RESEARCH PAPER – NUCLEI DETECTION

Team member: Srujani Elango

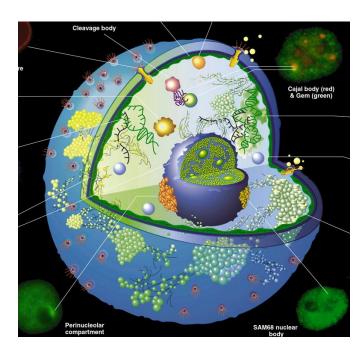
Topic: Finding the nuclei in divergent images to advance medical discovery

Abstract

The idea of this project is to spot nuclei to speed up curing process for every disease. The aim is to create an algorithm to automate nucleus detection so that researchers can identify each individual cell in a sample, and by measuring how cells react to various treatments, the researcher can understand the underlying biological processes at work. I tackled this problem by using the following segmentation models — U-Net, Mask RCNN and Resnet 101. For the first steps of understanding the problem I needed to get a domain knowledge of what cells and nuclei are, understanding what segmentation is and what are the various models used for it. I chose these three models to understand and compare how the three varied from each other and in my experiments, I found U-Net to have the best accuracy of 83% and it could segment the nuclei properly. Mask RCNN was the second best performing model and Resnet which is pretrained on the coco dataset was very resource exhaustive and generated only a few test images as there were 20000 training steps.

Steps in medical diagnosis

- 1. Locate cells in varied conditions by spotting nuclei
- 2. Develop drugs
- 3. Improve health and quality of life



Nuclei

Data

Data source: https://www.kaggle.com/c/data-science-bowl-2018/data

The dataset contains many segmented nuclei images. Images were acquired under a variety of conditions and vary in the cell type, magnification and imaging modality. The dataset is split into train and test images. There are 670 train samples and around 4000 test samples. This dataset split shows there are more test samples than train ones.

Exploratory Data Analysis

Displayed below are the image id's in stage 1 train folder. The train image id's with the label's encoded pixels are displayed in a tabular format. Nuclei are depicted as encoded pixel values. Each image in the train set has 40 masks on an average which brings the total count of train image size of around 30,000.

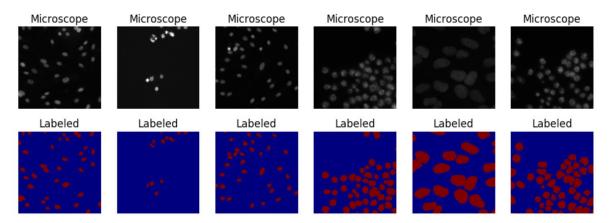
	lmageld	EncodedPixels
18480	a022908f1b7880838dbc0411e50828e64b4f5e0263afdf	[52841, 7, 53094, 11, 53349, 13, 53604, 14, 53
19560	a65bbfc5673e8053b6ce49f39c79cf3a846fe5cc46dd93	[63093, 9, 63609, 16, 64127, 20, 64646, 23, 65
15212	7b5987a24dd57325e82812371b3f4df7edc528e0526754	[27915, 6, 28169, 9, 28424, 12, 28679, 14, 289

The images are accessed by splitting the path using slash \ for train and test.

	path	lmageld	ImageType	TrainingSplit	Stage
22502	D:/Kaggle/Nuclei/input\stage1_train\b98681c748	b98681c74842c4058bd2f88b06063731c26a90da083b1e	masks	train	stage1
27649	D:/Kaggle/Nuclei/input\stage1_train\e7a3a7c994	e7a3a7c99483c243742b6cfa74e81cd48f126dcef00401	masks	train	stage1

The images from microscope and the labeled masks are shown in the following figure:

Labeled Masks



The following are some statistics describing mask counts, minimum nuclei size, maximum nuclei size, mean nuclei size and standard deviation of nuclei size

