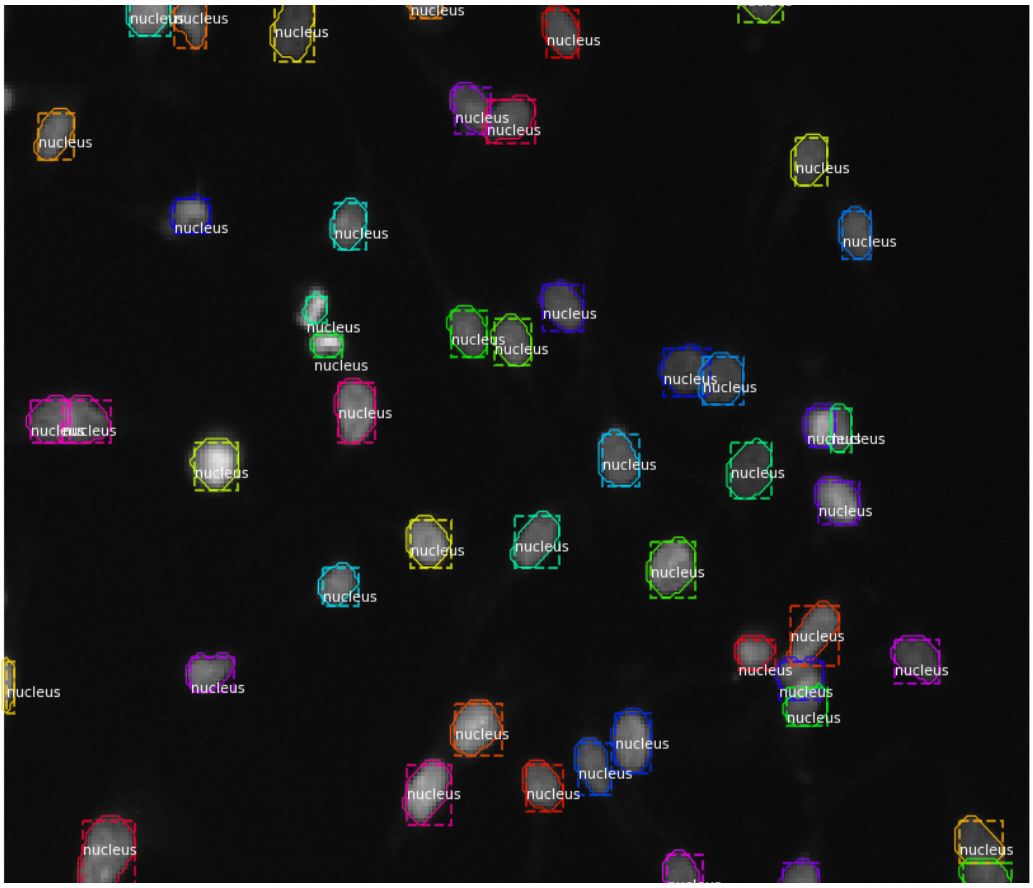
A Vision For Medicine



Dillon Donohue

04.25.2018

Udacity - Machine Learning Nanodegree

Capstone

I. PROJECT DEFINITION

INTRODUCTION

This project aims to study and apply image segmentation to biological cells for the purpose of automating nucleus detection and identification. Medical research, including drug development, pathological research, and tissue engineering, all begin with studying cell cultures and how they respond to chemical and physical stimuli. This project is motivated by the 2018 Data Science Bowl competition held by Kaggle (www.kaggle.com). The resulting computer vision model has been used in the competition. Image segmentation of human cells, and mammalian cells in general, is a core component of biological and medical research. A vast amount of information can be learned from studying cell phenotype and morphology including cell type and health. For this reason, computers have been employed to automate the task of identifying and interpreting cells since the 1960s (Meijering, 2012). The cell nucleus is the focal point for image segmentation in this project because it is often the most easy to identify feature of an individual cell.

I am personally very motivated to contribute to the effort of improving automated cell segmentation. I have spent many hours manually labeling and masking images of cells and can relate to the tedious nature of this work. As an undergraduate studying Chemical and Biological Engineering at the University of Colorado Boulder, I worked in a tissue engineering lab that extensively studies human cells and their interactions with synthetic polymers designed to emulate biological conditions. It is easy to understand that freeing scientists from manually studying images of cells can both increase throughput of images and reduce variation in segmentation. Variation arises from the natural variability of human interpretation. Enhanced throughput of cell images has the power to expedite medical and health advancements. This will allow scientists to focus on the more challenging aspects of research and increase the throughput of research itself.

PROBLEM STATEMENT

The focus of this project is employing computer vision and convolutional neural networks (CNN) for the task of identifying and masking cell nuclei in divergent images. This computer vision model is built with the intention to generalize across multiple input variables including cell type, magnification, and image modality (brightfield and fluorescence) as seen in Figure 2. Existing CNN models built for image segmentation are explored and tested as possible solutions for this project. The model of focus is Facebook AI Research group's Mask R-CNN (He, Gkioxari, Dollar, & Girshick, 2017; Ronneberger, Fischer, & Brox, 2015). The objective is to push the state of the art in cell nuclei segmentation by submitting the most performant model from this project to the 2018 Data Science Bowl (2018 DSB) competition hosted by Kaggle.

Two models are built and used in this project, the solution model and the benchmark model. The solution model is composed of the Mask R-CNN model tailored to the 2018 DSB.

The benchmark model is a simple baseline model by which the solution model can be compared.

METRICS

Conceptually speaking, evaluation of the benchmark model and solution model are performed by comparing the segmented masks produced by the model with a set of ground truth masks held by Kaggle. For this project, Kaggle's evaluation metrics are used to both determine performance of the solution model and compare this model's performance with the Kaggle leaderboard. The evaluation metrics for the 2018 Data Science Bowl are a comparison of predicted masks and ground truth masks using intersection over union (IoU) (2018 Data Science Bowl Evaluation). Intersection over union of a set of predicted pixels in an object versus the true pixels of the object is found using the equation:

$$IoU(A,\ B)\ =\ \frac{A \cap B}{A \cup B}$$

A series of IoU threshold values have been defined over which an average precision will be calculated for all masks in an image for a given threshold. A predicted mask is considered correct if the IoU of the predicted mask versus the ground-truth mask is greater than the threshold value. The threshold values have been specified as 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95. Precision for a given threshold and image is calculated as:

$$P(t)\ =\ \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

In the above equation, TP is the count of true positives, FP is false positives, and FN is false negatives. A mean precision for an image is then calculated by summing the precision for each IoU threshold and dividing by the number of thresholds:

$$\frac{1}{|thresholds|} \sum_{t} \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

Finally, the mean precision for each image is averaged over all images in the test set to produce a mean average precision. The mean average precision (MaP) is the metric used in this project and by Kaggle to assign a score for each model. The scoring metric has been implemented in the Jupyter Notebook file "capstone_test_score.ipynb". All model scores have been evaluated using this notebook.

