

COMP700 Conclusion

Author: Alexander Goudemond, Student Number: 219030365

In this notebook, the author will summarise the work done so far, as well as display some valuable insights to the reader. The hope is that this assists the reader with understanding how the project progressed, as well as assist the marker by condensing all the knowledge into 1 place, for ease of reference

This notebook has several sections.

Project journey

This project started with the intention to conduct experiments into cell tracking on 10 datasets, provided by the Cell Tracking Challenge (CTC). However, the author ran into so many challenges, that they were not able to explore cell tracking, in the end.

Instead, the author focussed on the different techniques one could conduct to prepare the 10 datasets for cell tracking, by investigating both Traditional Segmentation techniques, and Neural Network Segmentation techniques

At the close of the project, the author had enough information to condense and summarise the problems, methods and insights for this goal.

This notebook hopes to explain in greater detail what the different experiments conducted yielded, and how the knowledge assisted the project.

Previous Notebook Summary

This project generated many Jupyter Notebooks, in order to easily work through the 10 datasets. The notebooks saw many revisions, and in the end, 20 presentable notebooks were formulated and used in this project.

A table summarising this information is shown here:

Notebook Number	Programming Environment	Notebook Size (MB)	Notebook Summary
001	VS Code	0.004	<i>Welcome and Setup</i> - Explains how to recreate environment to run all notebooks, and dependencies
002	VS Code	5.94	<i>Dataset Exploration</i> - Investigates how to represent images together, colourmaps, and hidden information
003	VS Code	6.67	<i>Dataset Colour Exploration</i> - colourmap comparison and side-by-side video comparison
004	VS Code	19	<i>Dataset Initial Pre-Processing</i> - preprocessing techniques

005	VS Code	5.59	<i>Dataset Secondary Pre-Processing</i> - preprocessing techniques
006	VS Code	1.13	<i>Processing Images</i> - bulk morphological processing
007	VS Code	16.9	<i>Initial Segmentation</i> - traditional segmentation techniques
008	VS Code	12	<i>Secondary Segmentation</i> - traditional segmentation technique
009	VS Code	0.0666	<i>Bulk Segmentation</i> - apply bulk thresholding with OpenCV and generate side-by-side video comparison
010	VS Code	1.11	<i>Isolate Training Data</i> - this notebook organises images and masks for segmentation
011	Google Colab	5.1	<i>Neural Network Benchmark Preparation</i> - prepares training data for raw dataset models, as a benchmark
012	Google Colab	17.9	<i>Neural Network Benchmarks</i> - trains 18 models on Raw datasets
013	Google Colab	27.6	<i>Neural Network Benchmark Reflection</i> - reflect on model success and compare information
014	VS Code	0.0111	<i>Dataset Tertiary Pre-Processing</i> - reviews that processed datasets are best option for Neural Network Segmentation
015	Google Colab	0.348	<i>Neural Network Processed Preparation</i> - prepares training data for processed dataset models
016	Google Colab	3.3	<i>Neural Network Processed Images</i> - trains 6 models on processed datasets
017	Google Colab	8	<i>Neural Network Processed Reflection</i> - reflect on model success and compare information
018	Google Colab	10.8	<i>Neural Network Predictions altogether</i> - load and predict models side-by-side, as well as mix-and-match models and datasets
019	Google Colab	8.4	<i>Neural Network Image Mean IoU Scores</i> - load models and datasets and conduct image Mean IoU Score
020	VS Code	0.060	<i>Conclusion</i> - condenses all prior work and insights into 1 notebook
TOTAL (MB)		141.93	

Dataset Visualization


All 10 Datasets used:

 Visualization of all 10 datasets

Dataset processing attempts

As one can see, the images below did not assist with the processing of the images...


<!--

 alt text


--> *Histogram Equalized Images:*

 Histogram Equalized Images


<!--

 alt text

--> *127 Thresholded Images:*

 127 Thresholded Images


<!--

 alt text

--> *OpenCV '17' Thresholded Images:*

 OpenCV '17' Thresholded Images


<!--

 alt text

--> *Otsu Thresholding:*

 Otsu Thresholding


<!--

 alt text

--> *Blurred Adaptive Thresholded Mean:*

 Blurred Adaptive Thresholded Mean

<!--


 alt text

--> *Opened At Start Blurred Adaptive Threshold Mean:*

 Opened At Start Blurred Adaptive Threshold Mean

These images were the best results yet, and were used in the processed image datasets!


