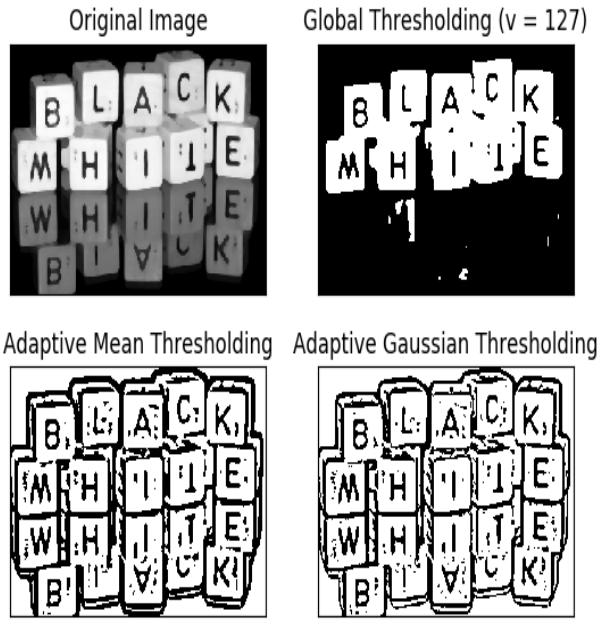
- Output is binary image which is easier to reprocess for further classification of image.
- Fast and simple for implementation.
- Computationally inexpensive.
- Can be used in real time applications.

Cons :

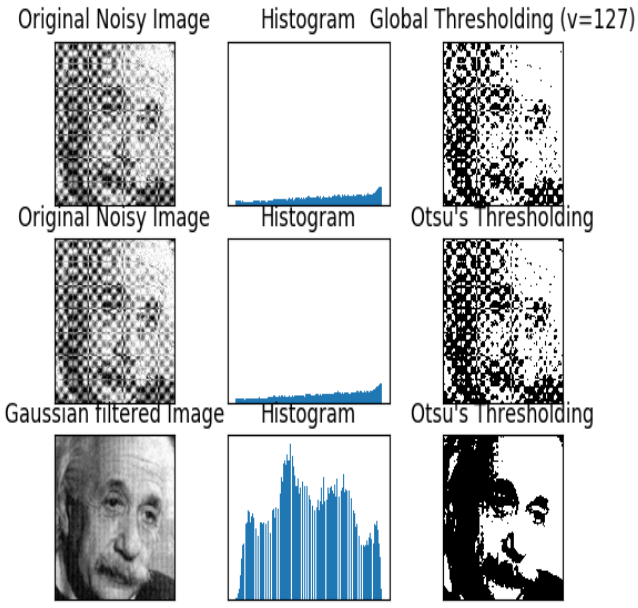
- If you get the threshold wrong the results can be disastrous means a wrong choice may result into over or under segmentation.
- Uneven illumination can really upset a single valued thresholding scheme.

Results :

```
__author__ = 'Shivam'

import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('bw.png',0)
img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresho
images = [img, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
```




```
__author__ = 'Shivam'

import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('EinStein.jpg',0)

# global thresholding
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img, 0, th1,
          img, 0, th2,
          blur, 0, th3]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=1
```



Watershed -

The aim of the watershed transform is to search for regions of high intensity gradients (watersheds) that divide neighbored local minima (basins).

- Highest gradient magnitude intensities (GMIs) correspond to watershed lines, represent the region boundaries.
- Water placed on a common watershed line flows downhill to a common local intensity minimum (LIM).
- Pixels draining to a common minimum form a segment.

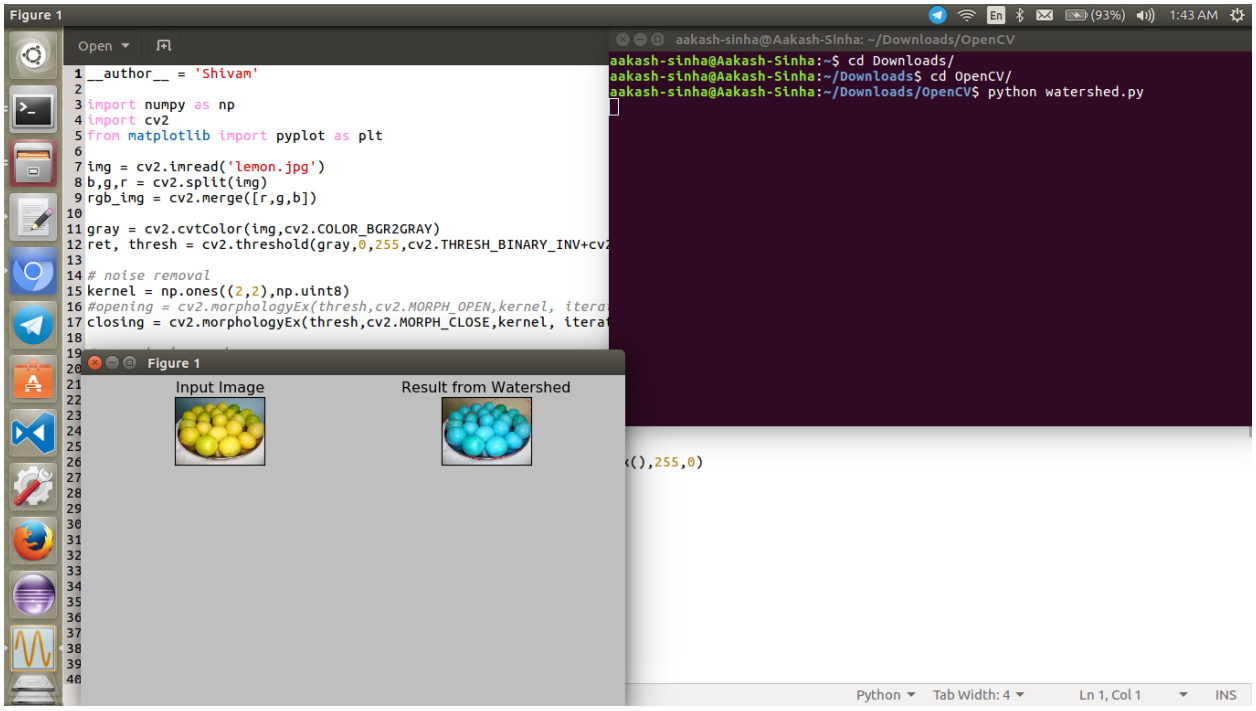
Pros :

- Form closed and connected regions.
- Traditional edge based techniques form disconnected boundaries
- This is in contrast to split and merge methods lead sometimes to unstable results.
- The union of all the regions forms the entire image region.

Cons :

- Sometimes produces excessive over segmentation.

Results :



K Means Clustering:

K -means clustering algorithm is an **unsupervised algorithm** and it is used to segment the interest area from the background.

The algorithm takes a 2 dimensional image as input. Various steps in the algorithm are as follows:

- Compute the intensity distribution(also called the histogram) of the intensities.
- Initialize the centroids with k random intensities.
- Repeat the following steps until the cluster labels of the image does not change anymore.
- Cluster the points based on distance of their intensities from the centroid

intensities.
- Compute the new centroid for each of the clusters.

Pros :

- For small values of k, k-means is computationally faster.
- Eliminates noisy spots.
- Reduces false blobs.
- More homogeneous regions are obtained.
- Simple easy to implement and interpret clustering results

Cons :

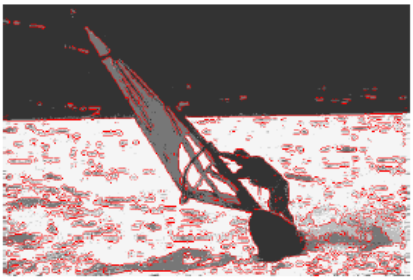
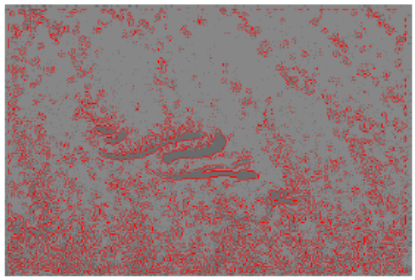
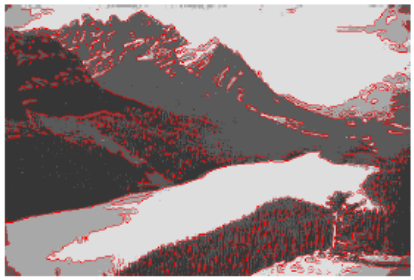
- Difficult to predict k with fixed number of clusters.
- Sensitive to initialization condition of cluster number and centre.
- Computationally expensive.
- Doesn’t works well with non globular clusters.

Results :

Original Image :



Clusters formed :

Artificial Neural Network was the “key” :

A segmentation methods used in Deep learning which is used to simulate the learning strategies of human brain for the purpose of decision making.

Following is the framework in which artificial neural networks (ANN) work:

- Assign random weights to all the linkages to start the algorithm.
- Using the inputs and linkages find the activation rate of output nodes.
- Using the activation rate of hidden nodes and linkages to output, find the activation rate of output nodes.
- Find the error rate at the output node and recallibrate all the linkages between hidden nodes and output nodes.
- Using the weights and error found at output node, cascade down the error to hidden nodes.
- Recalibrate the weights between hidden node and input nodes.
- Repeat the process till the convergence criterion is met.
- Using the final linkage weights score the activation rate of the output nodes.

Pros :

- Significantly improved the state of the art in semantic segmentation.
- No need to write complex programs.