a		•	4	. 1	
Sum	ma	r17 0	t a	h	Α

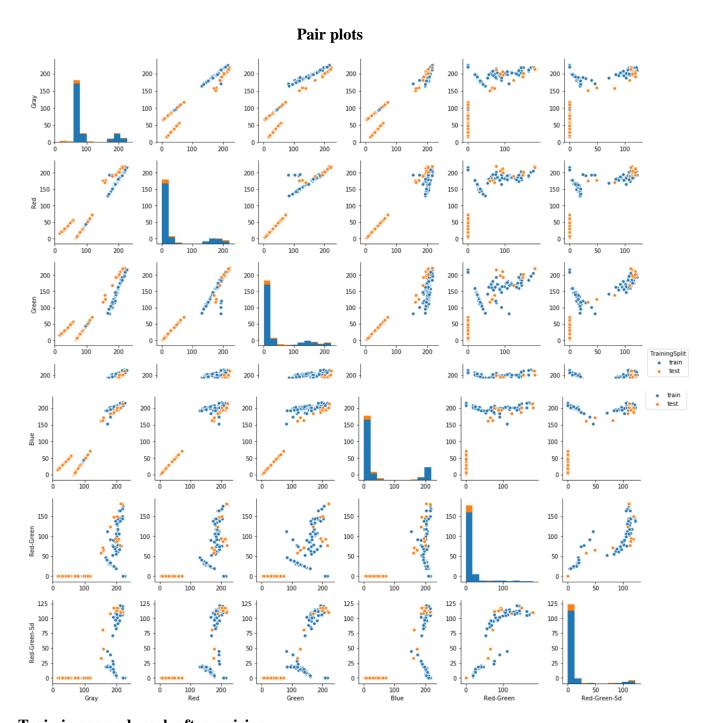
	mask_counts	nuclei_size_min	nuclei_size_max	nuclei_size_mean	nuclei_size_std
count	670.000000	670.000000	670.000000	670.000000	670.000000
mean	43.971642	69.270149	1243.029851	560.483462	306.627435
std	47.962530	197.402043	1346.599675	634.752932	359.936976
min	1.000000	21.000000	26.000000	26.000000	0.000000
25%	15.250000	24.000000	287.250000	150.521825	65.940617
50%	27.000000	33.000000	634.500000	265.434343	148.111038
75%	54.000000	63.000000	1733.000000	674.434370	394.974610
max	375.000000	4367.000000	11037.000000	7244.071429	1965.426189

The following is the value counts of images in train set. This analysis shows that 50% of the images have 256, 256 size and rest 50% have different sizes. Images must be resized for further steps.

Value counts

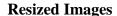
```
[11]: train_img_df['images'].map(lambda x: x.shape).value_counts()
:[11]: (256, 256, 3)
                        334
       (256, 320, 3)
                        112
                       92
      (520, 696, 3)
      (360, 360, 3)
                         91
      (1024, 1024, 3)
                       16
      (512, 640, 3)
                         13
      (603, 1272, 3)
                          6
      (260, 347, 3)
      (1040, 1388, 3)
      Name: images, dtype: int64
```

I used pair plots to observe the intensity of train and test images and also to identify different color groups in the dataset. This shows the color intensity for train and test are different and images are predominantly red, green, gray and blue. The below graph shows the distribution of the train and test images.



Train image and mask after resizing

The last step of data preprocessing is image resizing. I used skimage's resize to resize train and test images and masks.



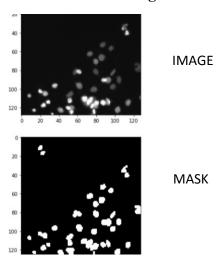


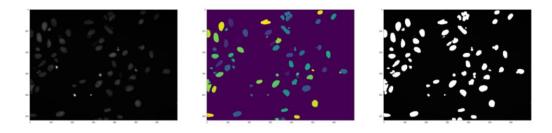
Image Segmentation

The technology of image segmentation is widely used in medical image processing, face recognition pedestrian detection, etc. The current image segmentation techniques include region-based segmentation, edge detection segmentation, segmentation based on clustering, segmentation based on weakly-supervised learning in CNN, etc.

Segmentation has two objectives. The first objective is to decompose the image into parts for further analysis. In simple cases, the environment might be well enough controlled so that the segmentation process reliably extracts only the parts that need to be analyzed further. For example, in the chapter on color, an algorithm was presented for segmenting a human face from a color video image. The segmentation is reliable, provided that the person's clothing or room background does not have the same color components as a human face.

The second objective of segmentation is to perform a change of representation. The pixels of the image must be organized into higher-level units that are either more meaningful or more efficient for further analysis (or both). A critical issue is whether or not segmentation can be performed for many different domains using general bottom-up methods that do not use any special domain knowledge.

Sample Image Segmentation



The sample images with the masks are displayed above.