<!--

 alt text

--> *Brightened, Dilated then Opened:*

 Brightened, Dilated then Opened

<!--

 alt text

--> *Brightened, Opened then Eroded:*

 Brightened, Opened then Eroded

Dataset traditional segmentation attempts

Unfortunately, traditional watershed contained poor results, because the original images were not distinct cells


<!--

 alt text

--> *Default Watershed:*

 Default Watershed

<!--

 alt text

--> *Watershed with OpenCV Threshold 17:*

 Watershed with OpenCV Threshold 17

Other segmentation attempts also failed:


<!--

 alt text

--> *Canny Edge Detection:*

 Canny Edge Detection

<!--

 alt text

--> *Simple Contours:*

 Simple Contours

<!--

 alt text

--> *Complete Contours:*

 Complete Contours


<!--

 alt text

--> *5 Means Clustering:*

 5 Means Clustering

<!--

 alt text

--> *Region Filled:*

 Region Filled

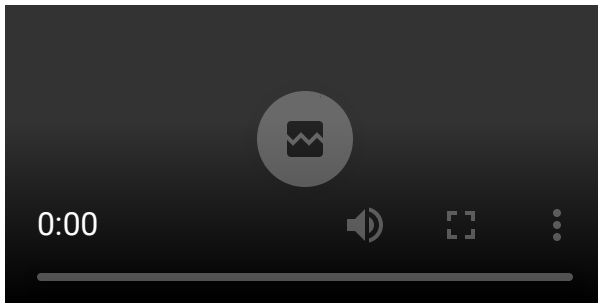
Useful Videos showing Colourmaps and their affect on the datasets

The following contain information that assisted the author with visualising the information present over time. More examples are available in the relevant folder!

```
In [7]: from IPython.display import Video
```

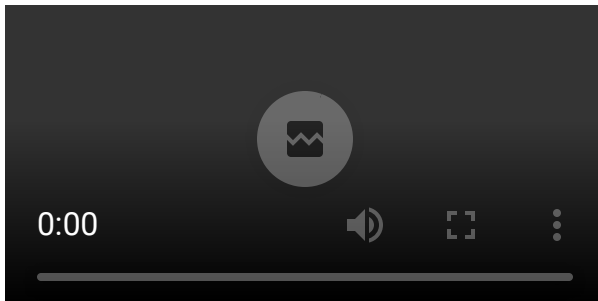
```
In [6]: Video("003_ColourExploration//BF-C2DL-HSC_01.mp4")
```

Out[6]:



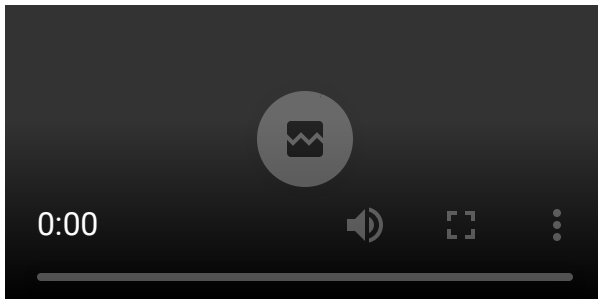
```
In [8]: Video("003_ColourExploration//DIC-C2DH-HeLa_01.mp4")
```

Out[8]:



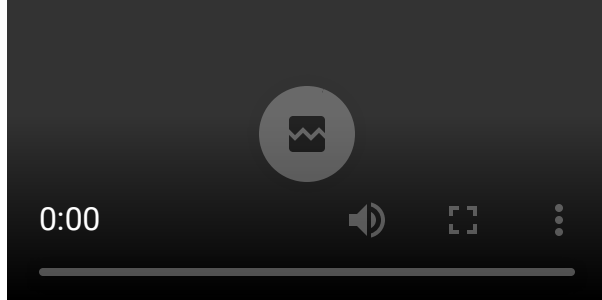
```
In [9]: Video("003_ColourExploration//Fluo-C2DL-Huh7_01.mp4")
```

Out[9]:



```
In [10]: Video("003_ColourExploration//Fluo-C2DL-MSC_01.mp4")
```

Out[10]:



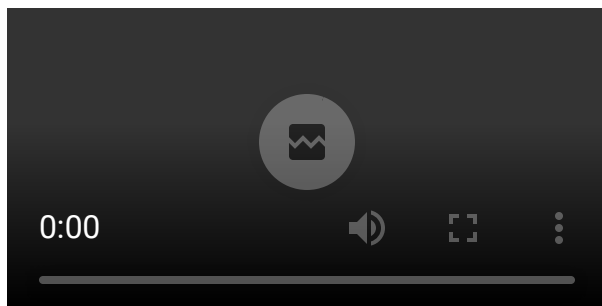
Useful Videos showing traditional segmentation techniques alongisde segmentation masks and tracking masks

The following contain information that assisted the author with visualising the information present over time. More examples are available in the relevant folder!

```
In [1]: from IPython.display import Video
```

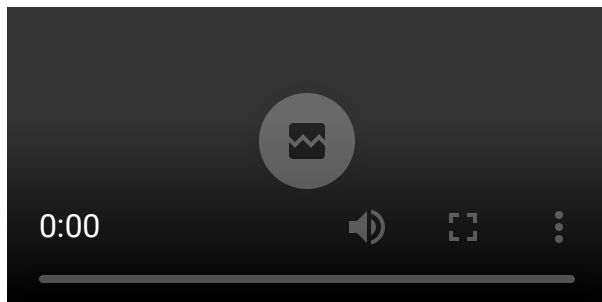
```
In [2]: Video("009_Segmentation_Videos//BF-C2DL-HSC.mp4")
```

Out[2]:



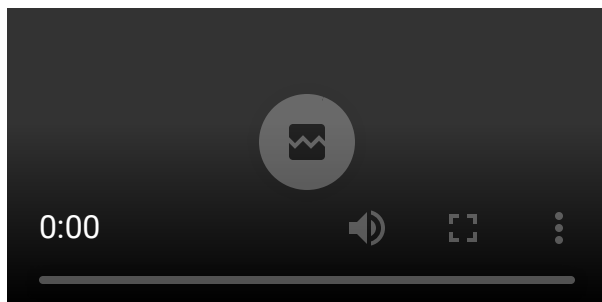
```
In [3]: Video("009_Segmentation_Videos//Fluo-N2DH-GOWT1.mp4")
```

Out[3]:



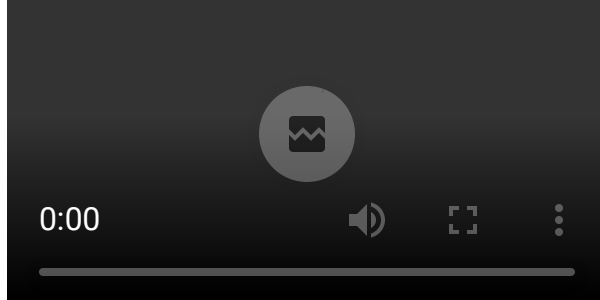
```
In [4]: Video("009_Segmentation_Videos//PhC-C2DH-U373.mp4")
```

Out[4]:



```
In [5]: Video("009_Segmentation_Videos//PhC-C2DL-PSC.mp4")
```

Out[5]:



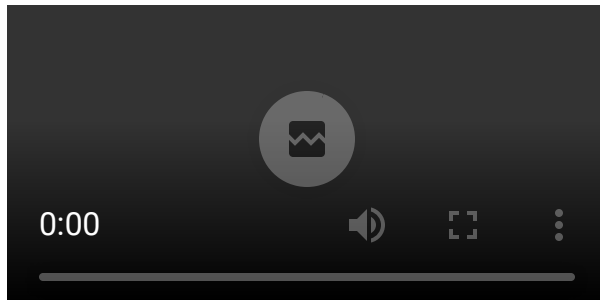
Training Data and Their Videos

The following contain information that assisted the author with visualising the information present over time. More examples are available in the relevant folder!