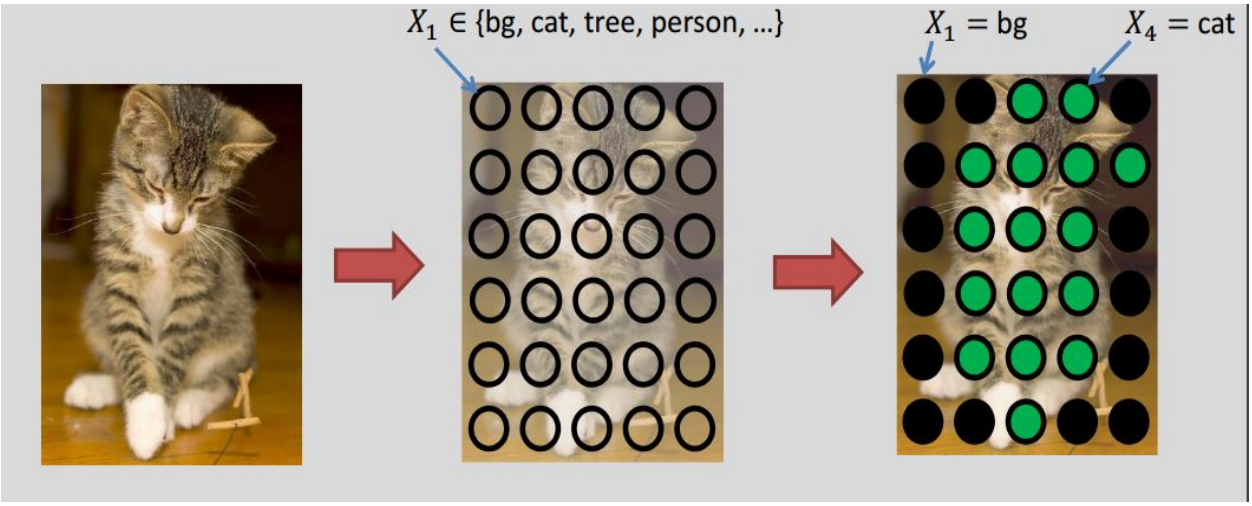
Cons :

- Poor object delineation: e.g. spatial consistency neglected.
- More wastage of time in training.

Conditional Random Field (CRF) :

This is a class of statistical modeling method often utilized for structured prediction and finds application in fields like pattern recognition and machine learning. CRFs belong to sequence modeling family. Where a discrete classifier works with predicting a single sample and marking it with a label,without considering neighboring elements, a CRF has the ability to consider context into account, like in the case of the linear chain CRF, popularly used in NLP, predicts upcoming/expected sequences of labels for a given sequence of input samples.

CRFs can be helpful in matching/utilizing defined or derived relationships between elements and give consistent interpretations.



The above picture is an example where we use the CRFs to find the pixels which belong to our subject of interest. We Intend to use CRFs as **RNN(Recurrent Neural Network)** to enhance the output results and get over the drawbacks which otherwise we would have faced.

RECURRENT NEURAL NETWORK (RNN) :

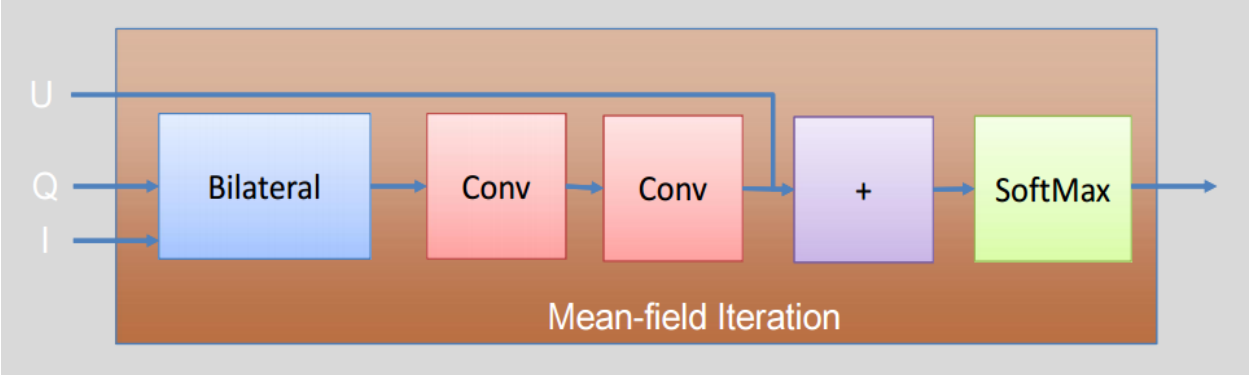
It is a class of artificial neural network, in which the connections between the units is similar to that of a directed cycle. This allows it to exhibit dynamic temporal behavior. Unlike their counterparts i.e feedforward neural networks, RNNs process arbitrary sequences of inputs via their internal memory. This makes them viable to perform certain tasks such as handwriting recognition and voice recognition.

In reinforcement learning, as in our case, it uses a method similar to least cost and knapsack method to evaluate its performance, as there is no instructor providing signals, which influences its input stream through output units connected to actuators.The **Recursive Neural Tensor Network** uses a tensor-based composition function for all nodes in the tree.

- Pairwise energies are defined for every pixel pair in the image.

$$E(\underline{x}) = \sum_i unary(x_i) + \sum_{i,j} pairwise(x_i, x_j)$$

Using the CRF as RNN would result in something like the picture below. This would drastically improve the segmentation quality.

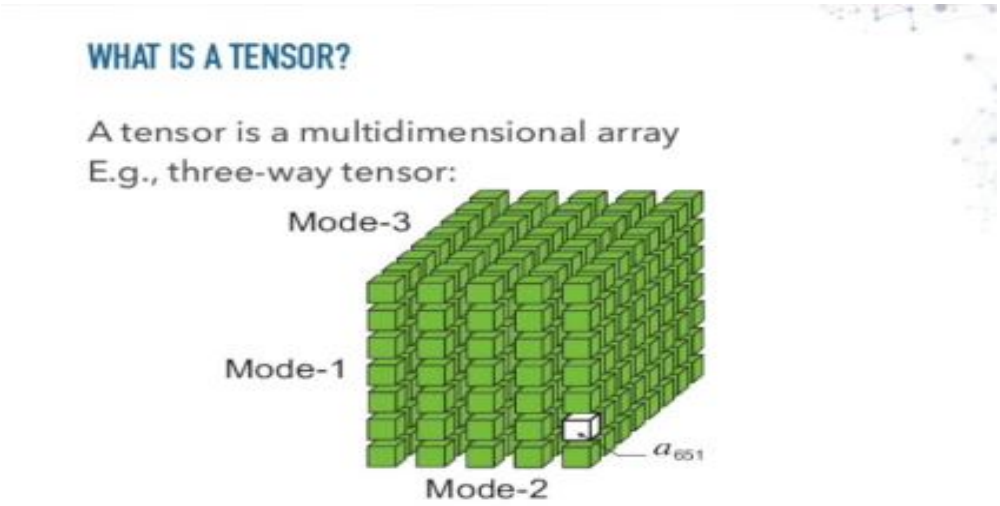


TENSORFLOW :

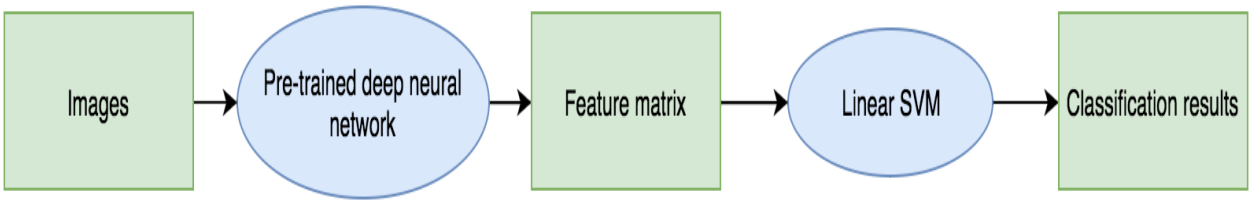
TensorFlow , developed by Google,is an open source software library for machine learning across a range of tasks, to meet the requirement of systems capable of building and training neural networks, to decipher patterns and correlations, analogous to the learning and reasoning which humans use.

Though there are quite many machine learning libraries, we chose tensorflow for the following reasons:

- Improves the portability of machine learning Tensor Flow can run on multiple CPUs and GPUs (with optional CUDA extensions).
- Accelerated research
- Model Parallelism (Iterative Map Reduction)
- Better than other Libraries like Thiano



Data is represented as Tensors in TF. It is a multi-dimensional array which is made up of numbers and they flow between operations. Hence the name.



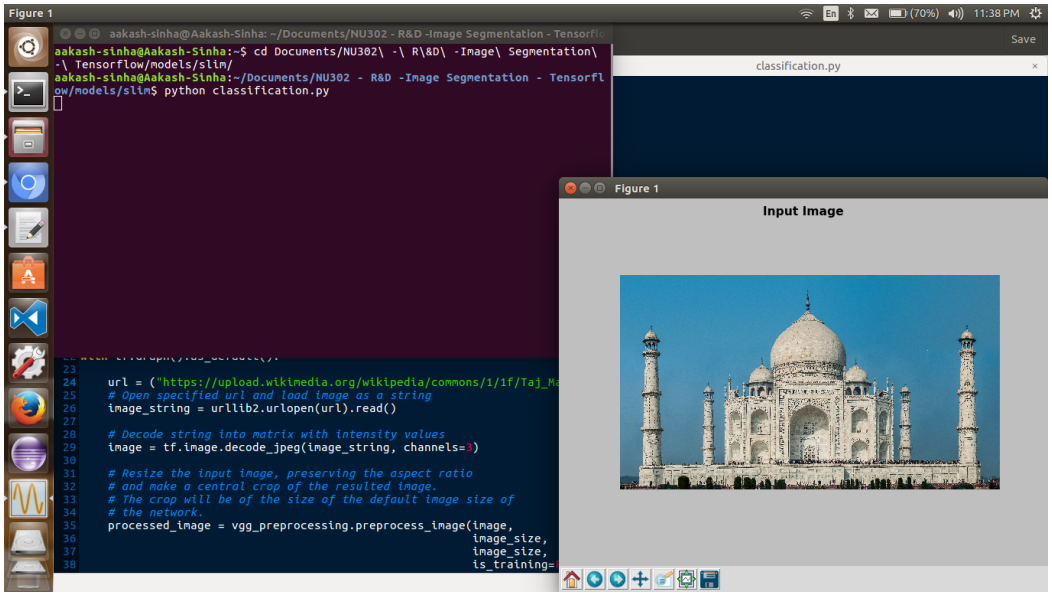
Dependencies used for Image Classification :