II. ANALYSIS

DATA EXPLORATION

This project will uses an image dataset that has been provided by Kaggle for the 2018 Data Science Bowl competition (2018 DSB). An investigation of the training data can be found in the Jupyter notebook named "capstone_inspect_data.ipynb". Images in this dataset come in the form of PNG files that range in pixel size from 256 x 256 up to 1040 x 1388 with several sizes in between. Figure 1 shows the image sizes and counts for the training set.

(256, 256, 3)	334
(256, 320, 3)	112
(520, 696, 3)	92
(360, 360, 3)	91
(1024, 1024, 3)	16
(512, 640, 3)	13
(603, 1272, 3)	6
(260, 347, 3)	5
(1040, 1388, 3)	1

Figure 1 - Training image dimensions and counts.

In addition to the variable size of the images, the dataset also contains images generated from different microscopy techniques. These techniques include bright-field microscopy and fluorescence microscopy. In terms of computer vision, the significance of the two techniques being used for capturing images is the differing background and foreground coloring in the images. In this dataset the bright-field images come in the form of histology stained nuclei and unstained nuclei. The stained nuclei appear as pink/purple objects on a white background while the unstained nuclei show as black objects on a white background. In contrast, fluorescent images show nuclei as white objects on a black background. See Figure 2 for representative images of the aforementioned imaging techniques. The images in Figure 2 also demonstrate the disparate image sizes as shown by the image axes. This variability in image size and color composition played a large role in the difficulty of building highly accurate computer vision models for segmentation and masking.

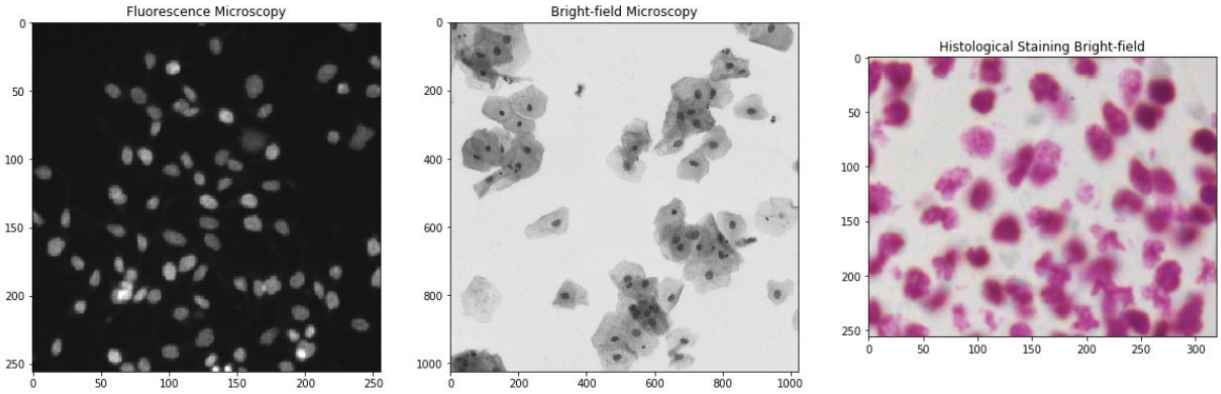


Figure 2 - Images from the Kaggle 2018 DSB dataset. Left image uses fluorescence microscopy, center is a bright-field image, and right is a histological image.

The Kaggle provided dataset has been broken into training and testing sets. The training set contains 670 image samples. Each input image has an associated set of segmented mask images for each nucleus found in the image. For example, a raw image containing 10 nuclei would be accompanied by 10 mask images. One for each segmented nucleus. See Figure 3 for an example of an input image alongside a series of segment mask images for this sample. The segmented mask images serve as labels for each nucleus in the original image. In other words, the mask images serve as the ground-truth images for all nuclei in the training set. Therefore, these masks are used to calculate error between a models predicted mask and the human

annotated mask (supplied mask). This is the error that will be used for optimization and backpropagation through the model to update model weights.

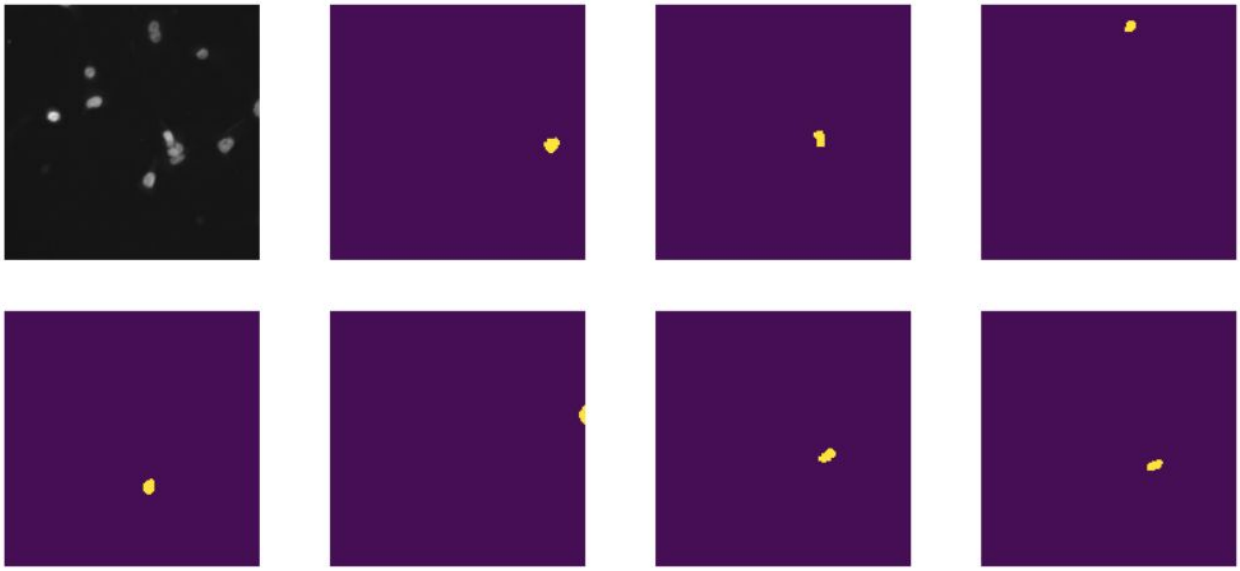


Figure 3 - Top left image is the input image of this training sample. The adjacent 7 images are the mask images for the individual nuclei in the input image. Note, not all masks are shown.

