

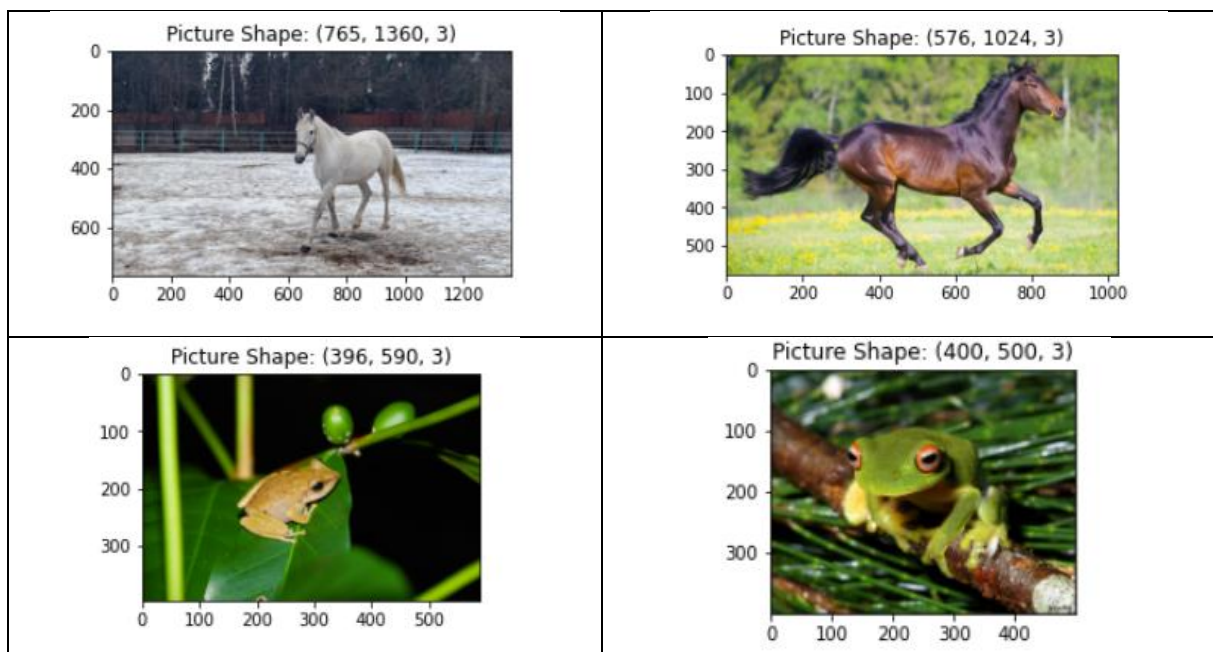
# HW 3 - Segmentation

Done with *colab* and then translated to *Jupyter*. Indexing results may vary if re-running notebook

## 1 Part 1: Classic Vs. Deep Learning-based Semantic Segmentation

### 1.1 Question 1

We were to load the frogs and horses' pictures provided.



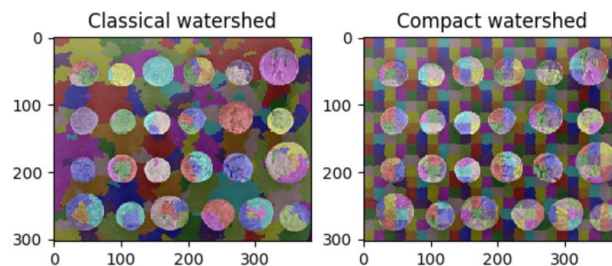
## 1.2 Question 2

We were to segment the horses and frogs pictures with 1 classic method for segmentation and 1 deep learning based method and discuss the results.









### 1.2.1 Classic Methods

We first applied 4 methods (using scikit-image) and optimized their parameters for each set of pictures:

- Felzenszwalb's and huttenlocher's
- SLIC
- Quickshift – fast mode seeking algorithm like *mean shift* which was learned in the classroom. Quickshift constructs a tree connecting each image pixel to its nearest neighbor which has the greater density value. Each pixel is connected to the closest higher density pixel that achieves a minimum on a distance function.
- Compact Watershed – an upgrade to the *classic watershed* which has a problem with over segmentation. This method introduces a compactness parameter which allows for the creation of a weighted distance metric combining the conventional appearance-based distance and the Euclidean distance of the pixel to the segment seed. The result is a constraint on the size and elongation of the segments.



We then applied N-cuts to reduce the number of segments.

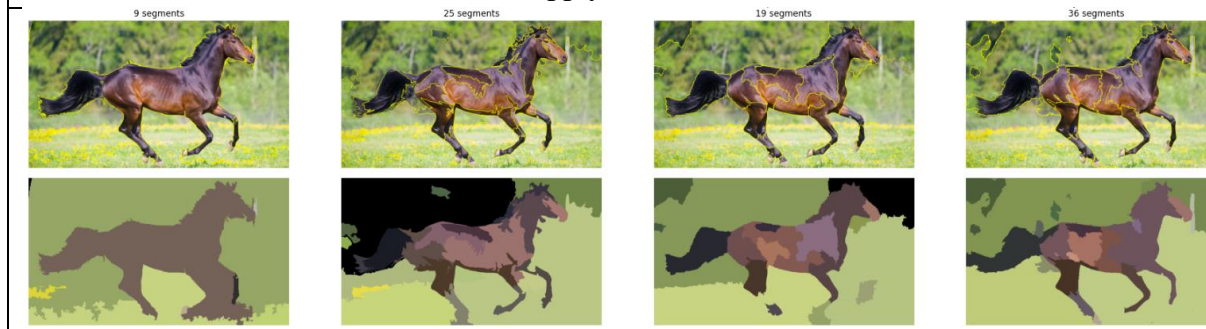
Felzenszwalb	SLIC	Quickshift	Compact Watershed
$scale = 1000$ $scale_{min} = 400$ $\sigma = 0.5$	$n_{segs} = 200$ $compactness = 10$ $\sigma = 1$	$kernel\ size = 7$ $max\ dist = 300$ $ratio = 0.3$	$markers = 200$ $compactnesses = 0.001$
90 segments  	140 segments  	266 segments  	209 segments  
Apply NCUT			