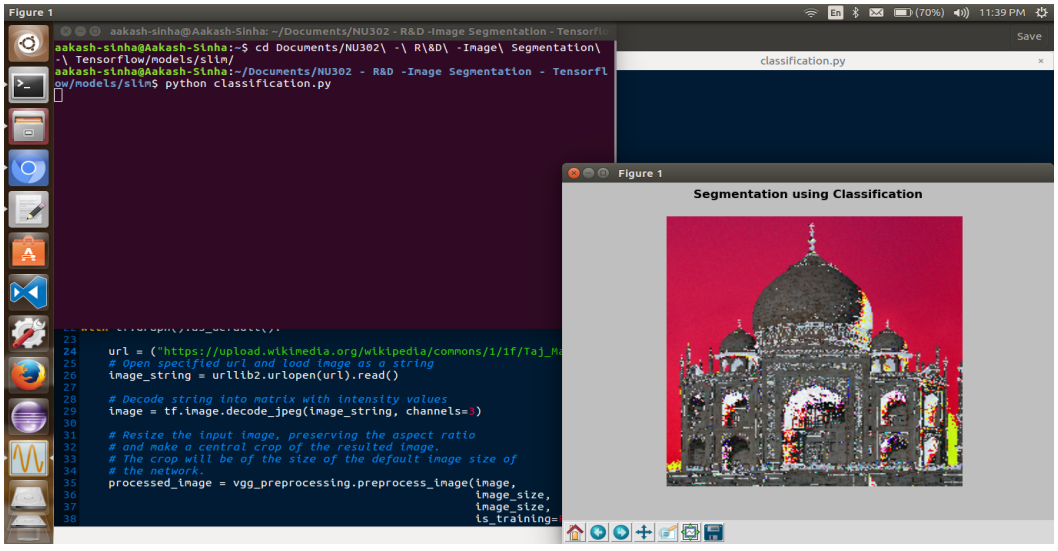
- **Slim Library :** It is a wrapper library around tensorflow, which contains three nets Resnet, VGG and inception. It gives a supportable build to tensorflow for processing images and segmentation
- **ResNet :** Residential Networking (ResNet) provides network and computer support for students living in residence halls and on-campus apartments. It is our goal to assist, educate, and support our customers in any way we can. We don't just fix problems–we help people. Support is provided at no extra charge, as the costs are included in the housing fees.
- **VGG :** refers to a deep convolutional network for object recognition developed and trained by Oxford's renowned Visual Geometry Group (VGG), which achieved very good performance on the ImageNet dataset.
- **Inception:** The goal of the inception module is to act as a "multi-level feature extractor" by computing *1×1*, *3×3*, and *5×5* convolutions within the *same* module of the network, the output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network.
- **Tensorflow r0.11 :** The version of tensorflow we are using
- **Scikit :** python library for image and scientific functions
- **MatPlotLib :** Matlab library for plotting and other Matlab functions
- **NumPy :** It is a numerical processing package that TensorFlow requires
- **Softmax Activation Function :** The **softmax function** is often used in the final layer of a neural network-based classifier. Such networks are

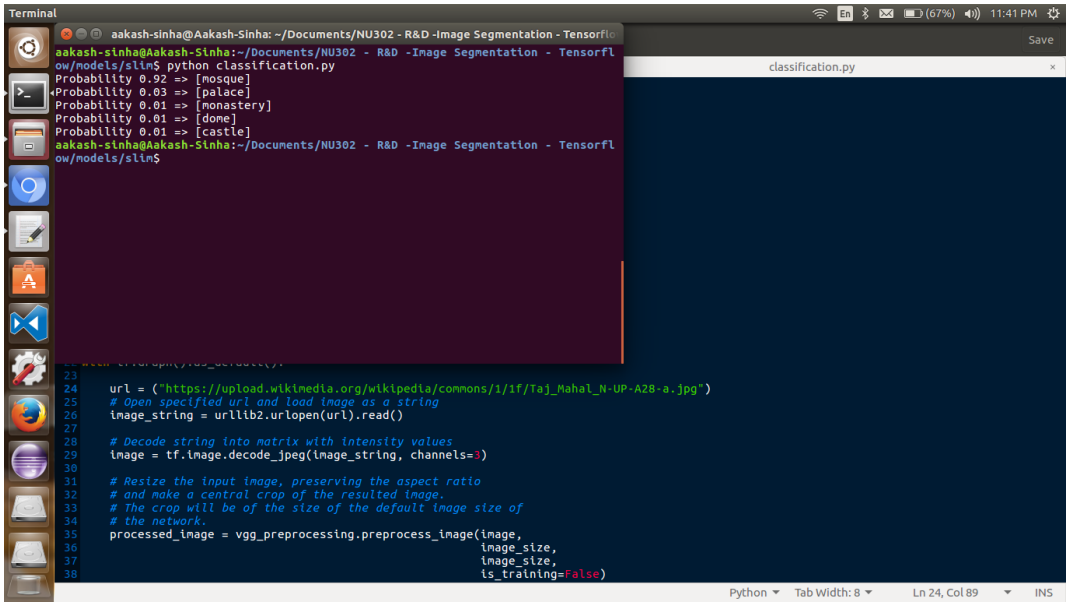
commonly trained under a log loss (or cross-entropy) regime, giving a nonlinear variant of multinomial logistic regression.

RESULTS :

Example 1 :






Terminal
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorflow/models/slim$ python classification.py
Probability 0.92 => [mosque]
Probability 0.03 => [palace]
Probability 0.01 => [monastery]
Probability 0.01 => [dome]
Probability 0.01 => [castle]
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorflow/models/slim$
classification.py
Save
url = ("https://upload.wikimedia.org/wikipedia/commons/1/1f/Taj_Mahal_N-UP-A28-a.jpg")
# Open specified url and load image as a string
image_string = urllib2.urlopen(url).read()
# Decode string into matrix with intensity values
image = tf.image.decode_jpeg(image_string, channels=3)
# Resize the input image, preserving the aspect ratio
# and make a central crop of the resulted image.
# The crop will be of the size of the default image size of
# the network.
processed_image = vgg_preprocessing.preprocess_image(image,
image_size,
image_size,
is_training=False)
Python
Tab Width: 8
Ln 24, Col 89
INS

Example 2 :

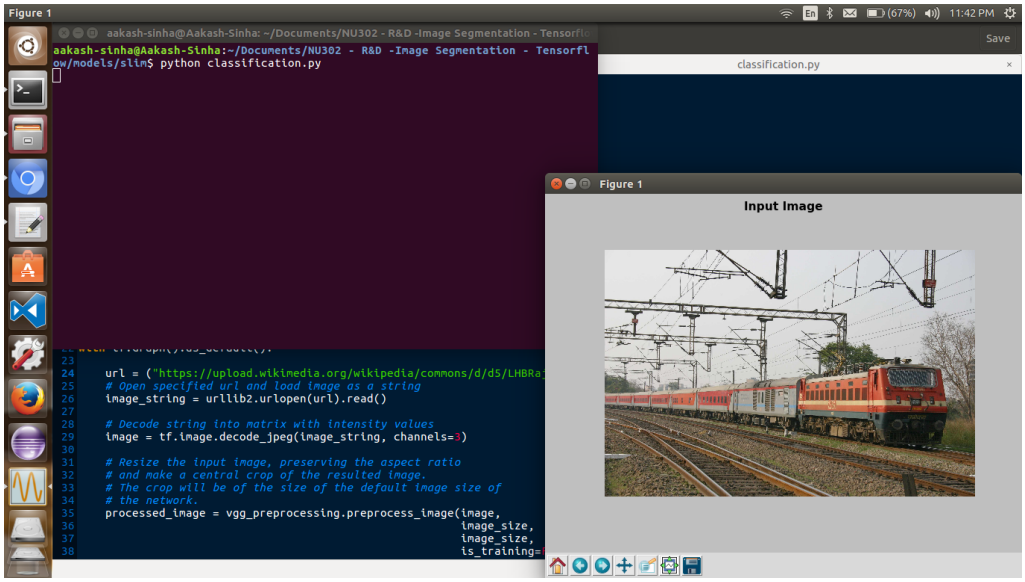


Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorflow/models/slim$ python classification.py
classification.py
Save
Figure 1
Input Image

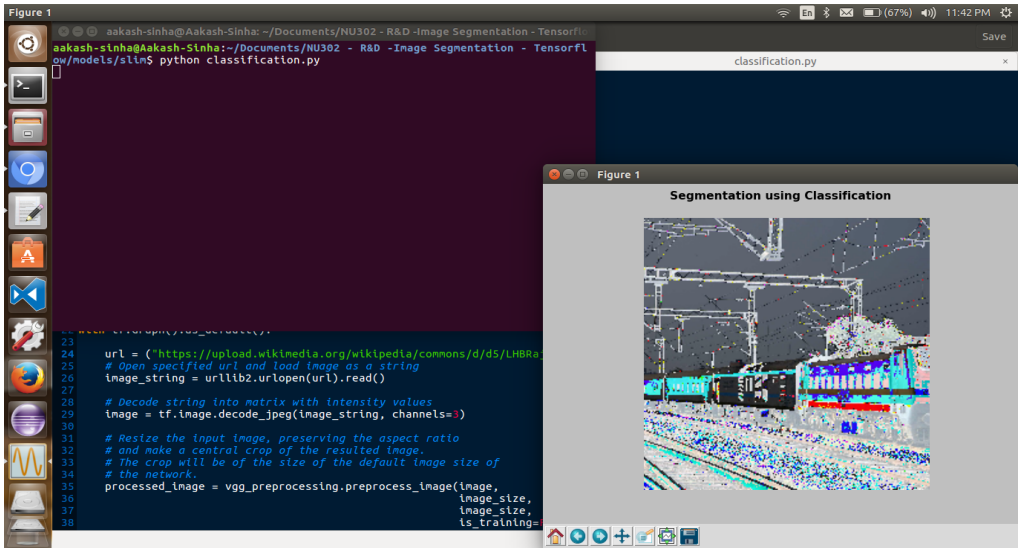
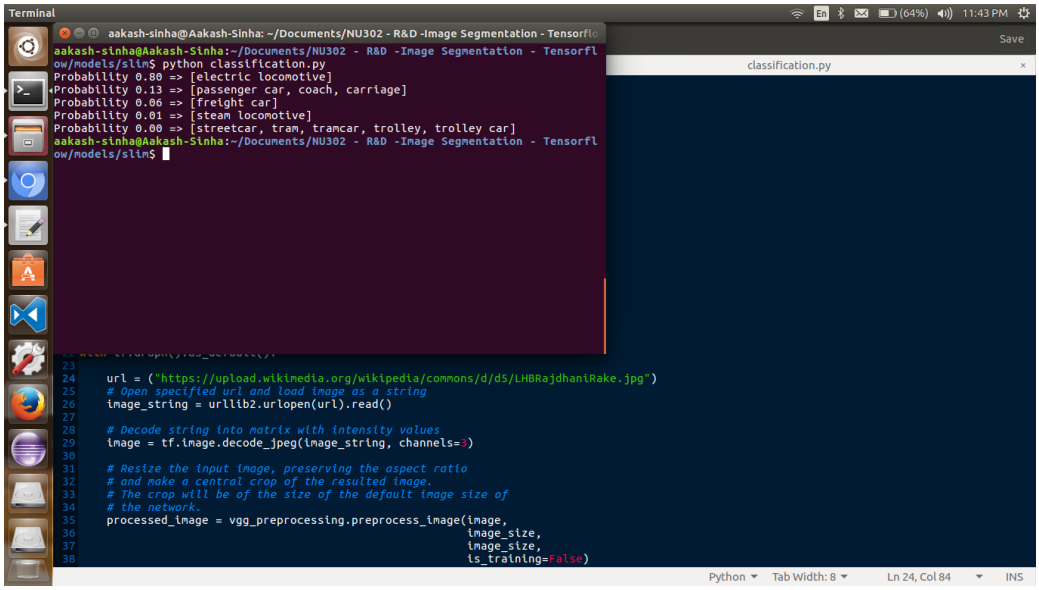


Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorfl
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorfl
ow/models/slim$ python classification.py
Save
classification.py
Figure 1
Segmentation using Classification
url = ("https://upload.wikimedia.org/wikipedia/commons/d/d5/LHBRa
# Open specified url and load image as a string
image_string = urllib2.urlopen(url).read()
# Decode string into matrix with intensity values
image = tf.image.decode_jpeg(image_string, channels=3)
# Resize the input image, preserving the aspect ratio
# and make a central crop of the resulted image.
# The crop will be of the size of the default image size of
# the network.
processed_image = vgg_preprocessing.preprocess_image(image,
image_size,
image_size,
is_training=


Terminal
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorfl
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorfl
ow/models/slim$ python classification.py
Probability 0.80 => [electric locomotive]
Probability 0.13 => [passenger car, coach, carriage]
Probability 0.06 => [freight car]
Probability 0.01 => [steam locomotive]
Probability 0.00 => [streetcar, tram, tramcar, trolley, trolley car]
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorfl
ow/models/slim$
Save
classification.py
url = ("https://upload.wikimedia.org/wikipedia/commons/d/d5/LHBRajdhaniRake.jpg")
# Open specified url and load image as a string
image_string = urllib2.urlopen(url).read()
# Decode string into matrix with intensity values
image = tf.image.decode_jpeg(image_string, channels=3)
# Resize the input image, preserving the aspect ratio
# and make a central crop of the resulted image.
# The crop will be of the size of the default image size of
# the network.
processed_image = vgg_preprocessing.preprocess_image(image,
image_size,
image_size,
is_training=False)
Python
Tab Width: 8
Ln 24, Col 84
INS

Example 3 :

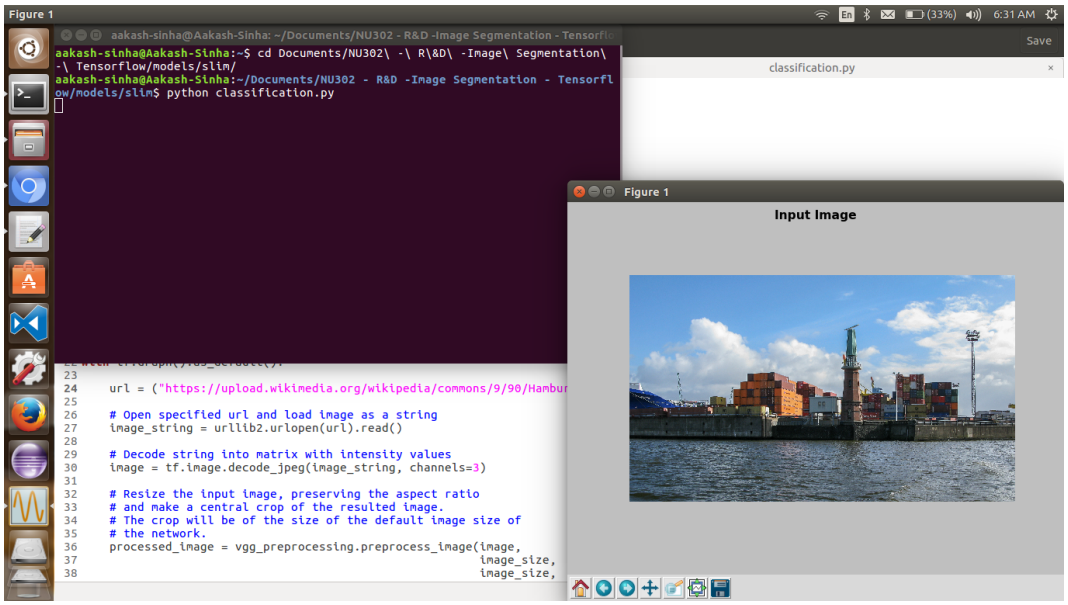


Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorflow
aakash-sinha@Aakash-Sinha:~$ cd Documents/NU302\ -\ R\&D\ -Image\ Segmentation\ -\ Tensorflow/models/slim/
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorflow/models/slim$ python classification.py
Save
classification.py
Figure 1
Input Image
url = ("https://upload.wikimedia.org/wikipedia/commons/9/90/Hambur
# Open specified url and load image as a string
image_string = urllib2.urlopen(url).read()
# Decode string into matrix with intensity values
image = tf.image.decode_jpeg(image_string, channels=3)
# Resize the input image, preserving the aspect ratio
# and make a central crop of the resulted image.
# The crop will be of the size of the default image size of
# the network.
processed_image = vgg_preprocessing.preprocess_image(image,
image_size,
image_size,

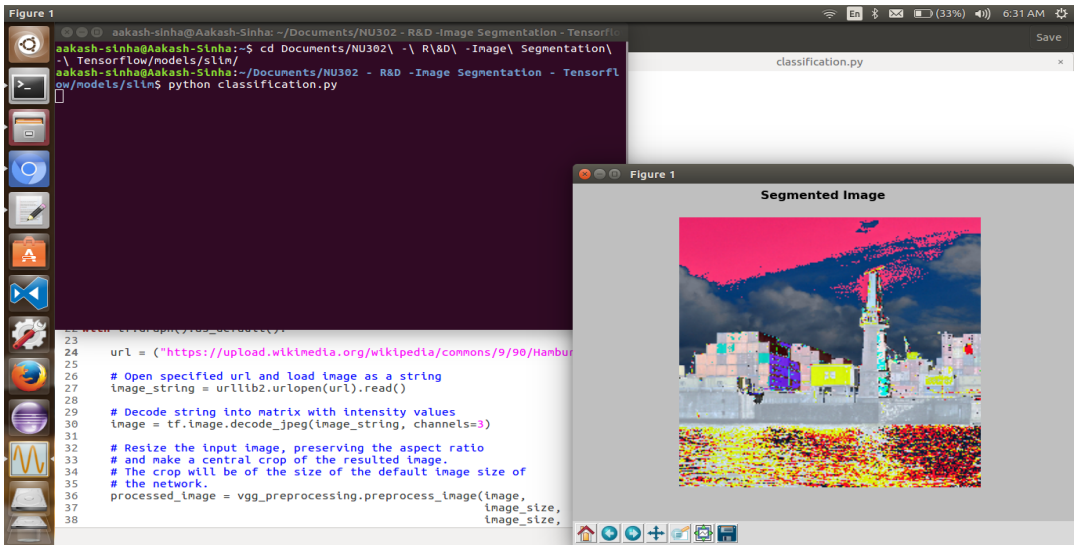
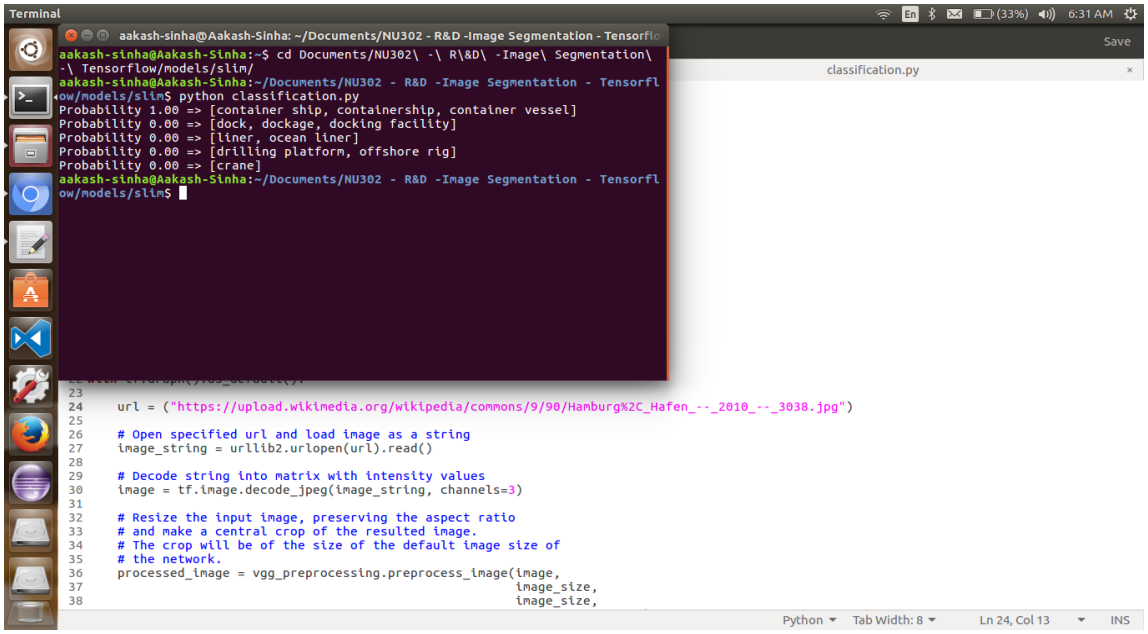


Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/NU302 - R&D -Image Segmentation - Tensorflow
aakash-sinha@Aakash-Sinha:~$ cd Documents/NU302\ -\ R\&D\ -Image\ Segmentation\ -\ Tensorflow/models/slim/
aakash-sinha@Aakash-Sinha:~/Documents/NU302 - R&D -Image Segmentation - Tensorflow/models/slim$ python classification.py
Save
classification.py
Figure 1
Segmented Image
url = ("https://upload.wikimedia.org/wikipedia/commons/9/90/Hambur
# Open specified url and load image as a string
image_string = urllib2.urlopen(url).read()
# Decode string into matrix with intensity values
image = tf.image.decode_jpeg(image_string, channels=3)
# Resize the input image, preserving the aspect ratio
# and make a central crop of the resulted image.
# The crop will be of the size of the default image size of
# the network.
processed_image = vgg_preprocessing.preprocess_image(image,
image_size,
image_size,



Pros :

- This program can be applied to a wide variety of images
- Gives results with higher accuracy than the traditional methods discussed earlier.
- It is faster and lighter than other methods implemented on various other machine learning libraries.
- Fully utilizes the capacity of the system reducing the human effort and minimizing chances of errors

Cons :

- Though it has high degree of accuracy, there is still a possibility of images not getting segmented properly
- The decision on training the machine on how many classes is a problem
- Finding the optimum learning rate is another big challenge
- There always exists a possibility of overtraining or undertraining the machine
- It cannot always identify the objects correctly

Image Segmentation using FCN + CRF as RNN

Dependencies of the Project :

- **Tensor Flow v0.12 :**

  We have used an Upgraded version of Tensor-Flow, because it is provided with a more stable build and the first time an RNN Cell is used, it caches its scope. All future uses of the RNN Cell will reuse variables from that same scope. This is a breaking change from the behavior of RN Cells in Tensor Flow versions.