The number of nuclei in each image varies substantially as seen in Figure 4. The average number of nuclei in a training image is about 44, while the most in one image is 375 nuclei and the least is 1. This very broad range of objects to segment presents a challenge when building a robust convolutional neural net.

count	670.000000
mean	43.971642
std	47.962530
min	1.000000
25%	15.250000
50%	27.000000
75%	54.000000
max	375.000000

Figure 4 - Training image mask stats.

The test set in the 2018 DSB dataset holds 65 input images for evaluating model performance. The competition is broken into two stages according to the two testing datasets. The two stage system in Kaggle competitions is used to favor models that generalize well to two test sets as opposed to a model that is overfit to a single test set.

This dataset is ideally suited for this project as it provides high quality training data specifically focused on cell nuclei. The images and masks were hand selected and annotated by scientist from the Broad Institute of MIT and Harvard for the purpose of nucleus segmentation and masking. Additionally, it should be mentioned that the images have been released under the Creative Commons license 0 and therefore are within the public domain (2018 Data Science Bowl).

COMPUTER VISION TECHNIQUES

Object detection, and specifically segmentation of cell nuclei, within the field of computer vision technique is a well studied concept. Despite object detection and image segmentation being the focus of many research groups, this topic in computer vision still has much room for growth and improvement. A relatively recent, and significant, advancement in image segmentation was the integration of convolutional neural networks into computer vision algorithms. CNNs are particularly adept at image classification due to their ability to identify patterns in images. The depth of convolutional neural networks can be increased to then merge patterns into higher level feature detection, and eventually robust classification. In the application of cell nuclei segmentation and masking, image classification alone isn't particularly helpful. The next step is to label any and all objects in an image as nucleus or not nucleus (background). This step is accomplished by employing a regional convolutional neural network (R-CNN). As depicted in the Faster R-CNN model in Figure 5, the input image is first passed through a convolutional neural network to generate a feature map (Ren, He, Girshick, & Sun, 2017).

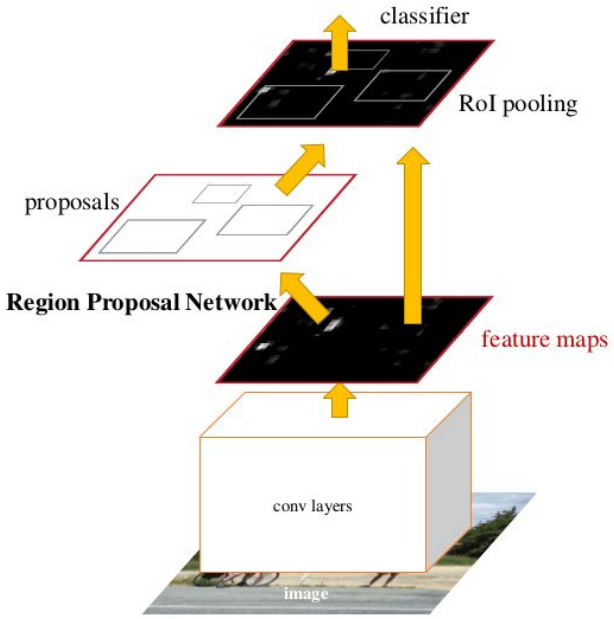


Figure 5 - Faster R-CNN architecture. A feature map is generated from the CNN backbone which is then used to create region proposals for all objects and ultimately classify the regions and build boulding boxes.

Next, a window is passed over the CNN feature map, where at each step different sized and shaped proposal boxes are applied to the feature map. These proposed regions are fed into a second convolutional network called the Region Proposal Network (RPN) (Xu, 2017). The RPN outputs a classification for each region (object or background) and a tightened bounding box for the object. Refer to Figure 6 for a schematic of the RPN functionality on top of the underlying CNN.

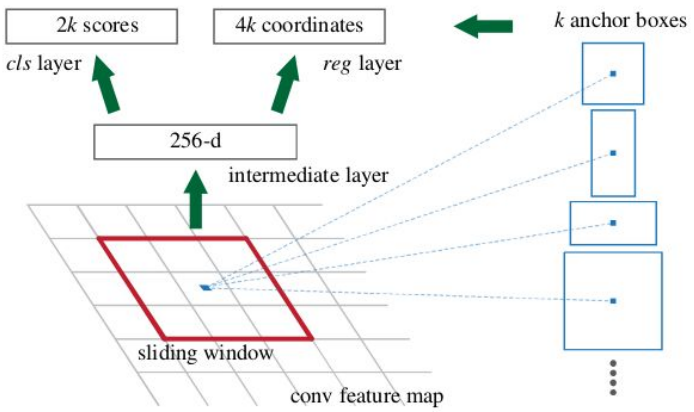


Figure 6 - Region Proposal Network. Regions are proposed according to sliding window and anchor box shapes. Region proposals are passed into convolutional network to generate classification scores and bounding boxes for objects.

At this stage in the evolution of CNN object detection, the model can identify each nucleus in an image and produce a bounding box for each nucleus. The last step in the segmentation and masking challenge of object detection is… masking! Incorporating masking into the R-CNN model is one of the more recent advancements in object detection. The Mask R-CNN model developed by the Facebook AI Research group is an example of an R-CNN architecture that has been adapted to producing binary masks (He et al. 2017). This was accomplished by first employing the Faster R-CNN architecture, then introducing a branch to the architecture as seen in Figure 7. The CNN backbone of Faster R-CNN feeds into an additional convolutional network, shown in white below the shaded Faster R-CNN architecture, which is designed to produce a binary matrix of 1s and 0s. The 1s represent all pixels that belong to a particular object while the 0s indicate background. A final step of applying the RoIAlign function to the binary matrix allows the masks to be accurately scaled up to the original image size and produce a more accurate mask of the segmented image (He et al. 2017).