In [11]: `from IPython.display import Video`

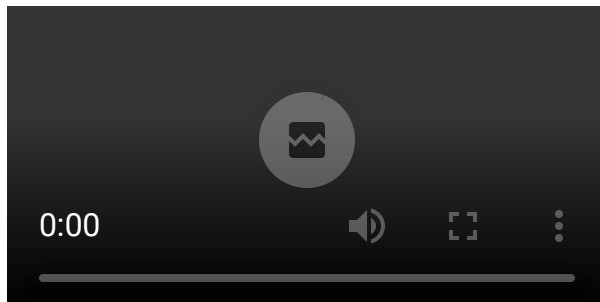
In [12]: `Video("010_Training_Data_Videos//GT//BF-C2DL-HSC_02.mp4")`

Out[12]:



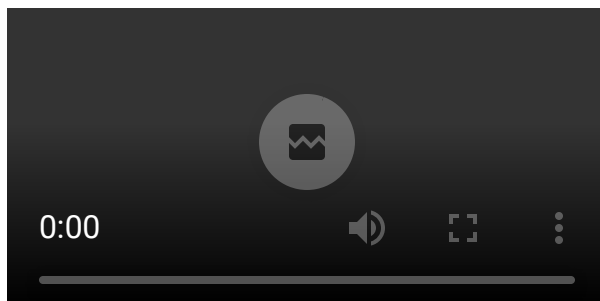
In [14]: `Video("010_Training_Data_Videos//GT//Fluo-N2DH-GOWT1_01.mp4")`

Out[14]:



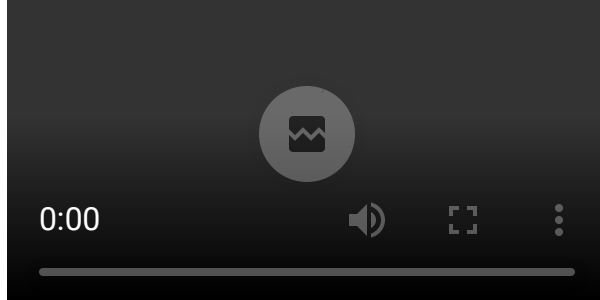
In [15]: `Video("010_Training_Data_Videos//ST//DIC-C2DH-HeLa_02.mp4")`

Out[15]:



In [16]: `Video("010_Training_Data_Videos//ST//PhC-C2DL-PSC_02.mp4")`

Out[16]:



UNet Implementation and Methodology

Below is a collection of useful information to explain why this project completed tasks in the way it did. What is included in this cell is a description to explain it in detail

The goal of this project was to compare segmentation techniques of 10 datasets from the CTC website. When the author moved onto Neural Network Segmentation, they encountered many hurdles to overcome:

- Convolutional Neural Networks (CNNs) seem to require images to be in float pixel range, NOT integer pixel range. Forcing the images to be in integer range results in the model you are training receiving a loss function with NaN values (*not a number* in python)
- The masks that were provided were multiclass masks - meaning that every individual cell was assigned a unique colour. This resulted in the model attempting to classify anywhere from 2 to 256 possible classes - which hurt the training significantly. The masks needed to be converted into binary masks before being used
- Early processing of the images saved the masks under a JPG extension - leading to the compression method from blurring the boundaries between mask values. this resulted in a mask with 12 classes being transformed into a class with upwards of 150... The author had to return and ensure the training images were saved as PNG, if conversion was necessary at all
- It is not possible to simply feed the entire collection of images into the UNet model, as each model has 1 of 12 possible dimensions! (Refer to the table below) This means that the author needed to decide on whether to resize the images, or crop them...
 - It was discovered that resizing the images affected the accuracy of the training model, and lost valuable data. For square images, it also introduced distortion
 - It was discovered that cropping the images would involve the largest image (1036, 1070, 3) would need to be reduced to the smallest image (512, 512, 3) - leading to under half of the original image being lost!
 - To address these challenges, cutting each image into smaller pieces of a consistent size if a good compromise. This is exactly what Patchify does - it produces square 'cake slices' or 'patches' or 'jigsaw pieces' out of a larger image. These patches then have a consistent size, and may be fed into the model!
- Once the patches of the image have been generated, the model also needs corresponding patches of masks! So the author needed to ensure the masks were patchified alongside the images, before training

- Ensuring the image paths were sorted before reading and patchifying the images was essential, as Google seems to randomly save images, for some reason (perhaps under a Hash Map?) The author struggled to identify this issue as in windows the filenames are sorted by default!
- Once the model has been trained, the model can then predict and compare the patchified predictions against the patchified masks. However, this is just 1 way of calculating the Mean IoU score! Another technique is to calculate the score on the entire image, which is what was achieved in notebook 019
- This technique is computationally involved, and the user needs to be careful when using usage restrictions in Google Colab, but allows any image of any dimension to be predicted in a Unet model!