- **VGG 16s Model :**

  VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet , which is a dataset of over 14 million images belonging to 1000 classes.

- **FCN 8s NET :**

  It is an 8 Layer Fully Convolutional Network which has been trained on a VGG-16 model to give precised output results in

- **CUDA :**

  CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).Most related work involving GPUs for segmentation is in the medical imaging domain, where the extra dimension of data (a volume instead of an image) has made speed a requirement rather than an option.

- **Scikit image library :**

  Scikit-image is an open source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

- **PASCAL VOC 2012 :**

  Table summarizes the number of objects and images (containing at least one object of a given class) for each class and image set.

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Aeroplane	88	108	90	110	178	218	-	-
Bicycle	65	94	79	103	144	197	-	-
Bird	105	137	103	140	208	277	-	-
Boat	78	124	72	108	150	232	-	-
Bottle	87	195	96	162	183	357	-	-
Bus	78	121	74	116	152	237	-	-
Car	128	209	127	249	255	458	-	-
Cat	131	154	119	132	250	286	-	-
Chair	148	303	123	245	271	548	-	-
Cow	64	152	71	132	135	284	-	-
Diningtable	82	86	75	82	157	168	-	-
Dog	121	149	128	150	249	299	-	-
Horse	68	100	79	104	147	204	-	-
Motorbike	81	101	76	103	157	204	-	-
Person	442	868	445	865	887	1733	-	-
Pottedplant	82	151	85	171	167	322	-	-
Sheep	63	155	57	153	120	308	-	-
Sofa	93	103	90	106	183	209	-	-
Train	83	96	84	93	167	189	-	-
Tvmonitor	84	101	74	98	158	199	-	-
Total	1464	3507	1449	3422	2913	6929	-	-

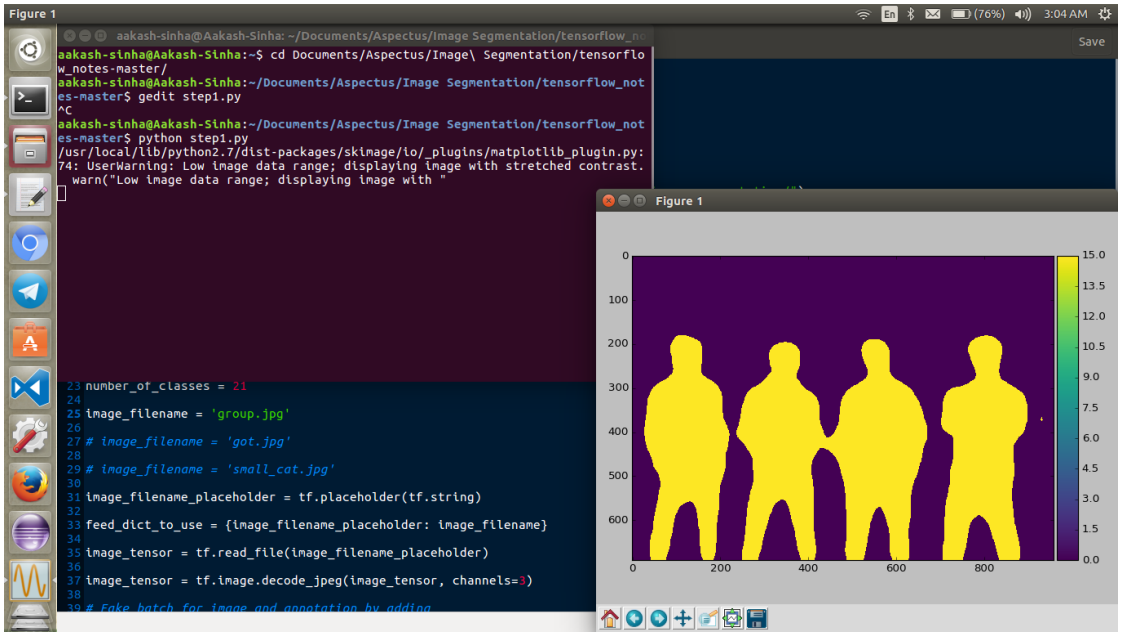
Using this image dataset which was used in PASCAL VOC 2012 competition we have trained our FCN 8s net for the given 21 classes.

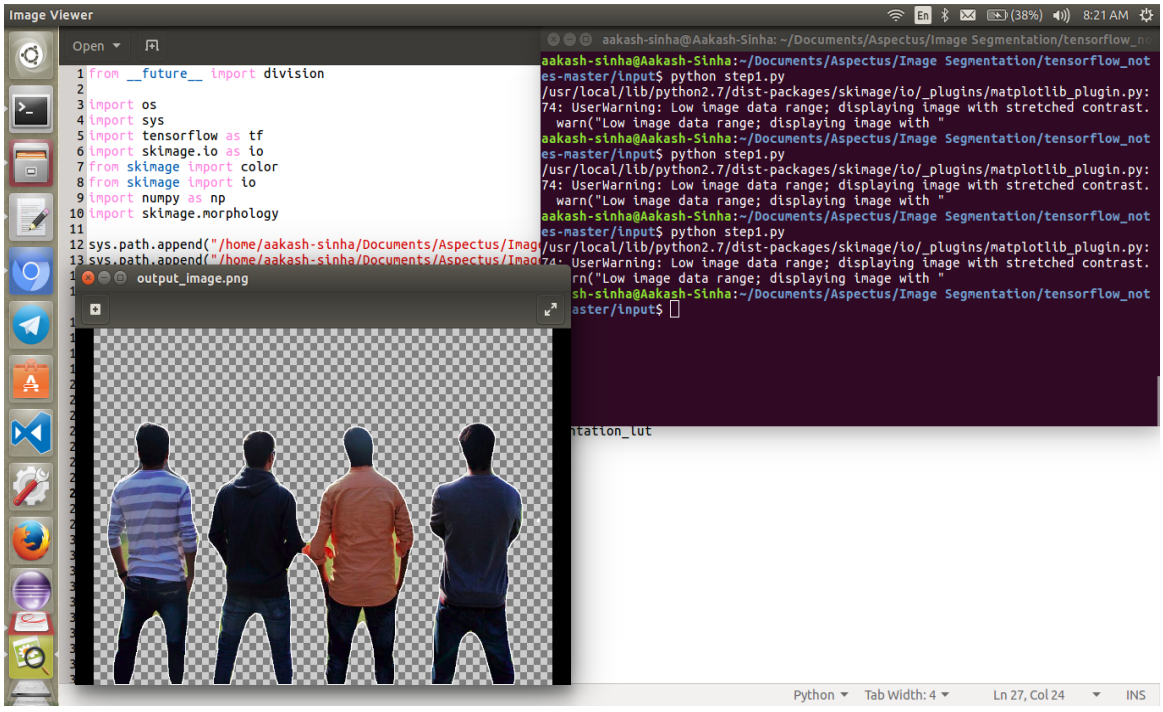
PROCEDURE :

- Import all the dependencies
- Input an image
- Load the tensors
- FCN-8s will resize the image and map it to every classifier.
- Create a session
- CRF is generated on a RNN to give a heatmap of segmented image
- Apply morphological operations for generating contour
- Use a prediction mask to retrieve only the foreground object
- Save the final segmented image in output folder
- Run a bash file to convert and resize the image into 512x512 .png
- Import the file to a custom batch of stickers
- Chat like a pro with our stickers!

Results of Image Segmentation :

Example 1 :





Example 2 :

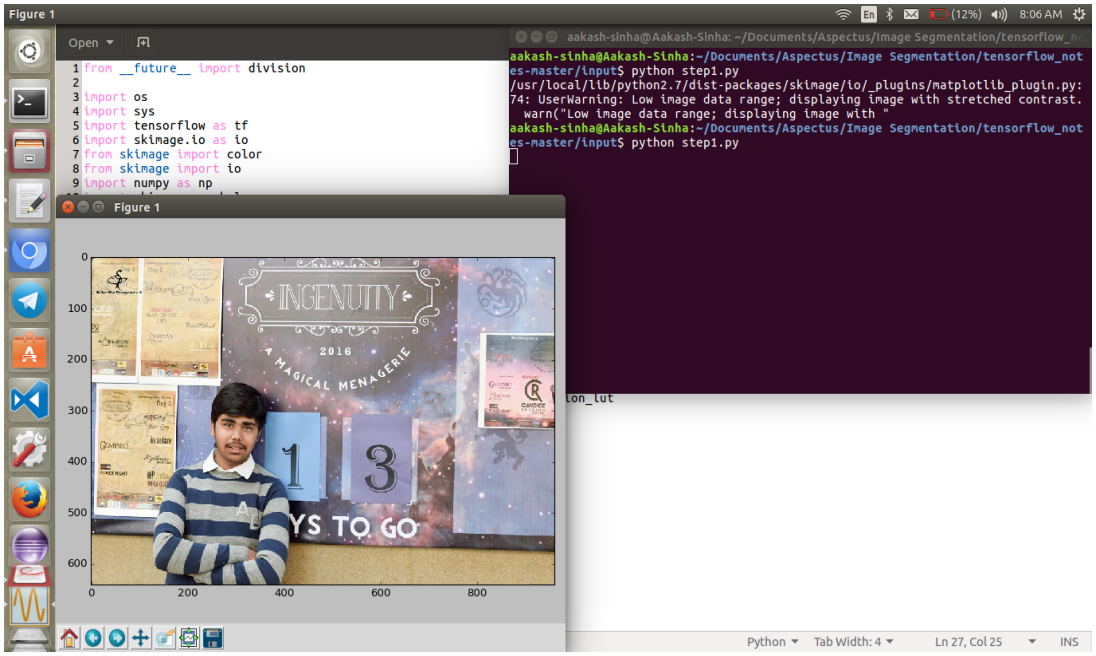


Figure 1
Open
from __future__ import division
import os
import sys
import tensorflow as tf
import skimage.io as io
from skimage import color
from skimage import io
import numpy as np
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
/usr/local/lib/python2.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:
74: UserWarning: Low image data range; displaying image with stretched contrast.
warn("Low image data range; displaying image with "
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
Figure 1
INGENUITY
2016
A MAGICAL MENAGERIE
Python
Tab Width: 4
Ln 27, Col 25
INS