Felzenszwalb	SLIC	Quickshift	Compact Watershed
$scale = 1000$ $scale_{min} = 400$ $\sigma = 0.5$	$n_{segs} = 200$ $compactness = 10$ $\sigma = 1$	$kernel\ size = 7$ $max\ dist = 300$ $ratio = 0.3$	$markers = 200$ $compactnses = 0.001$

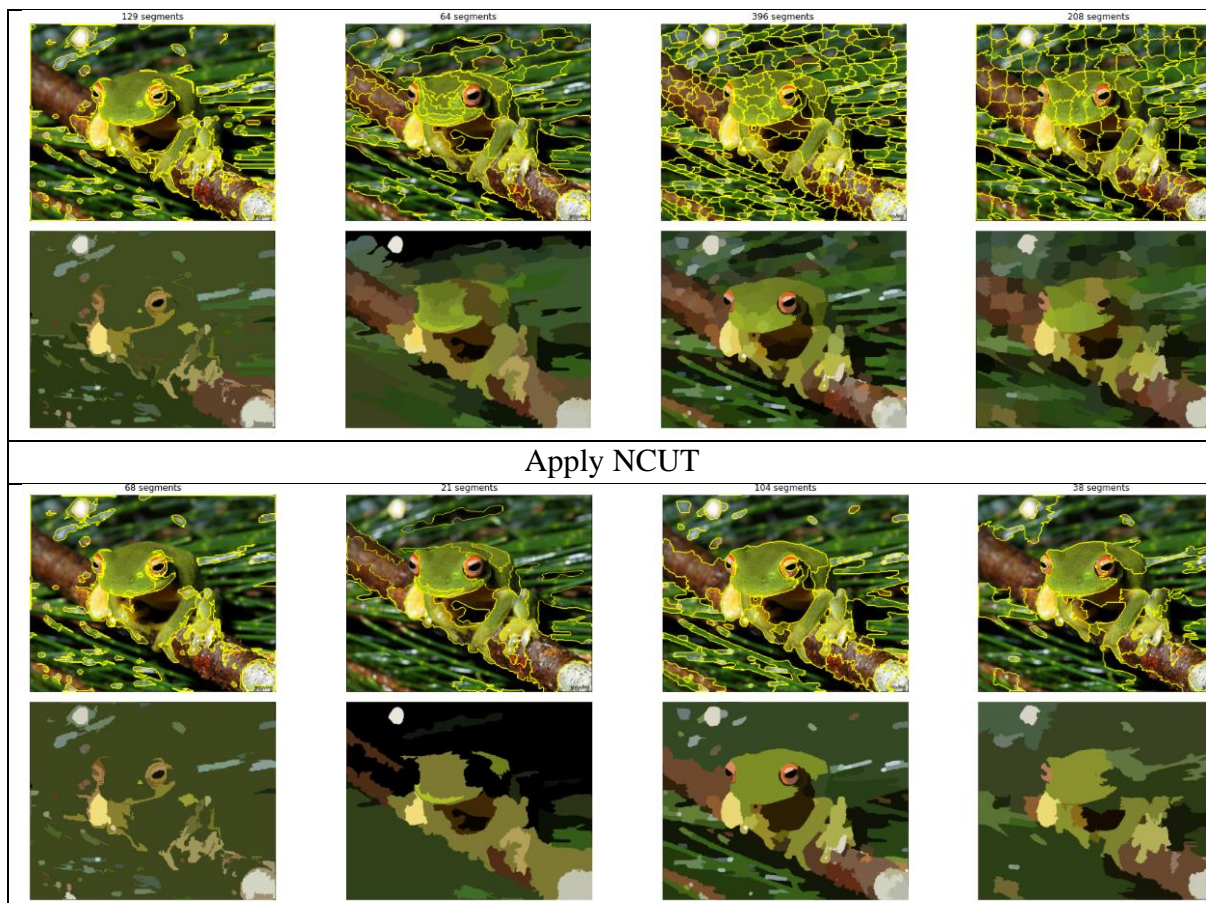










Apply NCUT

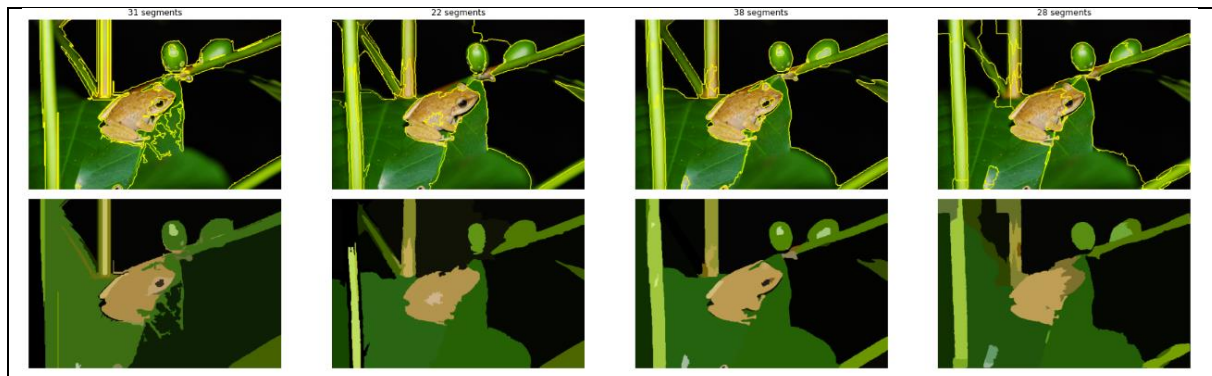


Felzenszwalb	SLIC	Quickshift	Compact Watershed
$scale = 400$ $scale_{min} = 50$ $\sigma = 1$	$n_{segs} = 200$ $compactness = 1$ $\sigma = 1$	$kernel\ size = 3$ $max\ dist = 10$ $ratio = 0.3$	$markers = 200$ $compactnses = 0.001$