Recall that we have the following information on dataset dimenions:

Dataset Name | Folder Dimensions | |-----|-----| BF-C2DL-HSC | (1010,1010,3) | BF-C2DL-MuSC | (1036,1070,3) | DIC-C2DH-HeLa | (512,512,3) | Fluo-C2DL-Huh7 | (1024,1024,3) | Fluo-C2DL-MSC/01 | 832,992,3) | Fluo-C2DL-MSC/02 | 782,1200,3) | Fluo-N2DH-SIM+/01 | (690,628,3) | Fluo-N2DH-SIM+/02 | (773,739,3) | Fluo-N2DH-GOWT1 | (1024,1024,3) | Fluo-N2DL-HeLa | (700,1100,3) | PhC-C2DH-U373 | (520,696,3) | PhC-C2DL-PSC | (576,720,3) |

As well as quantity of images:

Dataset Name	GT Mask Quantity	ST Mask Quantity
BF-C2DL-HSC/01	49	1764
BF-C2DL-HSC/02	8	1764
BF-C2DL-MuSC/01	50	1376
BF-C2DL-MuSC/02	50	1376
DIC-C2DH-HeLa/01	9	84
DIC-C2DH-HeLa/02	9	84
Fluo-C2DL-Huh7/01	8	N/A
Fluo-C2DL-Huh7/02	5	N/A
Fluo-C2DL-MSC/01	18	48
Fluo-C2DL-MSC/02	33	48
Fluo-N2DH-SIM+/01	30	N/A
Fluo-N2DH-SIM+/02	20	N/A
Fluo-N2DH-GOWT1/01	65	92
Fluo-N2DH-GOWT1/02	150	92
Fluo-N2DL-HeLa/01	28	92
Fluo-N2DL-HeLa/02	8	92
PhC-C2DH-U373/01	15	115
PhC-C2DH-U373/02	19	115
PhC-C2DL-PSC/01	2	300
PhC-C2DL-PSC/02	2	300
Total	578	7742

<!--

 alt text

--> *Project Methodology*:

 Project Methodology.png "Project Methodology")

Load model that was generated in Google Drive and Colab:

```
In [1]: !pip install --upgrade google-api-python-client
```

```
Requirement already satisfied: google-api-python-client in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (2.65.0)
Requirement already satisfied: google-auth<3.0.0dev,>=1.19.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-python-client) (2.14.1)
Requirement already satisfied: uritemplate<5,>=3.0.1 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-python-client) (4.1.1)
Requirement already satisfied: google-auth-http<2.0.0dev,>=0.1.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-python-client) (0.1.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-python-client) (2.10.2)
Requirement already satisfied: http<2.0.0dev,>=0.15.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-python-client) (0.21.0)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (1.56.4)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (3.20.3)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (2.28.1)
Requirement already satisfied: six>=1.9.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-auth<3.0.0dev,>=1.19.0->google-api-python-client) (1.16.0)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-auth<3.0.0dev,>=1.19.0->google-api-python-client) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-auth<3.0.0dev,>=1.19.0->google-api-python-client) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from google-auth<3.0.0dev,>=1.19.0->google-api-python-client) (0.2.8)
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from http<2.0.0dev,>=0.15.0->google-api-python-client) (3.0.9)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3.0.0dev,>=1.19.0->google-api-python-client) (0.4.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (1.26.12)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client) (2022.9.24)
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
```

In [2]: `!pip install protobuf==3.20.*`

```
Requirement already satisfied: protobuf==3.20.* in c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages (3.20.3)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\g5\anaconda3\envs\python_gpu\lib\site-packages)
```

In [5]: `# Code representation of architecture:`
`from keras.models import load_model`
`import google`

`def getModel(path, model_name):`
 `return load_model(path + model_name)`
`###`
`# Google Drive and Google Colab Content\COMP700_UNet_Models\GT_256\gt_256_model_2`
`model = getModel("Google Drive and Google Colab Content//COMP700_UNet_Models//GT_256//",`

`model.summary()`

Model: "UNET_Model_Dimension_256_2"