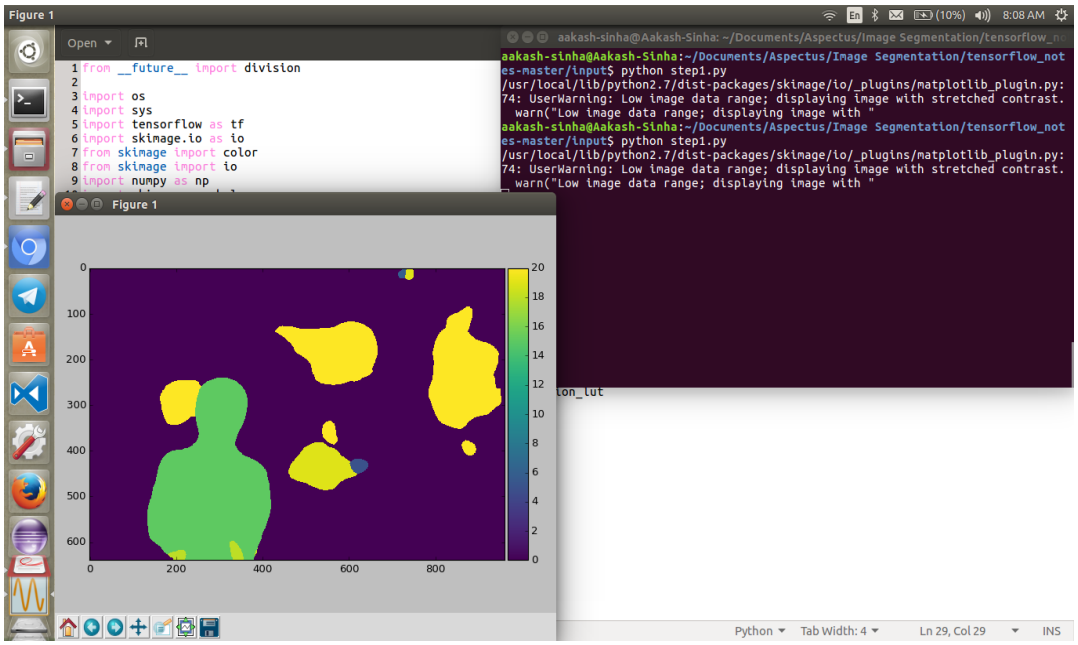


Figure 1
Open
from __future__ import division
import os
import sys
import tensorflow as tf
import skimage.io as io
from skimage import color
from skimage import io
import numpy as np
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
/usr/local/lib/python2.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:
74: UserWarning: Low image data range; displaying image with stretched contrast.
warn("Low image data range; displaying image with "
Figure 1
Python
Tab Width: 4
Ln 29, Col 29
INS

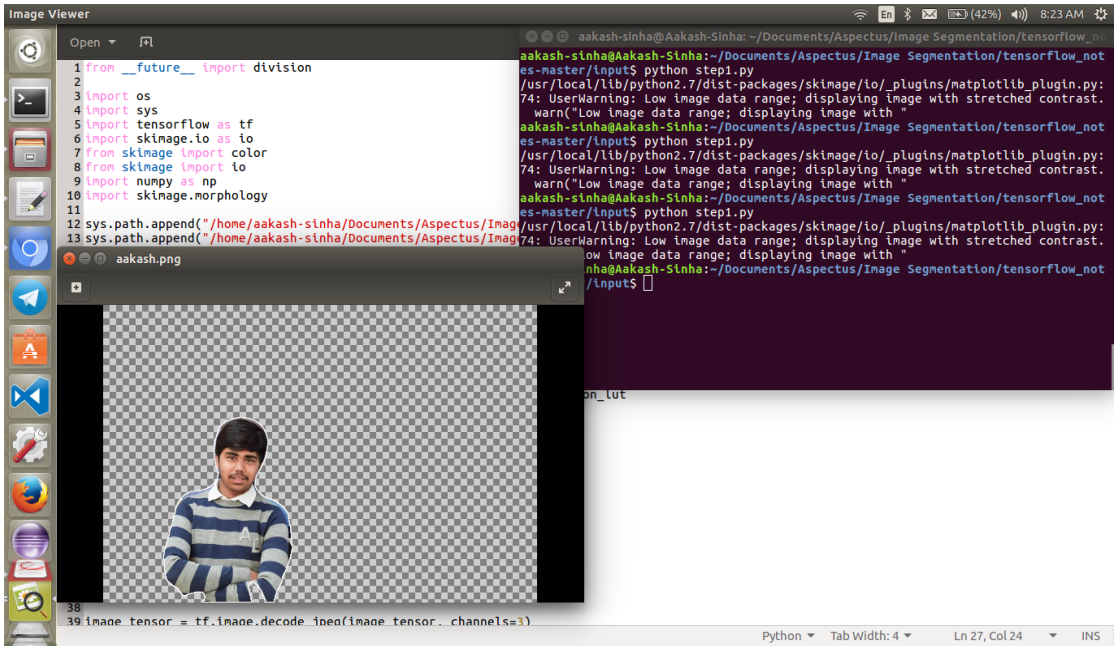


Image Viewer
Open
aakash.png
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
/usr/local/lib/python2.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:
74: UserWarning: Low image data range; displaying image with stretched contrast.
warn("Low image data range; displaying image with "
Python
Tab Width: 4
Ln 27, Col 24
INS

Example 3 :

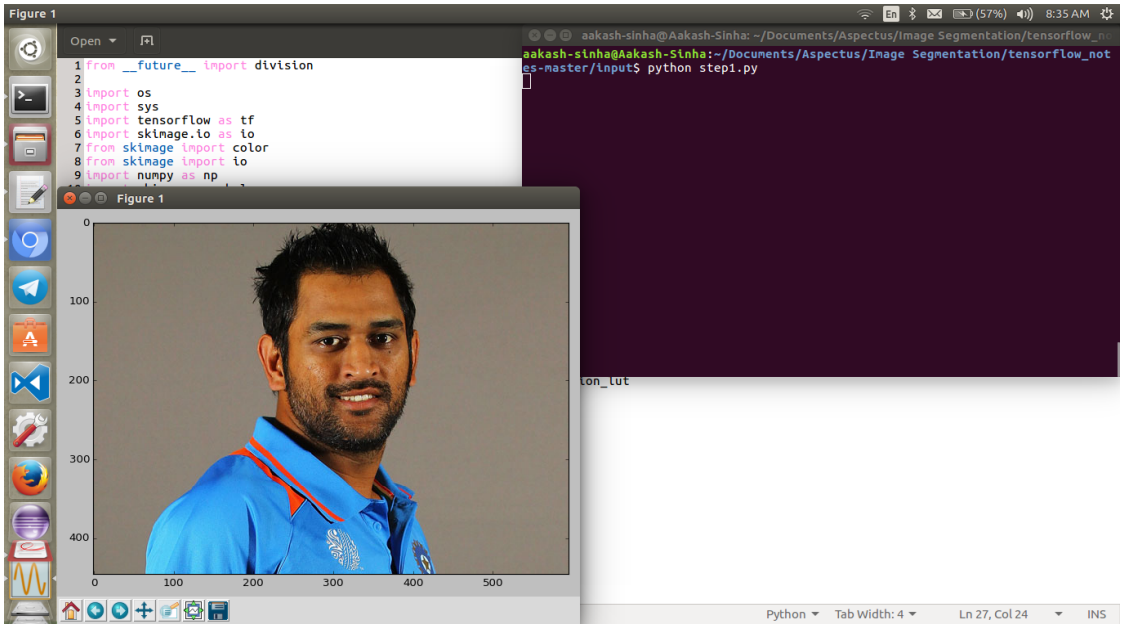


Figure 1
Open
Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
Python
Tab Width: 4
Ln 27, Col 24
INS

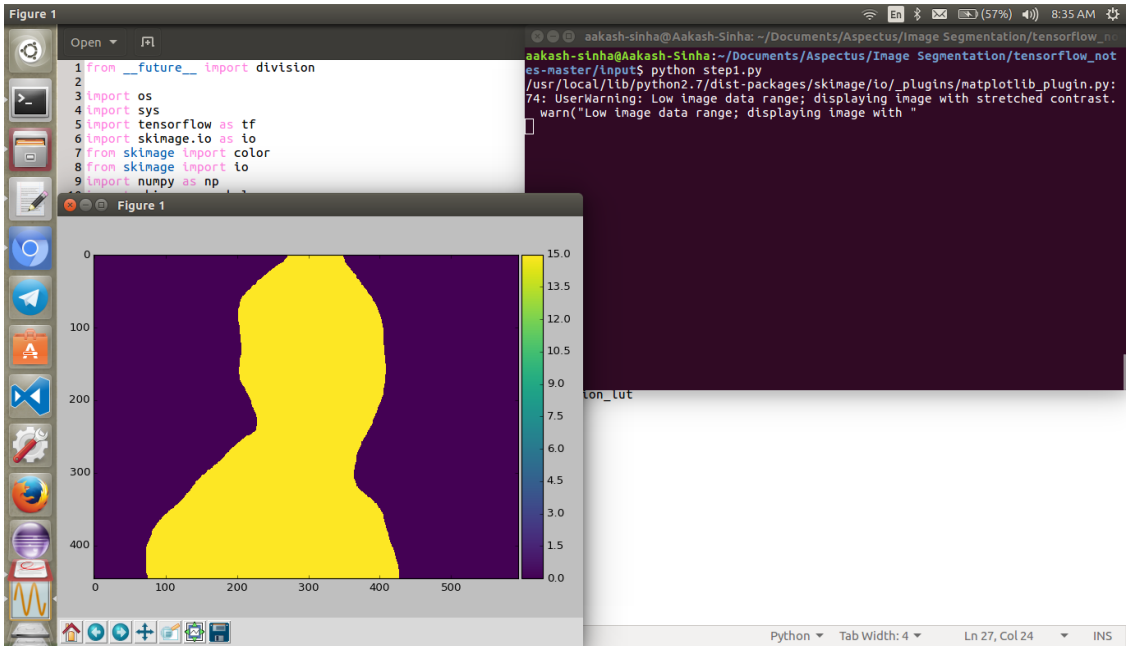


Figure 1
Open
from __future__ import division
import os
import sys
import tensorflow as tf
import skimage.io as io
from skimage import color
from skimage import io
import numpy as np
Figure 1
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
/usr/local/lib/python2.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:
74: UserWarning: Low image data range; displaying image with stretched contrast.
warn("Low image data range; displaying image with "
Python
Tab Width: 4
Ln 27, Col 24
INS