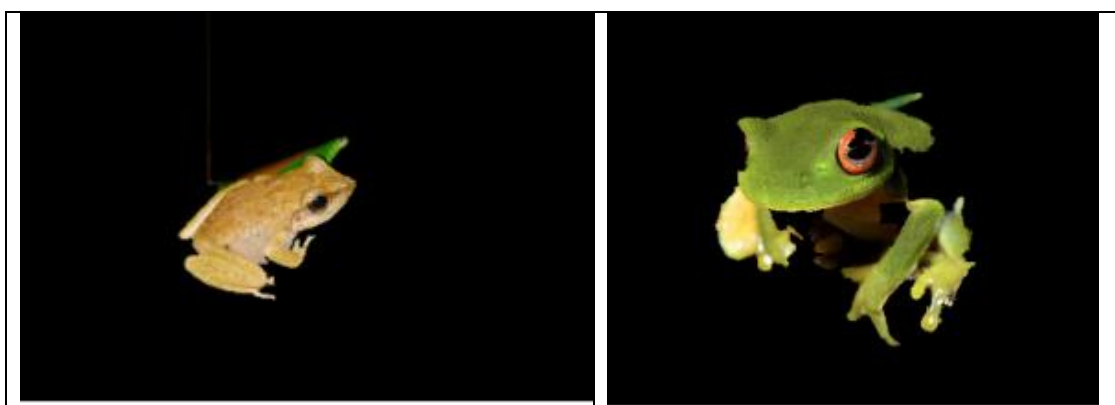


Felzenszwalb	SLIC	Quickshift	Compact Watershed
$scale = 400$ $scale_{min} = 50$ $\sigma = 1$	$n_{segs} = 200$ $compactness = 1$ $\sigma = 1$	$kernel\ size = 3$ $max\ dist = 10$ $ratio = 0.3$	$markers = 200$ $compactnses = 0.001$
81 segments  	107 segments  	320 segments  	204 segments  
Apply NCUT			



1. The first thing noticeable, is that for classic segmentation contrast is VITAL.
2. The trade-off is often between allowing irregular shapes for complex and varying geometries of the object of interest which also brings forth over segmentation, and over defining the segmentation which will cause a “miss”.
3. Also, right off the bat we noticed that Felzenszwalb’s and huttenlocher’s does a surprisingly good job! Especially for being the simplest and oldest of the algorithms. It is easily our favorite followed by compact watershed.
4. Worth noting that the Ncuts was “unstable” in the sense that it sometimes failed to converge. It did a really nice job unifying backgrounds but also unified some animal-background segments. Most notable on the compact-watershed white horse.

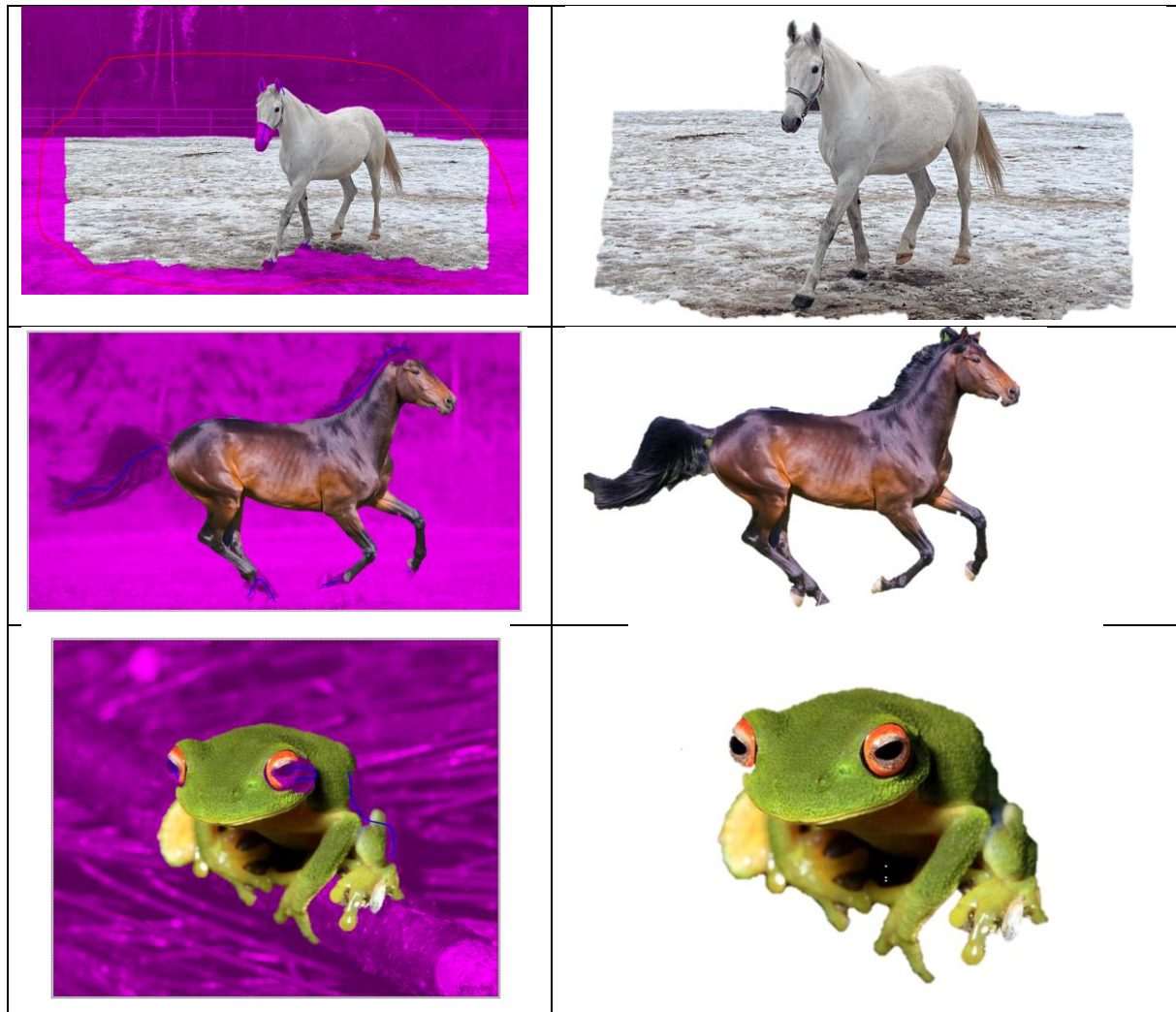
Choosing labels manually from each segmentation we obtained the following:



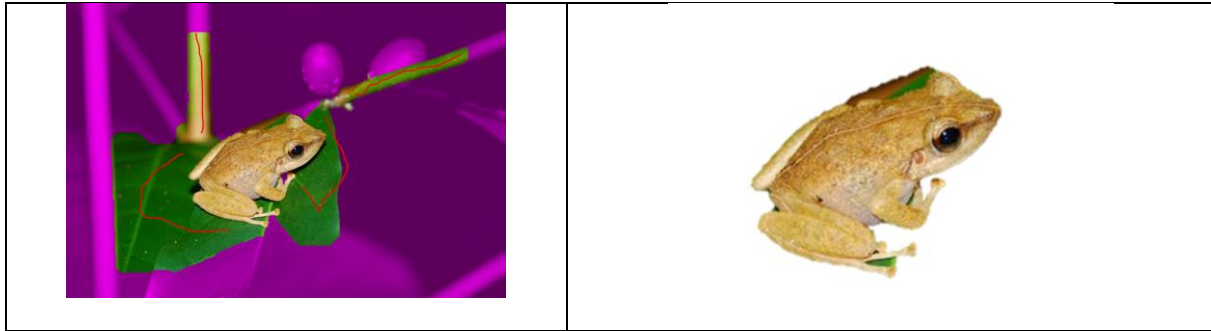


The next step towards automation should have been a clustering algorithm such as k-means followed by outputting a mask from a chosen label, but we deemed that too much work for an output that is already clarified.

So instead, we took the images to *PowerPoint* to check some *Lazy-Line* and *GrabCut* results.





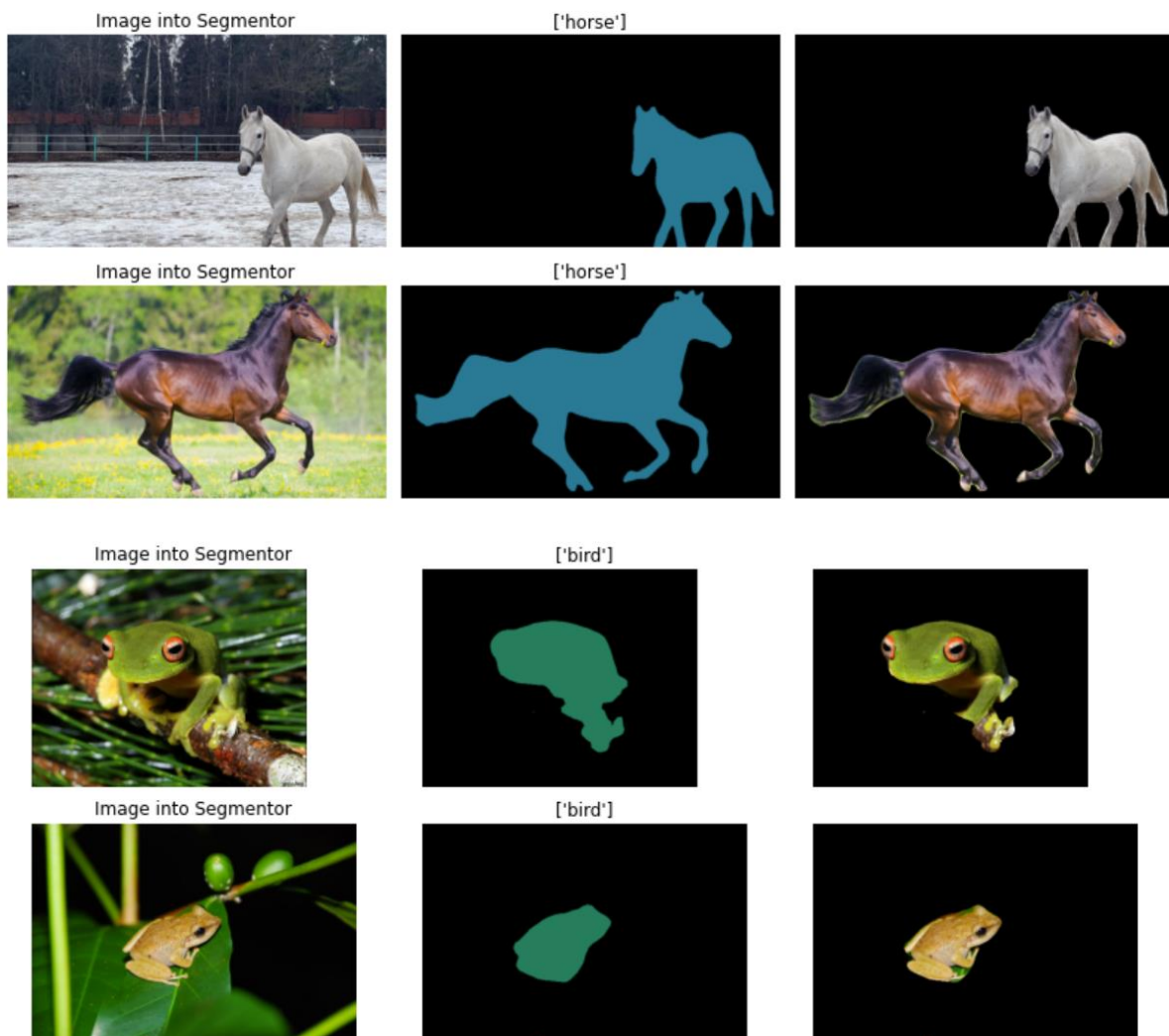


You have to give props to the Microsoft team for developing such an easy to use to algorithm which also produces such great results.