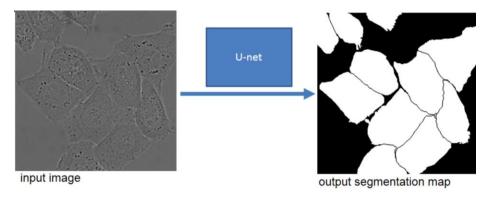
Suggested models

Re-implementation of Kaggle's best models

1. U-Net: CNN - for faster image segmentation

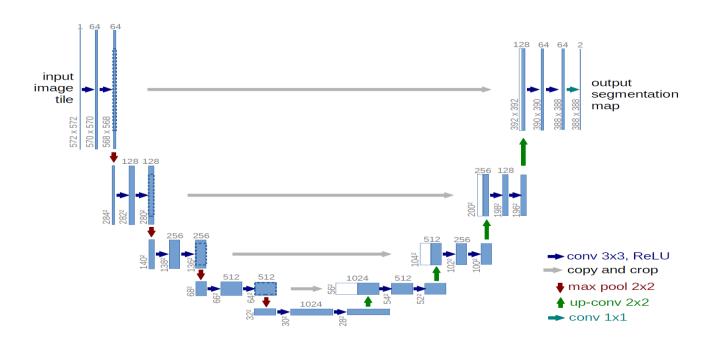
U-Net implementation is proposed by Ronneberger et al. developed along with tensorflow. The code has been developed and used for Radio Frequency Interference mitigation using deep convolutional neural networks. The network can be trained to perform image segmentation on arbitrary imaging data. The model starts with blocks of convolution layers and max pooling layers followed by deconvolution layers and recreating the image step by step. The output is similar to input image size with segmented or labeled masks.

Input image and Output image in U-Net



The input image is fed to the U-Net model and the sample output segmentation map is shown.

U-NET ARCHITECTURE



U-Net: Convolution layers

These layers convolve through the image and get a detail observation of the image. The model can be made more detail by increasing the convolution layers. I used 12 layers of convolution and few maxpooling layers to consider maximum values and to reduce architecture size.

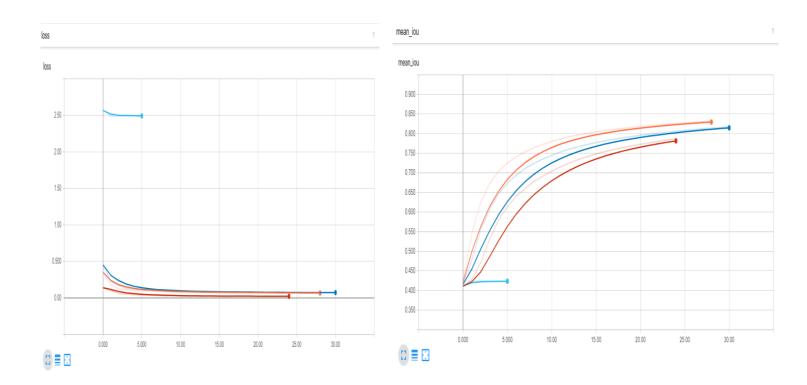
U-Net: Deconvolution layers

These layers are generated by using conv transpose function in tensorflow which performs the reverse of convolution. The output is generated by gradually recreating the image using convolution layers used previously.

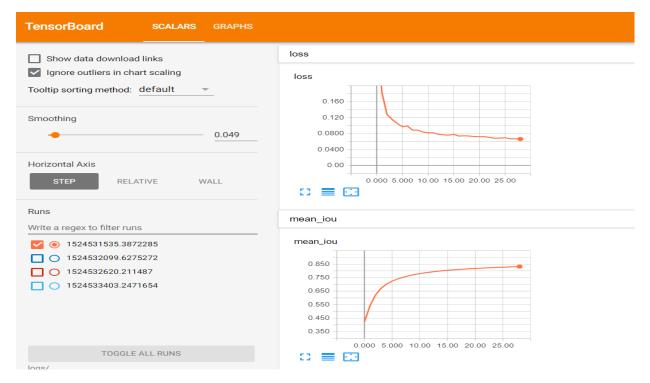
U-Net Tensorboard results

Comparing different hyper parameter changes and visualizing training in real time with the use of tensorboard.