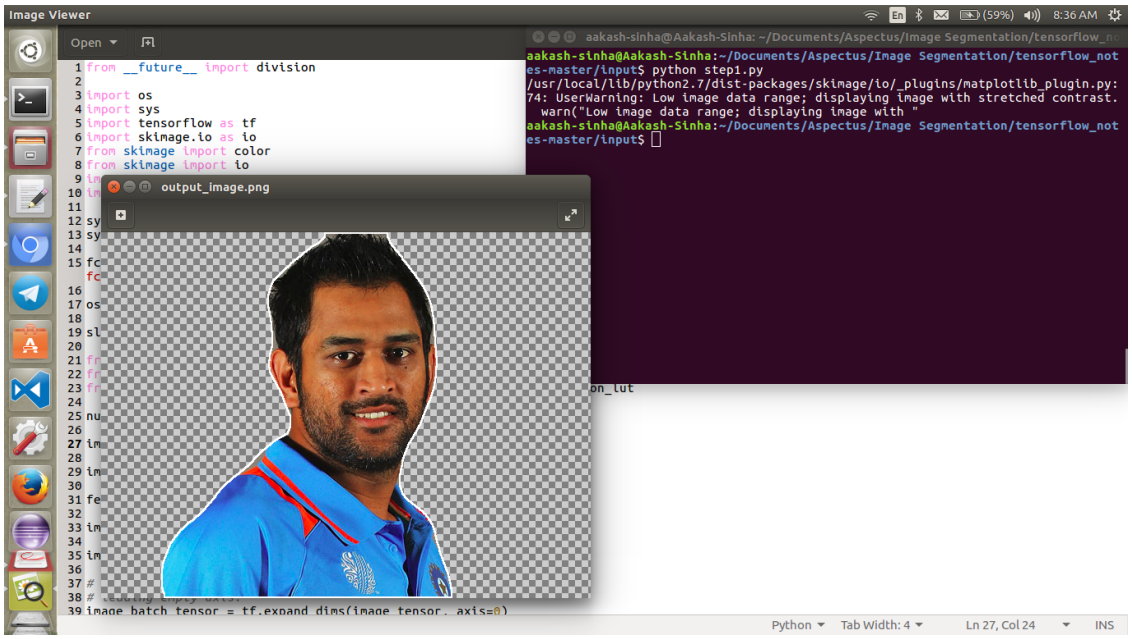


Image Viewer
Open
from __future__ import division
import os
import sys
import tensorflow as tf
import skimage.io as io
from skimage import color
from skimage import io
output_image.png
aakash-sinha@Aakash-Sinha: ~/Documents/Aspectus/Image Segmentation/tensorflow_not
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$ python step1.py
/usr/local/lib/python2.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:
74: UserWarning: Low image data range; displaying image with stretched contrast.
warn("Low image data range; displaying image with "
aakash-sinha@Aakash-Sinha:~/Documents/Aspectus/Image Segmentation/tensorflow_not
es-master/input$
Python
Tab Width: 4
Ln 27, Col 24
INS

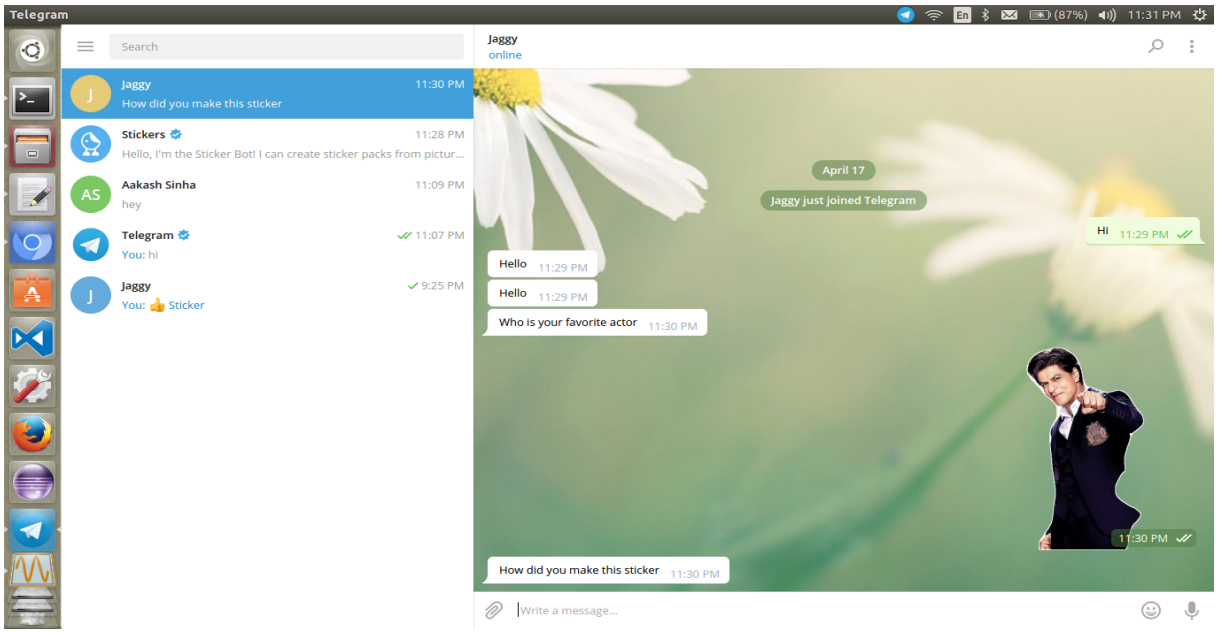
Telegram

Extending segmented image to create a sticker in Telegram :

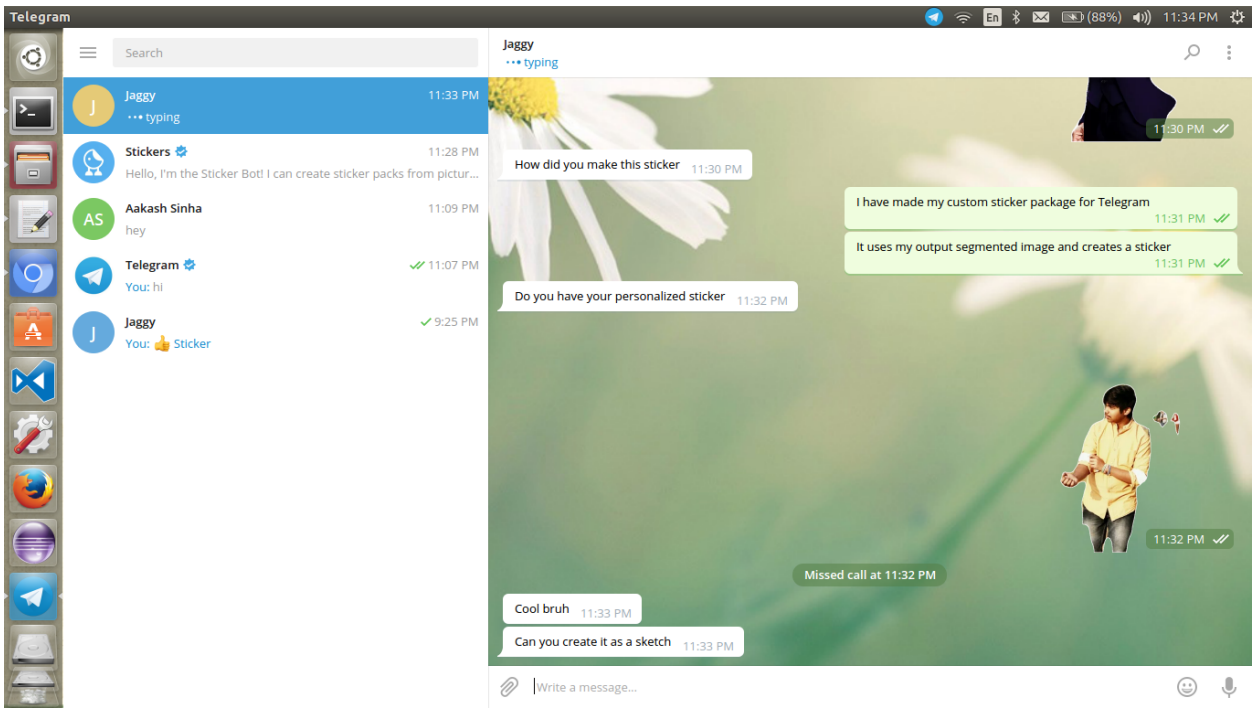
Telegram provides a very interactive bot also known as sticker bot who accepts the image file which should be in PNG format with a transparent layer and must fit into a 512x512 square (one of the sides must be 512px and the other 512px or less).
Our image segmenter code is returning us only the part of image which is foreground which is accurately classified as one of the 21 classes and is present in the maximum portion of the image, The beauty of our code is that it saves the copy of resultant segmented image in a different folder and the image is with a transparent film.
The problem now is that the resultant or extracted image is of the same resolutions as that of the original image which will never be directly accepted by sticker bot as it does not fit in the condition of size.
Therefore our team have made a code to automate the resizing of the image which is otherwise done manually. The resizing code takes an image as input and shrinks it to the specified size 512x512 here, for it to be acceptable by sticker bot.
Now after this we are ready to go provide the refined and resized image to sticker bot for it to make the segmented part of image as sticker.