It would be interesting to check segmentation on a video with *GrabCut+LazyLine* where the user draws line on the first frame and through the video the lines are adjusted via an optimization algorithm as a preprocess to the *LazyLine*.

### 1.2.2 Deep Method

We used *deeplabv3\_resetnet101* to produce the following results:





This is clearly just plain out better than anything we did with the classic segmentation.

One would argue: why use classic segmentation at all?

One would also remember that it took a great work to optimize all the millions of parameters in *deeplabv3* and that the classic segmentation method above only require 2-3 parameters each.



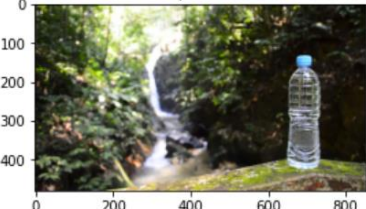
The classic segmentation methods are also not fitted to pre-defined objects.



### 1.3 Question 3

We were to pick 3 images and display them

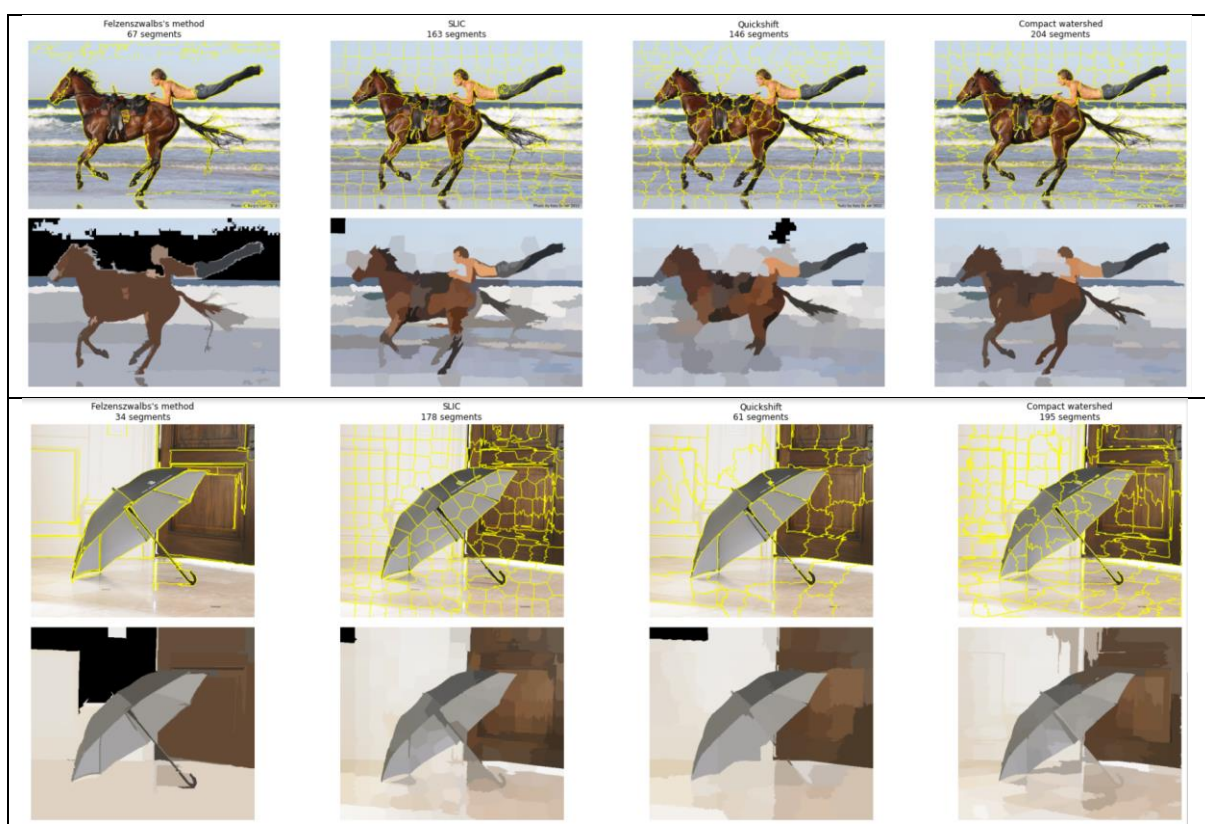
- Living being
- Commonly used object
- Not-so-commonly-used object

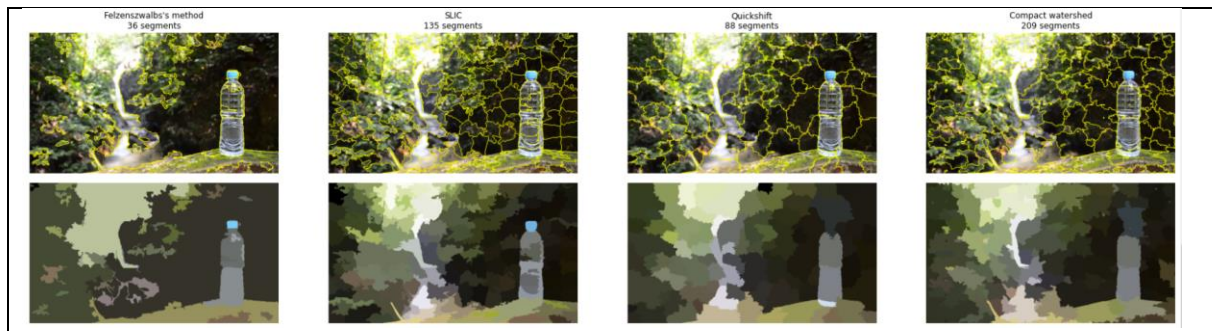
Not-so-commonly-used object	Living being	Commonly used object
<p>Picture Shape: (500, 580, 3)</p> 	<p>Picture Shape: (633, 950, 3)</p> 	<p>Picture Shape: (480, 852, 3)</p> 

### 1.4 Question 4

We were asked to apply the classic and deep methods on the chosen pictures and review.