Tensorboard comparison

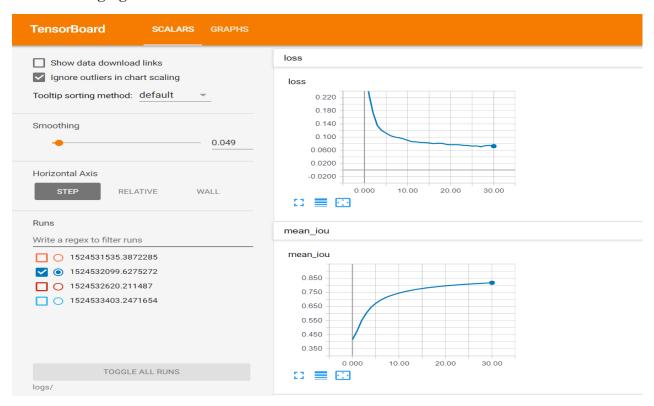


1. Default parameters chosen at initial step



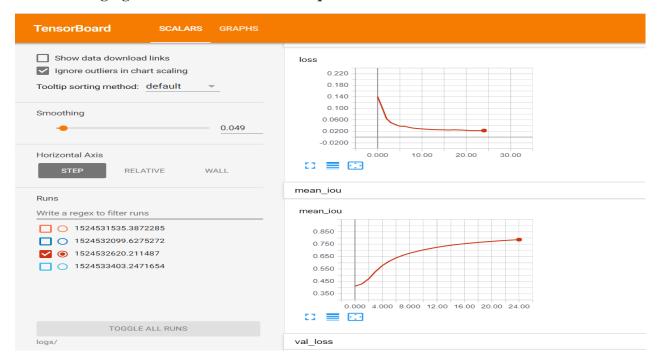
The particular model parameters I had chosen in the first proved to be the best.

2. Changing activation function to SELU



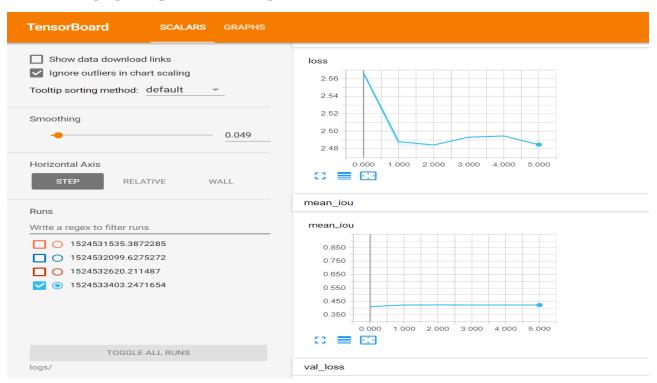
Changing the activation function to SELU which output normalized values didn't help with increasing the accuracy.

3. Changing the loss function to mean squared error



Loss function of mean squared error proved to be less efficient with the RELU activation function.

4. Changing the optimizer to Adagrad



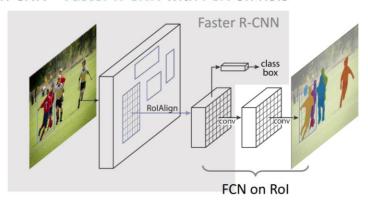
Using Adagrad as the optimizer resulted in early stopping as the loss was stagnant and the optimizer was not suitable for the particular problem in hand.

2. Mask RCNN

Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. When run without modifications on the original Faster R-CNN architecture, the Mask R-CNN authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image. Since image segmentation requires pixel level specificity, unlike bounding boxes, this naturally led to inaccuracies.

Mask R-CNN

Mask R-CNN = Faster R-CNN with FCN on Rols



Faster R-CNN

Faster R-CNN has two networks: region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the image segmentation network.

FCN – Fully convolutional network

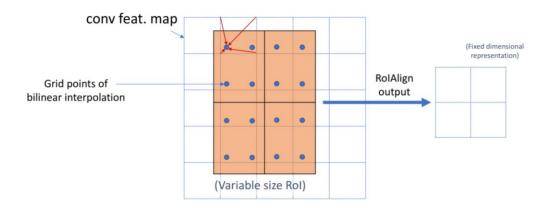
Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Fully convolutional networks take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning.

ROI (Region of interest) Align

RolAlign

FAQs: how to sample grid points within a cell?