Results :

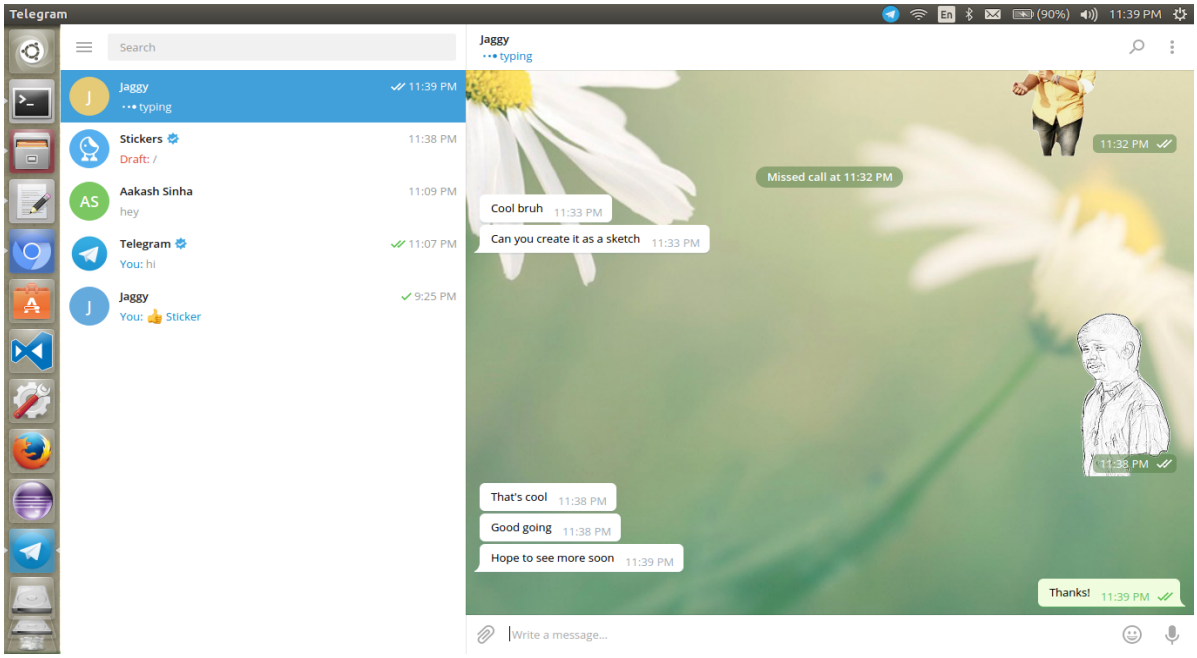
Example 1 :



Example 2 :

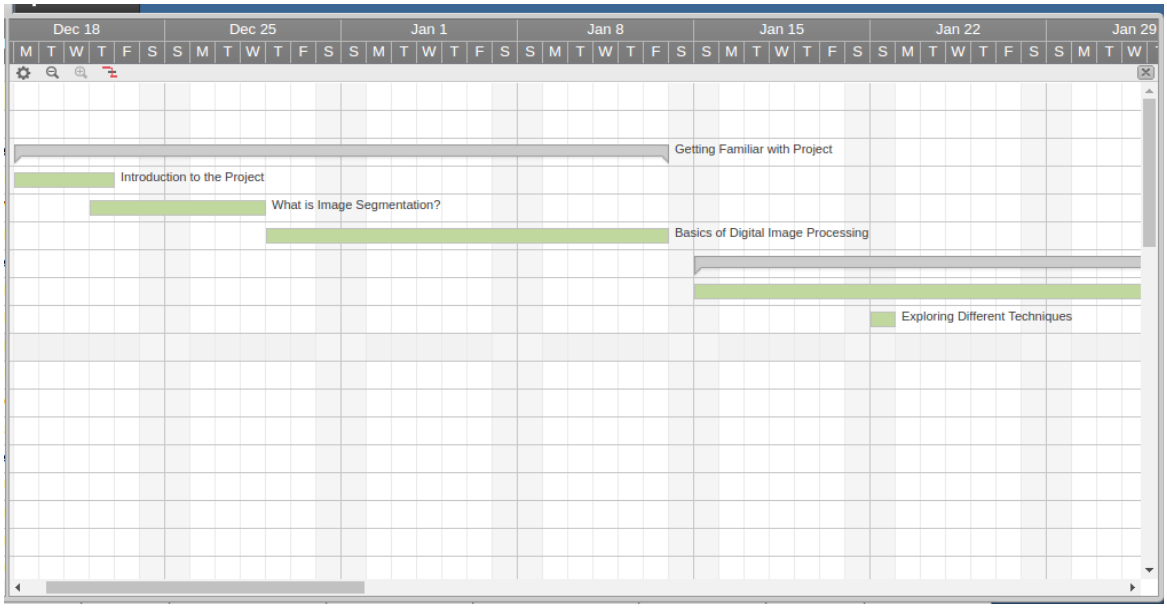


Example 3 :

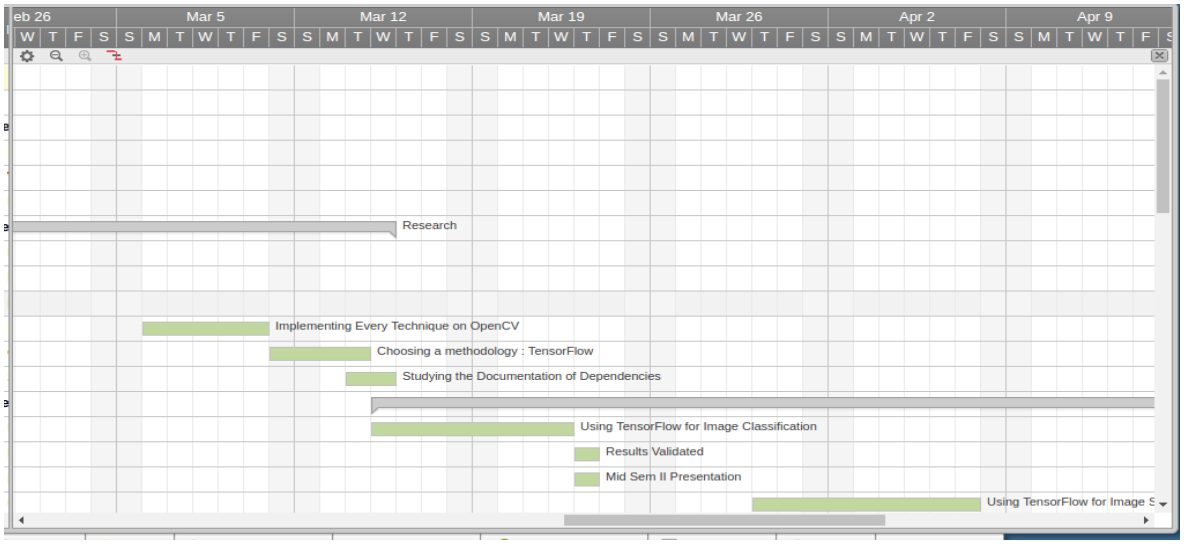

Telegram
Search
Jaggy
typing
11:39 PM
Stickers
Draft: /
11:38 PM
Aakash Sinha
hey
11:09 PM
Telegram
You: hi
11:07 PM
Jaggy
You: Sticker
9:25 PM
Jaggy
typing
11:32 PM
Missed call at 11:32 PM
Cool bruh 11:33 PM
Can you create it as a sketch 11:33 PM
11:38 PM
That's cool 11:38 PM
Good going 11:38 PM
Hope to see more soon 11:39 PM
Thanks! 11:39 PM
Write a message...

Project Timeline :

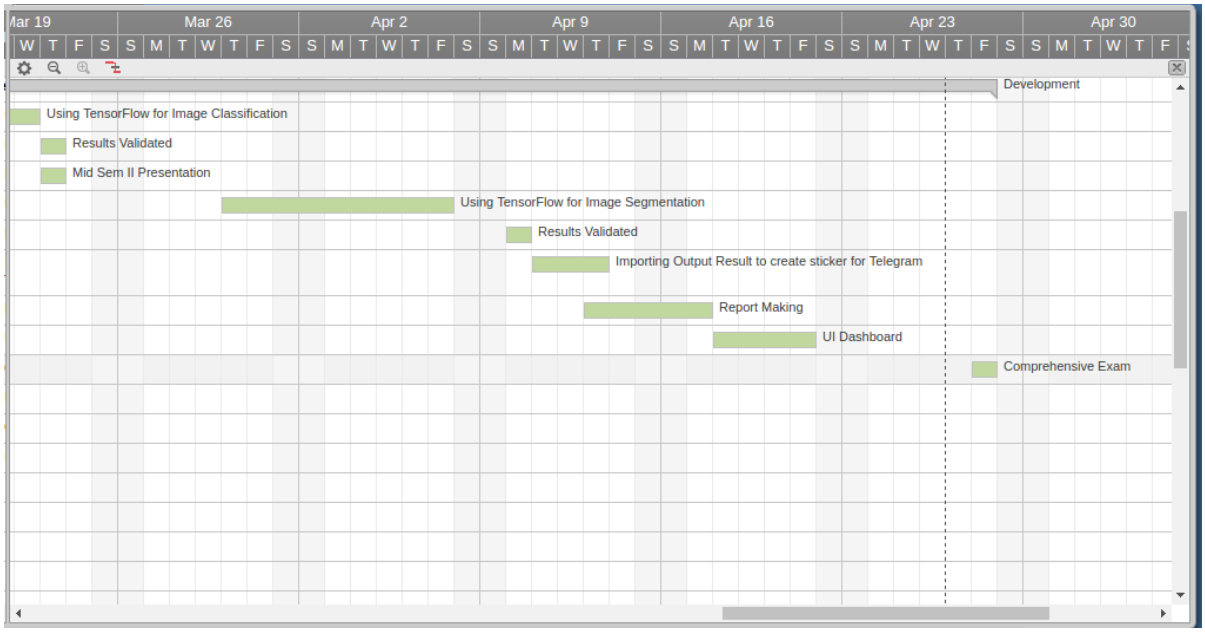
Phase 1 : Getting Familiar With Project



Phase 2 : Research



Phase 3 : Development



Conclusion :

- The output results generated were very accurate and efficient using TensorFlow.
- The Prime Objective of Improvement of Image Segmentation was achieved since we implemented several algorithms and we could get the best and efficient result using FCN + CRF as RNN.
- We could use the results of the segmented image in Telegram Chat Box as stickers.

References :

1. Digital Image Processing (by Gonzalez) 3rd Edition

2. Contour Detection and Image Segmentation (UC, Berkely)
http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2009-129.pdf

3. Various Segmentation Methods - 2014 - Dilprit and Jaspritender Kaur
http://ijcsmc.com/docs/papers/May2014/V3I5201499a84.pdf

4. Semantic Segmentation using Neural Network - 2015 - Sadeep Jayasumna
https://wwwf.imperial.ac.uk/~gmontana/talks/crfasrnn_presentation.pdf

5. Tensor Flow v0.12 : https://www.tensorflow.org/

6. VGG 16s Model : http://www.robots.ox.ac.uk/~vgg/

7. CUDA : http://www.nvidia.com/object/cuda_home_new.html

8. PASCAL VOC 2012 : http://host.robots.ox.ac.uk/pascal/VOC/

9. Scikit image library : http://scikit-image.org

10. Slim Framework : https://www.slimframework.com/