#### 1.4.1 Classic Method





Choosing labels manually resulted in:

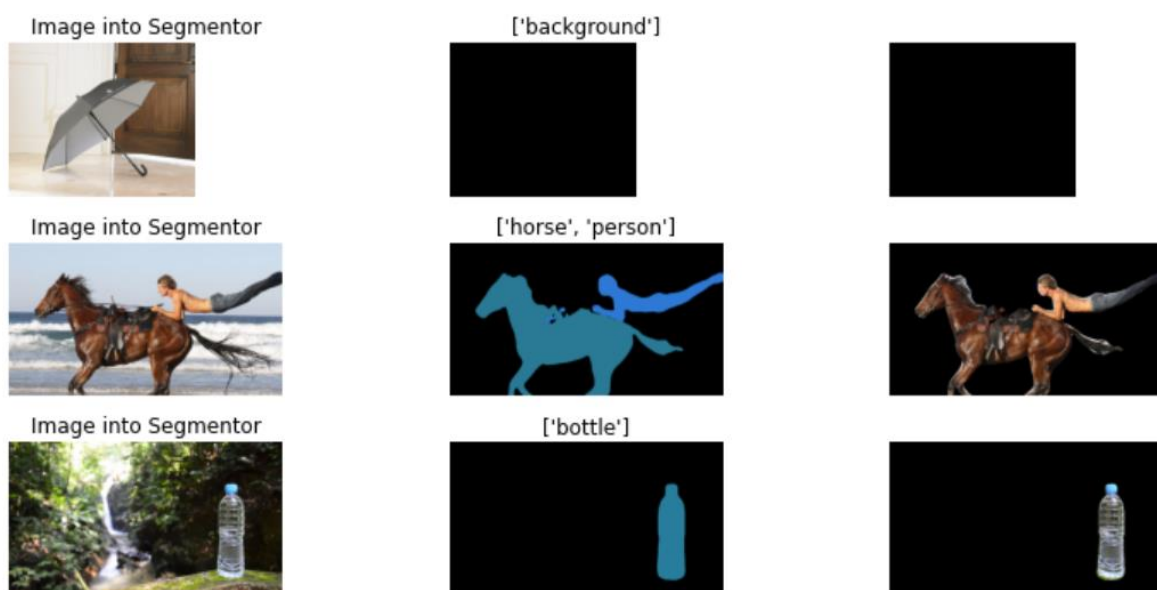


These results only affirm our insights from question 2:

- Thin objects such as the horse's legs, the person, and the umbrella's handle were sometimes swallowed up just as the bottle's top.
- The transparent bottle having two textures proved to be confusing to all algorithms.

### 1.4.2 Deep segmentation

Using *deeplabv3*:



The umbrella does not show up at all as the network was never trained on it.

As stated in question 2, deep segmentation is probably always preferred, but it is also a machine with millions of parameters calibrated for the task.

Classic segmentation requires only a few and can produce more than reasonable results.

### 1.5 Question 5

As segmentation can be sometimes rough around the edges, we were asked to view different approaches of fixing the problem – we chose preprocessing.

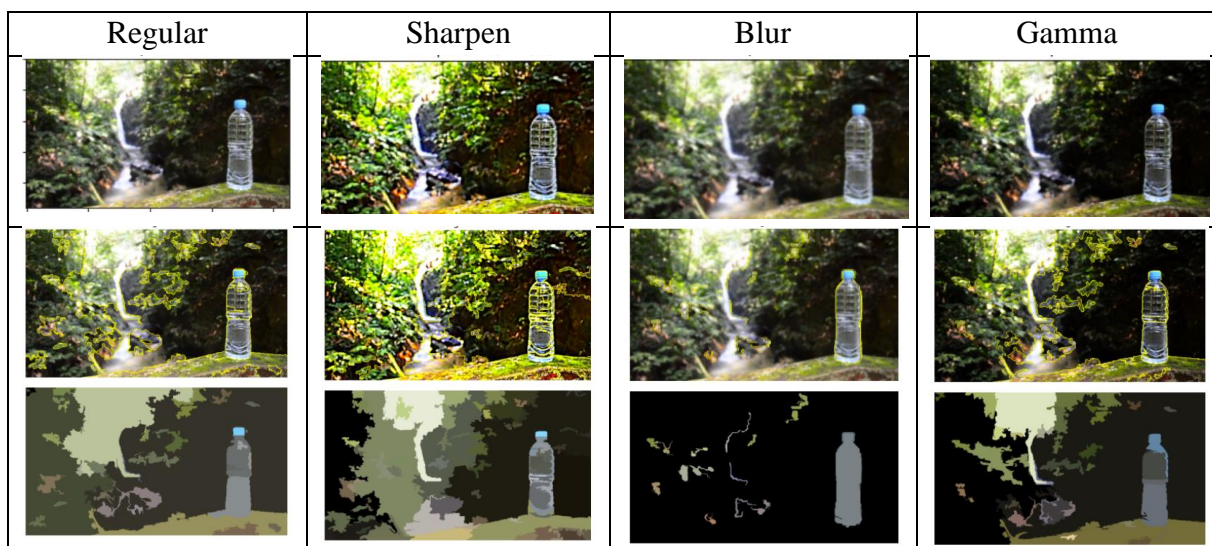
The intuition for classic segmentation follows on similarity between features for combining pixels and separation by boundaries for dis-association.

Hence, sharpening/blurring the images as a preprocessing step made a-lot of sense.

Also, because we noticed contrast has a distinguishable effect, we also decided to check if a gamma filter can improve the results.

#### 1.5.1 Classic

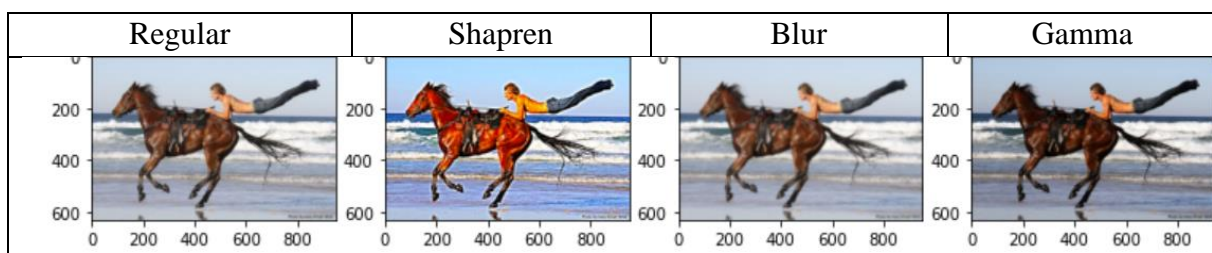
In the ipynb notebook you can find all methods used in Q2 and their results. We decided to show only Felzenswalbs's method in this report, and only for the bottle.