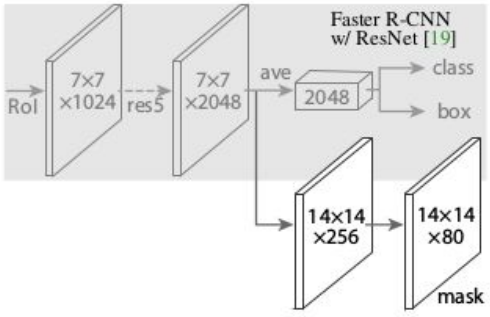


Figure 7 - Mask branch has been added to Faster R-CNN to develop Mask R-CNN architecture. Mask branch produces binary matrix to encode pixels as foreground (object) or background.

A key challenge in segmenting and masking nuclei found in the 2018 DSB dataset is the variance in size and color scheme of the input images. The Mask R-CNN model allows for varied image sizes and can scale images before feeding them into the neural network. For the purposes of this project, all images will be scaled to an input shape of 1024 x 1024 x 3. Images

with a height and/or width greater than 1024, will be scaled down proportionally. The input shape of 1024 x 1024 has been chosen to preserve nucleus size found in the larger images and increase nucleus size found in smaller images. Scaling down the larger images in the training set by any significant manner, such as 512 x 512, risks losing small nuclei.

The Mask R-CNN model requires a validation set to calculate loss during training. The validation set has been sourced from the training set by performing an 80:20 split on the data. 80% of the data, 536 images, are used for training and the remaining 20%, 134 images, are used for validation.

BENCHMARK

The performance of the Mask R-CNN model is demonstrated by comparing it to a benchmark model. Specifically, the mean average precision of the Mask R-CNN model is compared to that of a benchmark model. The benchmark model is based on classical computer vision techniques that are easy to implement and also generate modest results. For the purposes of this report, classical computer vision refers to the OpenCV computer vision library (or similar) for python. The implementation of the benchmark model can be found in the Jupyter notebook titled “capstone_benchmark.ipynb”.

The underlying approach of the computer vision used in the benchmark model is the separation of foreground and background in the image through statistical analysis of pixel color or intensity. Ideally, each image has a bimodal distribution of pixel intensity where a threshold intensity can be specified to classify all pixels into two groups. A foreground group and a backgroup group. Additional image processing techniques including opening and closing can be used to improve separation of adjacent nuclei. Finally, all foreground objects can be counted and their mask areas calculated. The benchmark model holds at least one advantage over Mask R-CNN in that it requires no training as it employs no machine learning techniques. Predicting masks for the test images produces an MaP score of 0.221. This score demonstrates that the benchmark model is capable of performing segmentation and masking, however the results are relatively poor. We will see how this compares to an implementation of Mask R-CNN with the hopes that machine learning can significantly boost our results.

III. METHODOLOGY

DATA PREPROCESSING

The nuclei segmentation and masking challenge of the 2018 Data Science Bowl warrants several data preprocessing steps to improve results from the solution model. Essentially, we seek to reduce variance in the input data to reduce the scope of the features that the model must interpret. Before model fitting can begin, the training data must be reviewed for correctness and completeness. As previously mentioned, the training data includes images and corresponding masks for all nuclei in the images. These training masks were generated

using a combination of automated techniques and manual annotation from scientists and technicians. Being that the focus of the 2018 DSB is improving nuclei segmentation and masking, it's no surprise that the techniques used to generate the training masks aren't without error. For this reason manual annotation was used to improve the training masks, but even humans make mistakes when tasked with large datasets. Needless to say, several errors exist in the training set, making fixing these errors the first preprocessing step. Figure 8 provides an example error found in the training set.

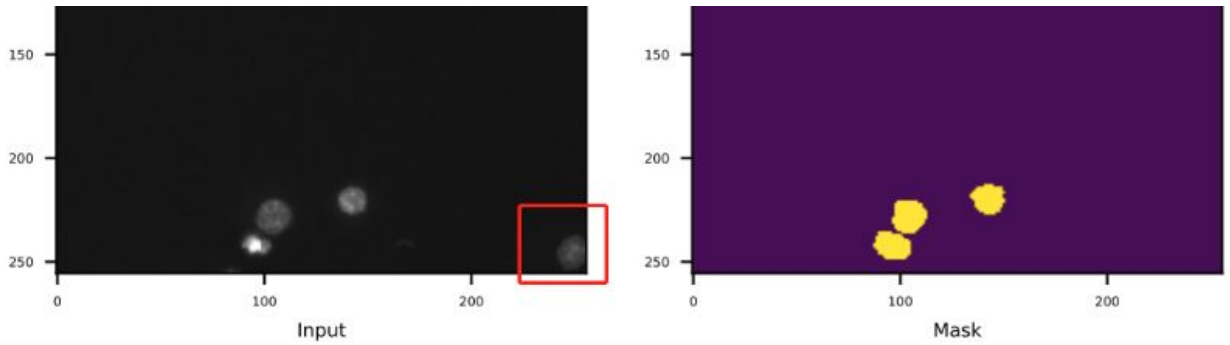


Figure 8 - Nucleus in bottom right corner was not segmented and masked in the provided training set. Manually annotating the masked training images like this one will improve solution model. (2018 Data Science Bowl - Data Quality Issues Discussion)

Due to the generous nature of Kaggle competition members, manually editing the training set was not necessary as a corrected training set has been publicly shared at: https://github.com/lopuhin/kaggle-dsbowl-2018-dataset-fixes.

The second data preprocessing step is adapting images to an input shape of 1024 x 1024 x 3. As previously described, images are scaled up or down to 1024 x 1024. In cases where images aren't square, the larger dimension, width or height, is scaled to 1024 and the shorter dimension is scaled proportionally to maintain original aspect ratio. Rectangular images are padded with zeros to fill the 1024 x 1024 input shape.