- · 4 regular points in 2x2 sub-cells
- · other implementation could work



A region of interest (often abbreviated ROI), are samples within a data set identified for a purpose. The concept of a ROI is commonly used in many application areas. For example, in medical imaging, the boundaries of a tumor may be defined on an image or in a volume, for the purpose of measuring its size

My model

1. Resnet – 101

Deeper neural networks are more difficult to train. A residual learning framework to ease the training of networks that are substantially deeper than those used previously. The layers are reformulated as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. On the ImageNet dataset evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
conv2_x	56×56	3×3, 64 3×3, 64	[3×3, 64]×3	1×1, 64 3×3, 64 1×1, 256	1×1,64 3×3,64 1×1,256	1×1, 64 3×3, 64 1×1, 256
conv3.x	28×28	[3×3, 128]×2	3×3, 128 3×3, 128 ×4	1×1, 128 3×3, 128 1×1, 512 ×4	1×1, 128 3×3, 128 1×1, 512 ×4	1×1, 128 3×3, 128 1×1, 512 x8
conv4_x	14×14	[3×3, 256]×2	[3×3, 256]×6	1×1, 256 3×3, 256 1×1, 1024 ×6	1×1, 256 3×3, 256 1×1, 1024	1×1, 256 3×3, 256 1×1, 1024
conv5_x	7×7	\[\begin{array}{c} 3 \times 3, 512 \\ 3 \times 3, 512 \end{array} \] \times 2	[3×3,512]×3	1×1,512 3×3,512 1×1,2048 ×3	1×1, 512 3×3, 512 1×1, 2048 ×3	1×1, 512 3×3, 512 1×1, 2048
	1×1	1	ave	rage pool, 1000-d fc,	softmax	
FL	OPs	1.8×10^9	3.6×10 ⁹	3.8×10 ⁹	7.6×10 ⁹	11.3×10 ⁹

RESNET ARCHITECTURE

Training steps

Metric - Intersection over union

Intersection over union IoU is a common metric for assessing performance in semantic segmentation tasks. In a sense, IoU is to segmentation what an F1 score is to classification. Both are non-differentiable, and not normally optimized directly. Optimizing cross entropy loss is a common proxy for these scores, that usually leads to decent performance, provided everything else has been setup correctly, e.g regularization, stopping training at an appropriate time.

$$I(X) = \sum_{v \in V} X_v imes Y_v$$
 $U(X) = \sum_{v \in V} X_v + Y_v - X_v imes Y_v$
 $IoU = rac{I(X)}{U(X)}$
 $loss = 1.0 - IoU$

IoU Metric equation

U-Net efficiency

By far, U-Net has the best efficiency validation mIOU value 83%. I used 19 layers of convolution and 4 deconvolution layers to attain this efficiency.

Efficiency screenshot

<pre>outputs = Conv2D(1, (1, 1), act model = Model(inputs=[inputs], model.compile(optimizer='adam', model.summary()</pre>	outputs=[outpu	ts])		metrics=[mean_iou])
conv2d_16 (Conv2D)	(None, 6	4, 64	, 32)	9248	dropout_8[0][0]
conv2d_transpose_4 (Conv2DTrans	(None, 1	28, 1	28, 16) 2064	conv2d_16[0][0]
concatenate_4 (Concatenate)	(None, 1	28, 1	28, 32) 0	conv2d_transpose_4[0][0] conv2d_2[0][0]
conv2d_17 (Conv2D)	(None, 1	28, 1	28, 16) 4624	concatenate_4[0][0]
dropout_9 (Dropout)	(None, 1	28, 1	28, 16) 0	conv2d_17[0][0]
conv2d_18 (Conv2D)	(None, 1	28, 1	28, 16) 2320	dropout_9[0][0]
conv2d_19 (Conv2D)	(None, 1	28, 1	28, 1)	17	conv2d_18[0][0]
Total params: 1,941,105 Trainable params: 1,941,105 Non-trainable params: 0					

Conclusion

The main goal of this problem was to understand various segmentation models and compare their efficiency to find the best suited one for the given problem. The architectures I tried are CNN, FCN, U-Net, Mask RCNN and Resnet-101. My hypothesis that U-Net would perform better when compared to other models was correct. The way I could improve on this problem is to spend time on gathering more training data given the split of training and test images. The images with the masks were generated clearly and the segments were spotted correctly.

References

- [1] http://kaiminghe.com/iccv17tutorial/maskrcnn_iccv2017_tutorial_kaiminghe.pdf
- [2] https://github.com/KaimingHe/deep-residual-networks
- [3] https://github.com/matterport/Mask_RCNN
- [4] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
- [5] https://github.com/shelhamer/fcn.berkeleyvision.org

LICENSE

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

This license is acceptable for Free Cultural Works.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.