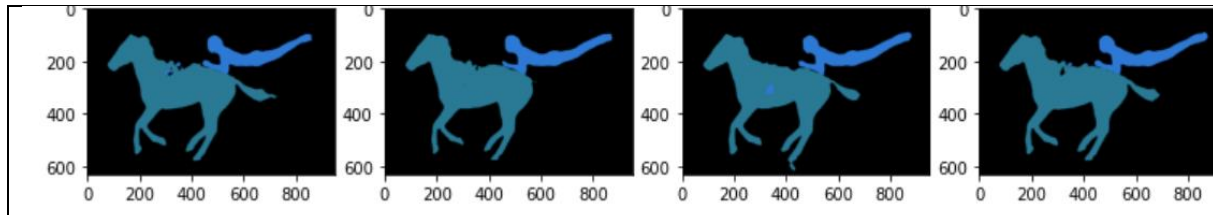
The only thing that seemed to work well was blurring the image, and that includes our deduction from the other images (not shown here) as-well.

#### 1.5.2 Deep

As the umbrella cannot be segmented by our deep network, and the bottle has too regular of a shape, we could only test for differences on the horse-person image.







Sharpening the image made us lose the horses tail, and blurring it shortened the tail surprisingly.

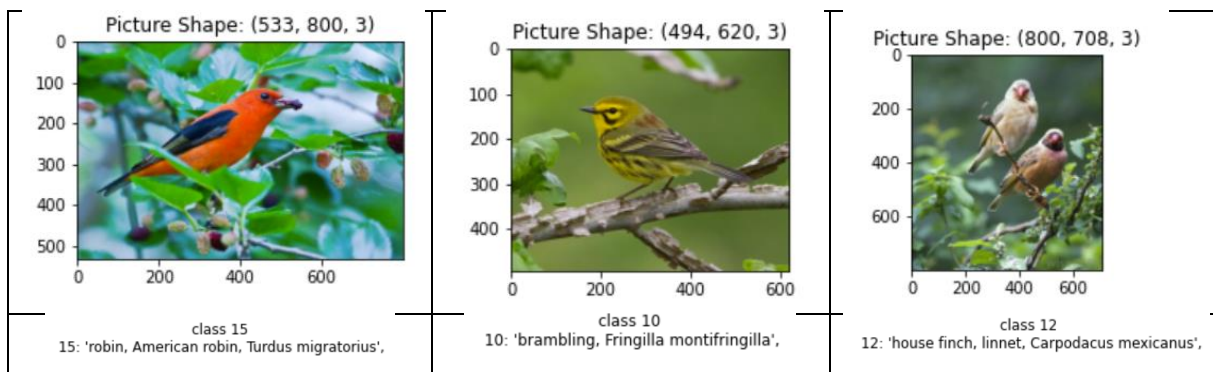
All in all, the preprocessing methods we tried did not improve on the deep segmentation.

### 1.6 Question 6

We were asked to load a pre-trained classifier. We chose *resnet152* from torchvision.

### 1.7 Question 7

We were to pick an animal in its natural habitat and display the image and the classifier's result. We decided to check 3 birds:



### 1.8 Question 8

Segmenting the animals using the deep method:

Image into Segmentor



['bird']



Image into Segmentor



['bird']



Image into Segmentor



['bird']



## 1.9 Question 9

Putting the animals on a different background-habitat





### 1.10 Question 10

Feed-forwarding the new images from question 9 into the pre-trained network and check the results. We show below the three top results for each image.

 <ol style="list-style-type: none"> <li>1. Macaw</li> <li>2. Dishwasher</li> <li>3. Plate rack</li> </ol>	 <ol style="list-style-type: none"> <li>1. Dishwasher</li> <li>2. Desk</li> <li>3. Microwave</li> </ol>	 <ol style="list-style-type: none"> <li>1. Dishwasher</li> <li>2. Partridge</li> <li>3. Plate rack</li> </ol>
 <ol style="list-style-type: none"> <li>1. Lakeside</li> <li>2. Boathouse</li> <li>3. Canoe</li> </ol>	 <ol style="list-style-type: none"> <li>1. Lakeside</li> <li>2. Boathouse</li> <li>3. Canoe</li> </ol>	 <ol style="list-style-type: none"> <li>1. Lakeside</li> <li>2. Boathouse</li> <li>3. Sea-lion</li> </ol>
 <ol style="list-style-type: none"> <li>1. Sandbar</li> <li>2. sarong</li> <li>3. coral reef</li> </ol>	 <ol style="list-style-type: none"> <li>1. Sandbar</li> <li>2. conch</li> <li>3. seashore</li> </ol>	 <ol style="list-style-type: none"> <li>1. Sandbar</li> <li>2. conch</li> <li>3. sunscreen/sunblock</li> </ol>
 <ol style="list-style-type: none"> <li>1. macaw</li> <li>2. indigo bunting</li> <li>3. African gray</li> </ol>	 <ol style="list-style-type: none"> <li>1. Sliding door</li> <li>2. Studio couch</li> <li>3. Home Theater</li> </ol>	 <ol style="list-style-type: none"> <li>1. Partridge</li> <li>2. House finch</li> <li>3. African Gray</li> </ol>

The classifier seems to have identified the surroundings more than the birds themselves even though in most pictures they are center stage. In some cases, even though an animal/object was detected, it was the wrong one! (see highlights).