Layer (type)	Output Shape	Param #	Connected to
=====			
=====			
Input (InputLayer)	[(None, 256, 256, 3 0		[])
)]		
c1_a (Conv2D)	(None, 256, 256, 16 448		['Input[0][0]']
)		
c1_b (Dropout)	(None, 256, 256, 16 0		['c1_a[0][0]']
)		

c1_c (Conv2D)	(None, 256, 256, 16	2320	['c1_b[0][0]']
)		
p1 (MaxPooling2D)	(None, 128, 128, 16	0	['c1_c[0][0]']
)		
c2_a (Conv2D)	(None, 128, 128, 32	4640	['p1[0][0]']
)		
c2_b (Dropout)	(None, 128, 128, 32	0	['c2_a[0][0]']
)		
c2_c (Conv2D)	(None, 128, 128, 32	9248	['c2_b[0][0]']
)		
p2 (MaxPooling2D)	(None, 64, 64, 32)	0	['c2_c[0][0]']
c3_a (Conv2D)	(None, 64, 64, 64)	18496	['p2[0][0]']
c3_b (Dropout)	(None, 64, 64, 64)	0	['c3_a[0][0]']
c3_c (Conv2D)	(None, 64, 64, 64)	36928	['c3_b[0][0]']
p3 (MaxPooling2D)	(None, 32, 32, 64)	0	['c3_c[0][0]']
c4_a (Conv2D)	(None, 32, 32, 128)	73856	['p3[0][0]']
c4_b (Dropout)	(None, 32, 32, 128)	0	['c4_a[0][0]']
c4_c (Conv2D)	(None, 32, 32, 128)	147584	['c4_b[0][0]']
p4 (MaxPooling2D)	(None, 16, 16, 128)	0	['c4_c[0][0]']

c5_a (Conv2D)	(None, 16, 16, 256)	295168	['p4[0][0]']
c5_b (Dropout)	(None, 16, 16, 256)	0	['c5_a[0][0]']
c5_c (Conv2D)	(None, 16, 16, 256)	590080	['c5_b[0][0]']
u6_a (Conv2DTranspose)	(None, 32, 32, 128)	131200	['c5_c[0][0]']
u6_b (Concatenate)	(None, 32, 32, 256)	0	['u6_a[0][0]', 'c4_c[0][0]']
c6_a (Conv2D)	(None, 32, 32, 128)	295040	['u6_b[0][0]']
c6_b (Dropout)	(None, 32, 32, 128)	0	['c6_a[0][0]']
c6_c (Conv2D)	(None, 32, 32, 128)	147584	['c6_b[0][0]']
u7_a (Conv2DTranspose)	(None, 64, 64, 64)	32832	['c6_c[0][0]']
u7_b (Concatenate)	(None, 64, 64, 128)	0	['u7_a[0][0]', 'c3_c[0][0]']
c7_a (Conv2D)	(None, 64, 64, 64)	73792	['u7_b[0][0]']
c7_b (Dropout)	(None, 64, 64, 64)	0	['c7_a[0][0]']
c7_c (Conv2D)	(None, 64, 64, 64)	36928	['c7_b[0][0]']
u8_a (Conv2DTranspose)	(None, 128, 128, 32)	8224	['c7_c[0][0]']
)		
u8_b (Concatenate)	(None, 128, 128, 64)	0	['u8_a[0][0]', 'c2_c[0][0]']
)		

c8_a (Conv2D)	(None, 128, 128, 32 18464	['u8_b[0][0]']
)	
c8_b (Dropout)	(None, 128, 128, 32 0	['c8_a[0][0]']
)	
c8_c (Conv2D)	(None, 128, 128, 32 9248	['c8_b[0][0]']
)	
u9_a (Conv2DTranspose)	(None, 256, 256, 16 2064	['c8_c[0][0]']
)	
u9_b (Concatenate)	(None, 256, 256, 32 0	['u9_a[0][0]',
)	'c1_c[0][0]']
c9_a (Conv2D)	(None, 256, 256, 16 4624	['u9_b[0][0]']
)	
c9_b (Dropout)	(None, 256, 256, 16 0	['c9_a[0][0]']
)	
c9_c (Conv2D)	(None, 256, 256, 16 2320	['c9_b[0][0]']
)	
Output (Conv2D)	(None, 256, 256, 3) 51	['c9_c[0][0]']

```

=====
=====
Total params: 1,941,139
Trainable params: 1,941,139
Non-trainable params: 0

```

Full output here:

Model: "UNET_Model_Dimension_256_2"