IMPLEMENTATION

Applying the Mask R-CNN architecture to the 2018 DSB challenge is carried out in two steps. These steps are summarized as 1) adapting existing classes in the algorithm to the 2018 DSB dataset and 2) fitting the model. The two important classes to inherit and repurpose are *Config* and *Dataset*. The *Config* class contains all of the readily adjustable parameters of the model including image min and max dimensions, learning rate, weight decay, and many more. These parameters are the focus of the second step in implementation; fitting and further refinement of the model. The *Dataset* class is essentially a skeleton on which to build the necessary functions for importing images and masks. The shape of the input matrix for importing images and masks is the most important detail when building the *load_image()* and *load_mask()* functions.

The *load_image()* function loads a single image when passed the directory path of the

image. This function must then return a three dimensional matrix with the shape: (image height, image width, image depth = 3). It is important to note that the depth of the image matrix must be 3 as the Mask R-CNN architecture is build to work with color images.

The *load_mask()* function loads all masks for one image when passed the directory holding the masks for said image. This function also returns a three dimensional matrix with a slightly different shape: (mask height, mask width, mask depth = # masks). Mask height and width should be the same dimensions as image height and width for the corresponding image. Mask depth is equal to the number of individual masks for the image where each slice in the depth axis contains a binary matrix for one mask. For example, an image with 10 nuclei would have 10 masked nuclei and therefore the *load_mask()* function would return a matrix with a depth of 10.

The dataset is prepared for use by instantiating *Dataset* objects, one for each subset: training, validation, and testing. The *Dataset* objects contain information about the dataset, such as file names and file paths as opposed to the images themselves. Each image and mask are loaded one by one as they are needed for training, validation, and testing to reduce memory load. The Keras implementation of the Mask R-CNN model used in this project can be found at the Matterport Mask R-CNN repository: (https://github.com/matterport/Mask_RCNN).

The Mask R-CNN architecture in this project uses a ResNet backbone as the underlying CNN. Microsoft has developed several deep convolutional neural network architectures such as ResNet50 and ResNet101. The ResNet architecture is composed of a series of “residual network” blocks as depicted in Figure 9 *(Xu 2017)*. The number in the ResNet name indicates the number of layers in the neural network.

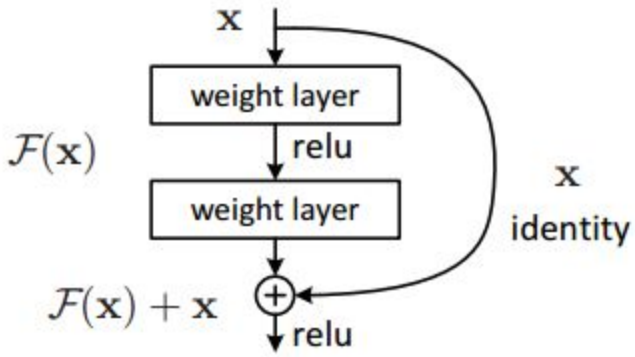


Figure 9 - ResNet block, the building block of ResNet50.

The Mask R-CNN model is instantiated with a number of parameters specified for the 2018 DSB dataset which remain constant for the duration of this project. Refer to the “capstone_main.ipynb” Jupyter notebook for these parameters and their prescribed values. This project makes use of pretrained weights for the ResNet architecture. The weights are generated from fitting the model to the MS COCO dataset and are available at the Matterport Mask R-CNN repository. Fitting the model to the cell nuclei dataset with a pretrained-weights starting point

leverages patterns and shapes that the model has been trained to recognize. A final variable in fitting with pretrained weights is selecting the number of layers to train and freeze. Preliminary testing has shown that training only the heads of the model, or training only the top few layers doesn't perform as well as training all layers. This project implements the Mask R-CNN model using MS COCO pretrained weights with all layers set to trainable for model fitting. By default, Mask R-CNN utilized stochastic gradient descent for weight optimization.

As mentioned, the model fitting step involves defining the parameters that will remain fixed through the refinement stage of optimizing the model. The parameters in question can be found in the *Config* class as seen in "capstone_main.ipynb" Jupyter notebook. An important attribute in determining the fixed parameters are identifying those that can be specified by studying the dataset. For example we know that the training dataset contains images with dimensions ranging from a minimum of 256 and a maximum of 1388. We can set *IMAGE_MIN_DIM* and *IMAGE_MAX_DIM* to 1024 to maintain the features of the larger images and also enhance features of smaller images. By setting the max dimension to 1024, the input image shape for the model will be set to (1024, 1024, 3). Other parameters that will not change during model refinement include but are not limited to *RPN_ANCHOR_SCALES, RPN_ANCHOR_RATIOS, DETECTION_MAX_INSTANCES, RPN_TRAIN_ANCHORS_PER_IMAGE, MAX_GT_INSTANCES*. Studying the training dataset shows that images contain up to 375 nuclei and about 99% of nuclei has aspect ratios between 0.25 and 4. Refer to the "capstone_inspect_data.ipynb" notebook for supporting analysis. It is likely that test images will display nuclei that differ from those in the training set, however parameters such as *RPN_ANCHOR_SCALES, RPN_ANCHOR_RATIOS,* and *RPN_TRAIN_ANCHORS_PER_IMAGE* must be limited or caped to allow for reasonable training times.

Variable parameters in this project are defined to be backbone architecture, learning rate, and object detection minimum confidence. These parameters are selected for exploration because determining their optimum values isn't easily deduced by studying the dataset. When performing gradient descent optimization it's particularly important to work with an appropriate learning rate. Too high of a learning rate will cause the model to jump over the global minimum error while too low of a value will prevent the model from finding a minimum in a time efficient manner. With this in mind, training is carried out with learning rates of varying orders of magnitude. A baseline number of epochs for model fitting is set to 10 while exploring the parameters in question. 10 epochs will provide enough iterations over the training set to tune model weights while conserving on time required to perform fitting.

Following training, the model is tested on the test set to determine a final score. As discussed in the *Metrics* section, the model is scored by calculating mean average precision of the predicted masks versus the ground truth masks. Run length encoding functions by parsing a dataset or file and storing data in terms of what values are present and the number of times they sequentially repeat (Run Length Encoding - GeeksforGeeks 2010). The full implementation of the solution model is seen in the Jupyter notebook "capstone_main.ipynb" and an identical

implementation in “main.py”. The python file is used for convenience of training the model on a cloud server.