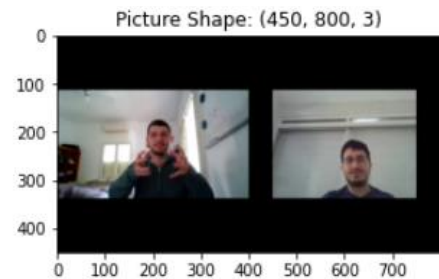
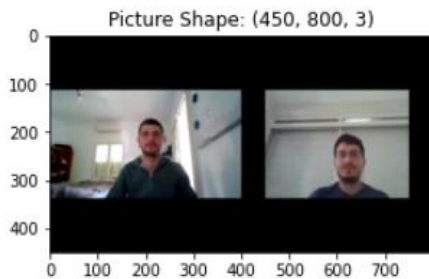
The answers are most definitely different from the ones in Q7.

## 2 Part 2 – ‘Jurassic Fishbach’

### 2.1 Question 1

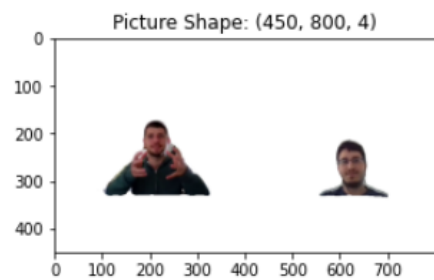
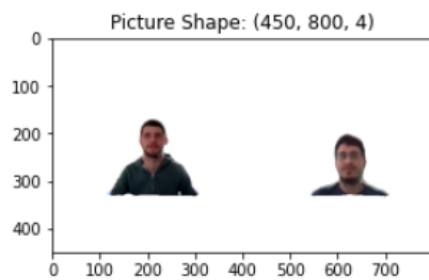
We were to film a short video of our selves and display two frames.

We used ZOOM to have us both in the film.



### 2.2 Question 2

We were to segment ourselves from the video and display two frames. We choose to use *deeplabv3* for this task.



### 2.3 Question 3

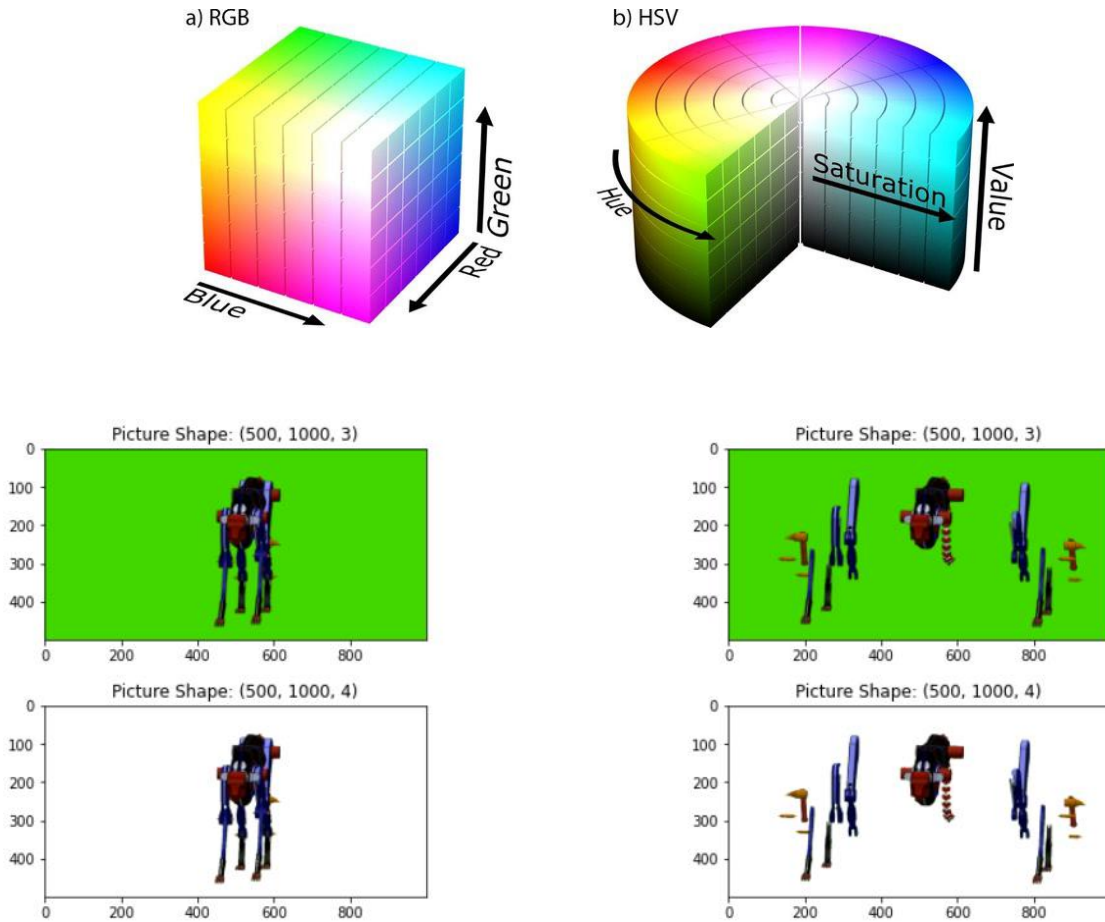
We were to take a green screen video, segment it and display two frames of it.

We made our own green-screen video using a model from *GrabCad* and solidworks

Seeing how the *GrabCad* model is rather dark and has no green parts to it, we decided to use thresholding for the segmentation task. We found that the HSV color space is a-lot easier to segment upon as colors and brightness are on perpendicular axes.

The mask was created with the following code:

```
1. HSV_Mask=(HSV_image[:, :, 0]>40) & (HSV_image[:, :, 0]<90) & (HSV_image[:, :, 2]>150)
```



## 2.4 Question 4

Putting it all together, we were to paste our segmented selves and the segmented green-screen video.

In the video you can see two djinnis arise from the sea to manipulate a dog robot on the beach.



You can view the whole movie on YOUTUBE:

<https://www.youtube.com/watch?v=fY-u-Ap9NBM&feature=youtu.be>