Layer (type)	Output Shape	Param #	Connected to
=====			
Input (InputLayer)	[(None, 256, 256, 3)]	0	[]
c1_a (Conv2D)	(None, 256, 256, 16)	448	['Input[0] [0]']
c1_b (Dropout)	(None, 256, 256, 16)	0	['c1_a[0][0]']
c1_c (Conv2D)	(None, 256, 256, 16)	2320	['c1_b[0][0]']
p1 (MaxPooling2D)	(None, 128, 128, 16)	0	['c1_c[0][0]']
c2_a (Conv2D)	(None, 128, 128, 32)	4640	['p1[0][0]']
c2_b (Dropout)	(None, 128, 128, 32)	0	['c2_a[0][0]']
c2_c (Conv2D)	(None, 128, 128, 32)	9248	['c2_b[0][0]']
p2 (MaxPooling2D)	(None, 64, 64, 32)	0	['c2_c[0][0]']
c3_a (Conv2D)	(None, 64, 64, 64)	18496	['p2[0][0]']
c3_b (Dropout)	(None, 64, 64, 64)	0	['c3_a[0][0]']
c3_c (Conv2D)	(None, 64, 64, 64)	36928	['c3_b[0][0]']
p3 (MaxPooling2D)	(None, 32, 32, 64)	0	['c3_c[0][0]']
c4_a (Conv2D)	(None, 32, 32, 128)	73856	['p3[0][0]']
c4_b (Dropout)	(None, 32, 32, 128)	0	['c4_a[0][0]']
c4_c (Conv2D)	(None, 32, 32, 128)	147584	['c4_b[0][0]']
p4 (MaxPooling2D)	(None, 16, 16, 128)	0	['c4_c[0][0]']
c5_a (Conv2D)	(None, 16, 16, 256)	295168	['p4[0][0]']
c5_b (Dropout)	(None, 16, 16, 256)	0	['c5_a[0][0]']
c5_c (Conv2D)	(None, 16, 16, 256)	590080	['c5_b[0][0]']
u6_a (Conv2DTranspose)	(None, 32, 32, 128)	131200	['c5_c[0][0]']
u6_b (Concatenate)	(None, 32, 32, 256)	0	['u6_a[0][0]',

			'c4_c[0][0]'
c6_a (Conv2D)	(None, 32, 32, 128)	295040	['u6_b[0][0]']
c6_b (Dropout)	(None, 32, 32, 128)	0	['c6_a[0][0]']
c6_c (Conv2D)	(None, 32, 32, 128)	147584	['c6_b[0][0]']
u7_a (Conv2DTranspose)	(None, 64, 64, 64)	32832	['c6_c[0][0]']
u7_b (Concatenate)	(None, 64, 64, 128)	0	['u7_a[0][0]', 'c3_c[0][0]']
c7_a (Conv2D)	(None, 64, 64, 64)	73792	['u7_b[0][0]']
c7_b (Dropout)	(None, 64, 64, 64)	0	['c7_a[0][0]']
c7_c (Conv2D)	(None, 64, 64, 64)	36928	['c7_b[0][0]']
u8_a (Conv2DTranspose)	(None, 128, 128, 32)	8224	['c7_c[0][0]']
u8_b (Concatenate)	(None, 128, 128, 64)	0	['u8_a[0][0]', 'c2_c[0][0]']
c8_a (Conv2D)	(None, 128, 128, 32)	18464	['u8_b[0][0]']
c8_b (Dropout)	(None, 128, 128, 32)	0	['c8_a[0][0]']
c8_c (Conv2D)	(None, 128, 128, 32)	9248	['c8_b[0][0]']
u9_a (Conv2DTranspose)	(None, 256, 256, 16)	2064	['c8_c[0][0]']
u9_b (Concatenate)	(None, 256, 256, 32)	0	['u9_a[0][0]', 'c1_c[0][0]']
c9_a (Conv2D)	(None, 256, 256, 16)	4624	['u9_b[0][0]']
c9_b (Dropout)	(None, 256, 256, 16)	0	['c9_a[0][0]']
c9_c (Conv2D)	(None, 256, 256, 16)	2320	['c9_b[0][0]']
Output (Conv2D)	(None, 256, 256, 3)	51	['c9_c[0][0]']

=====

Total params: 1,941,139
Trainable params: 1,941,139
Non-trainable params: 0

<!--

 alt text

--> *Unet Architecture:*

 Unet Architecture