REFINEMENT

Grid-search is used as the optimization technique for tuning parameters. The parameters of focus are backbone architecture, learning rate, and detection minimum confidence. An array of parameter setpoints is specified in an attempt to elucidate where, in the vast parameter landscape, the most performant model resides. Backbone architecture is tested in both the ResNet50 and ResNet101 configurations. The assumption made going into the architecture selection is that larger CNNs are more prone to overfit the data and suffer from error due to variance. For this reason, it is hypothesized that ResNet50 will better generalize to the dataset than ResNet101, and therefore ResNet50 will perform better in testing. Learning rate has much more freedom for experimentation and is tested at values of 0.01, 0.001, and 0.0001. These learning rates are selected as viable setpoints because these three orders of magnitude produced desirable results in the paper *Mask R-CNN* published by Facebook AI Research. The final parameter evaluated in grid search is detection minimum confidence, and is tested at values of 0.5, 0.7, and 0.9. This range of minimum confidence values is used to determine how much noise the model can tolerate when it comes to correctly identifying objects.

Grid-search is performed by adjusting the aforementioned parameters and fitting the model to the training data while keeping all other variables constant. Table 1 below displays model scores for each combination of learning rate and detection minimum confidence with a ResNet50 backbone. Table 2 displays a similar array of scores, but these are generated using a ResNet101 backbone.

ResNet50	DETECTION_MIN_CONFIDENCE		
LEARNING_RATE	0.5	0.7	0.9
0.01	0.407	0.392	0.403
0.001	0.401	0.400	0.348
0.0001	0.340	0.353	0.308

Table 1 - Mean Average Precision scores on test set with ResNet50 backbone.

ResNet101	DETECTION_MIN_CONFIDENCE		
LEARNING_RATE	0.5	0.7	0.9
0.01	0.416	0.380	0.402
0.001	0.438	0.416	0.406
0.0001	0.407	0.393	0.352

Table 2 - Mean Average Precision scores on test set with ResNet101 backbone.

Table 1 shows the optimal configuration of a ResNet50 Mask R-CNN model, given the scope of this project, to have a learning rate of 0.01 and a detection minimum confidence of 0.5. This model configuration produced a high score of 0.407.

With ResNet101 providing the underlying CNN backbone, Table 2 shows that the optimal configuration has a learning rate of 0.001 and a detection minimum confidence of 0.5. The ResNet101 version produced a slightly higher MaP score of 0.438. This refinement exercise demonstrates that the most performant Mask R-CNN model, for the purposes of this investigation, is parameterized with a ResNet101 backbone, a learning rate of 0.001 and a detection minimum confidence of 0.5. Grid-search can be particularly helpful in hinting at trends in model performance as a function of parameter setpoints and may suggest which parameters have the greatest impact.

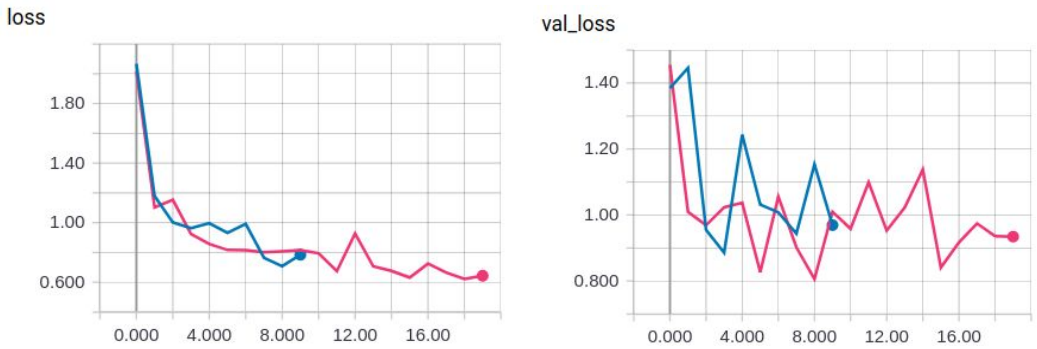


Figure 10 - Model training with LR = 0.001 and Det. Min Confidence = 0.5 for 10 and 20 epochs.

In an attempt to further improve MaP of the best performing model, the model is trained once more using a larger number of epochs. Increasing epochs provides more iterations for neural network weight optimization. This final model fitting experiment trains for 20 epochs using a learning rate of 0.001 and detection minimum confidence of 0.5. Surprisingly, this model doesn't produce lower validation loss as seen in Figure 10. It appears that the lowest validation loss occurs at epoch 9 (shown as 8 in the plot). The model weights at epoch 9 achieves a MaP of 0.433. After exploring extended training, the solution model is chosen to be the model found using grid search that generated an MaP of 0.438 seen in Table 2.

IV. RESULTS

MODEL EVALUATION & VALIDATION

The single strongest argument for selecting Mask R-CNN as the solution model to the nuclei segmentation and masking challenge is the research discussed in the *Mask R-CNN* paper. This paper details the specific purpose and strengths of the Mask R-CNN model which are object detection, object segmentation, and object masking. The Mask R-CNN architecture is designed for applications such as the 2018 Data Science Bowl. Furthermore, the Mask R-CNN architecture is demonstrated as an appropriate solution for this project by reviewing scores shown in Table 1 and Table 2. We can see that the solution model achieves substantially higher scores than the benchmark model.

The robustness of the solution model to variable input data is demonstrated by the use of training and testing datasets that contain divergent imaging techniques and image sizes. The models ability to adapt to variable input data is further tested and observed by applying data augmentation to the training set. This augmentation includes image rotation, and shearing. Specifically, half of the training set was augmented by a 45 degree rotation and a 15% shear. The training data augmentation effectively replaces half of the data with new images. The solution model training was replicated using the ResNet101 backbone, LR = 0.001, and detection minimum confidence of 0.5 over the course of 10 epochs with the aforementioned training data augmentation. This model generates an MaP score of 0.423, only 0.015 less than the model trained on un-augmented data.

A priority in developing the solution model and algorithm, as stated in the 2018 DSB competition description, is robustness to variable input data. In other words, the model must generalize well to images of nuclei sourced from different microscopy and imaging techniques. This is evident in the dataset provided for the competition. Training and testing images were taken using techniques such as bright-field, histology staining, and fluorescence microscopy. Training the model on these divergent imaging techniques helps prevent overfitting and forces the model generalize to a wide range of data. The aptitude of the Mask R-CNN model implemented in this project can be seen in the previous Refinement section. Another encouraging factor when considering this model is it's relatively high performance when compared to the Kaggle leaderboard.

JUSTIFICATION

The objective of this project is to help push the state-of-the-art technology in image segmentation and masking with the specific application of cell nuclei masking. The highest score produced by the Mask R-CNN solution model is encouraging and suggests that this model is well suited to the problem. When the solution model is compared to the classical computer vision benchmark score, the advantages of regional convolutional neural networks

become more obvious. The solution model's highest MaP result of 0.438 is a 98% improvement over the benchmarks score of 0.221. The higher score of the Mask R-CNN model is attributed to the fact that convolutional neural networks are designed to be trained on a dataset which allows them to learn the patterns and features of the dataset. By contrast, classical computer vision relies on hard-coded rules that do not adapt to a dataset and are only as performant as the coding built into them.

When comparing the solution model's score to the Kaggle leaderboard, we can see that this particular model likely hasn't uncovered new insights in nuclei segmenting and masking. The top of the leaderboard includes scores ranging from 0.567 to 0.634 that were produced by experts in cell nuclei masking and other forms of computer vision. While this comparison doesn't invalidate the use of Mask R-CNN as a solution model, it only suggests the high level of difficulty found in this challenge. In fact, several of the competitors at the top of the Kaggle leaderboard employ variants of the Mask R-CNN architecture and also make use of advanced pre-processing and post-processing techniques.

V. CONCLUSION

FREE-FORM VISUALIZATION

Figure 11 below shows several images from the test set and their respective mask predictions. Qualitative analysis of these examples finds that the mask predictions are fairly accurate and approach the performance of manual annotation by a human. Figures 11 and 12 in particular demonstrate the solution models ability to accurately predict nuclei masks. The low MaP scores seen in the *Refinement* section are a result of predicted masks often being smaller than ground truth masks. False positive predictions and false negative predictions are also a major contributor to low scores. Coloring in the ground truth and predicted mask images is only used to differentiate between nuclei.

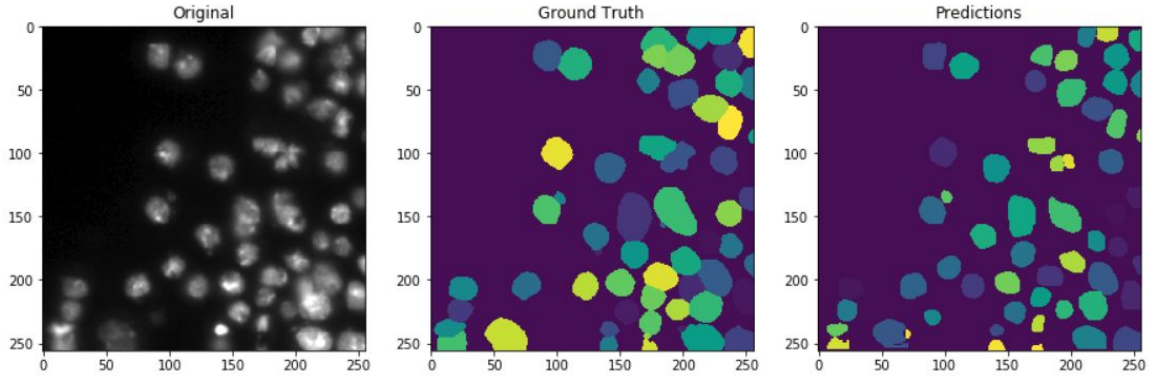


Figure 11 - Model successfully predicts most masks. Masks are smaller on average than ground truth mask.

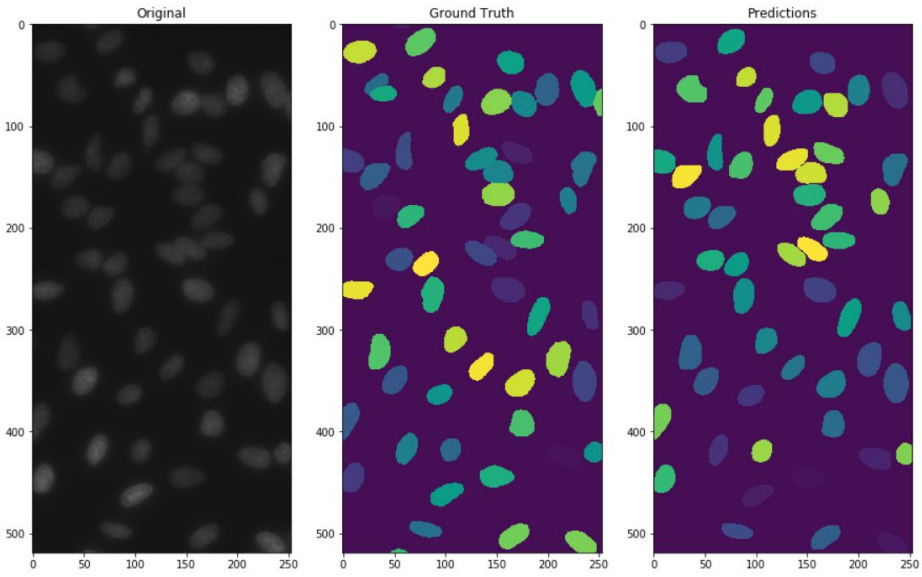


Figure 12 - Model accurately predicted most nuclei masks.

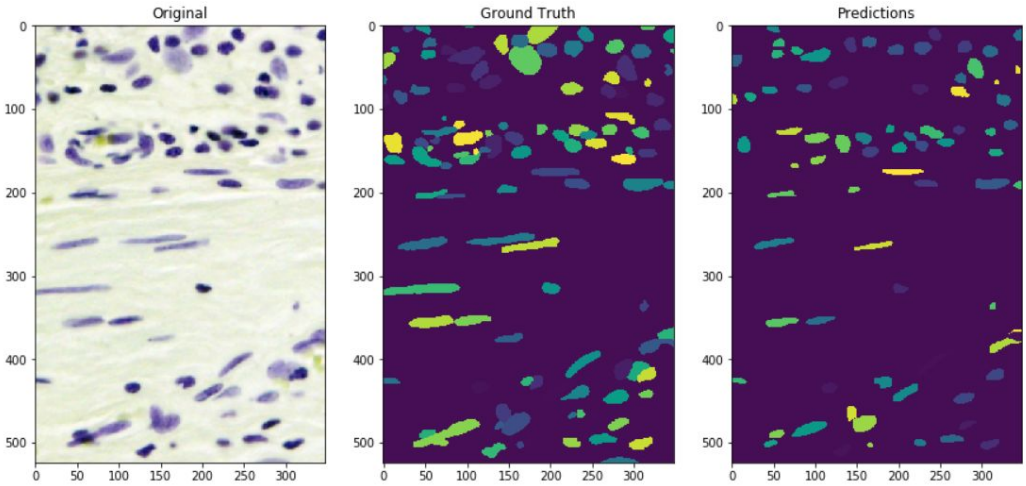


Figure 13 - Input image is clearly much more challenging for model to accurately predict. Most masks are smaller than ground truth and many are missing altogether.

Figure 13 sufficiently reflects the shortcomings of the solution model as most predicted masks are smaller than the corresponding ground truth mask and many masks failed to be predicted.

REFLECTION

To recap the objective and achievements of the project, object detection and segmentation is a highly challenging yet immensely useful domain within computer vision. Research and development of convolutional neural networks has enabled huge gaines in tasks such as cell nuclei segmentation and masking for a multitude of reasons. CNN's are particularly adept at learning patterns and features present in image datasets and being able to apply these generalized features to new images. The invention of models such as Mask R-CNN shows how the feature map output from a single CNN can serve several purposes from classification to multiple object detection, and even masking all detected objects with pixel level accuracy.

The 2018 Data Science Bowl presents several challenges in image segmentation and masking. To start, the training data consists of images sourced from multiple microscopy and imaging techniques allowing nuclei to take a wide range of sizes and colors. Additionally, nuclei can take on a variety of shapes from being perfectly circular to being very elongated ellipses and everything in between. A final challenge work noting is the potentially very high number of nuclei in a single image with nuclei often overlapping. The training set contains images with as few as a single nucleus to as many as 375 nuclei.

The solution model implements Mask R-CNN with a ResNet101 backbone and a training scheme that tunes weights in all layers of the CNN. The ResNet101 backbone is initialized with pretrained weights that were generated by fitting Mask R-CNN to the MS COCO dataset for common object detection. The model is then fit to the 2018 DSB training dataset using stochastic gradient descent optimization over the course of 10 epochs. Grid search is used to study the effect of varying learning rate and object detection minimum confidence. Model refinement produced the MaP score of 0.438 with a learning rate of 0.001 and detection minimum confidence of 0.5. The mean average precision metric uses intersection over union as the underlying precision measurement. IoU is a common technique for quantifying performance of image segmentation techniques as it reflects precisely how well a predicted mask lines up and overlaps with a ground truth mask.

A particularly interesting finding in this project goes hand-in-hand with the challenge of the project; segmenting and masking object with high accuracy is very difficult. Cell nuclei are relatively simple shapes that usually circular or elliptical in construction. Going into this project, it was assumed that sophisticated models such as Mask R-CNN that employ deep CNN backbones would easily learn the patterns and features present in images of nuclei. By contrast, the model struggles to generalize to the many different sizes and shapes of nuclei. Included in this project is varying color of nuclei and background which further complicates the task of generalizing. It’s difficult to specify all parameters of Mask R-CNN that will help the model generalize well across nuclei that vary by size, shape and color.

While the solution model demonstrated significant gains over the benchmark model, I feel that further improvement is necessary before it would be an appropriate real world solution. The final results of the Kaggle 2018 DSB show that better performing image segmentation and masking techniques do exist. The highest achieving submissions produced scores in the range of 0.609 - 0.631. Some of these techniques include image preprocessing and post-processing to assist the CNN model in detecting nuclei and make the predicted masks more accurate. For example, many predicted masks are smaller than the ground truth mask, prompting the question that perhaps expanding predicted masks in post-processing could improve results.

IMPROVEMENT

Two elements of possible improvement are easily identified at the conclusion of the project. First, model training could be significantly lengthened to further approach a global error minimum. In particular, the number of epochs could be increased and learning rate could be adjusted to incrementally smaller values to narrowing on the best performing neural network weights. For example, the model could be trained using a schedule of 30 epochs at a learning rate of 0.001, another 40 epochs with LR = 0.0001, and a final fitting step of 50 epochs with LR = 0.00001. By decreasing the learning rate, we can find the global error minimum by taking many small steps in this convex error space. A second avenue for model improvement would come in the form of broadening the grid search to cover more parameters. Much of the success of Mask R-CNN comes from applying proper parameter values according to the dataset. Perhaps parameters such as *RPN_NMS_THRESHOLD, DETECTION_NMS_THRESHOLD,* or *MASK_SHAPE* could be further explored to better understand their relationships with the model's overall performance.

As hinted at in *Reflection* above, the solution model would function as a useful benchmark model for designing an improved model and algorithm for image segmentation and masking. The Kaggle 2018 DSB leaderboard alone shows that better solution models exist. This makes the solution model's implementation of Mask R-CNN a good starting point for further parameter exploration and enhanced preprocessing and post-processing development.

VI. References

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2017.322

Meijering, E. (2012). Cell Segmentation: 50 Years Down the Road [Life Sciences]. *IEEE Signal Processing Magazine*, *29*(5), 140–145.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Lecture Notes in Computer Science* (pp. 234–241).

Xu, J. (2017, August 14). An Intuitive Guide to Deep Network Architectures – Towards Data Science. Retrieved April 3, 2018, from https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41

2018 Data Science Bowl, https://www.kaggle.com/c/data-science-bowl-2018

2018 Data Science Bowl Evaluation, https://www.kaggle.com/c/data-science-bowl-2018#evaluation

2018 DSB Creative Commons License, https://www.kaggle.com/c/data-science-bowl-2018/discussion/47864

https://www.geeksforgeeks.org/run-length-encoding/