Mulitloss Instance Localization & Segmentation

Praveen Kumar
27 April 2018

I. Definition

Project Overview

Drug discovery and research is very essential for improving quality of life for everyone on the planet. Historically, the number of drugs that can be developed for a billion dollars of R&D spending has been decreasing exponentially over the decades, as described by Eroom's law. For example, in the 1950s, one could develop 30 new drugs for a cost that wouldn't even find a single drug today. One major bottle neck in drug development is identifying cell nuclei in microscopy images. Researchers often test thousands of compounds and their variants on cells to find the one that might be a good drug. Researchers prepare batches of cells and apply a different compound to each batch and take microscopy images. Then they identify which batch responded well by looking to find a batch in which cells became healthier.

The first step in identifying the cell characteristics in the images that the researcher is studying is identifying the nuclei. Since nuclei of cells are at the center of the cell body, a stain that reveals nuclei gives an image of the cell batch in which each cell can be distinguished without much overlap on others. There are hand coded algorithms that do a decent job in identifying cell nuclei when the nuclei are well rounded in shape, regular and not very crowded. Sometime nuclei have very unusual shapes. Sometimes cells can be hard to distinguish in a tissue sample. This can sometimes result in the researcher manually hand label cells by looking at thousands of microscopy images. This is highly time consuming, and time that could have been put to better use.

The dataset used for this project is from the 2018 datascience bowl competition hosted on kaggle[0] by Broad Institute of Harvard and MIT. Details about the nature of the dataset will be discussed in a later section of this report.

Problem Statement

The goal of the project create a robust deep learning model that can understand what nuclei look like and identify them in microscopy images regardless of cell type, tissue sample, stain used, scale, illumination, microscopes, resolution, experimental setup, etc. without human (biologist) intervention.

The machine learning problem to solve is as follows: A multi loss, regression (localization and mask generation) + classification (nucleus detection with respect to background) problem that takes microscopy images of cell cultures, detect cell nuclei, localize them and generate image masks representing each nucleus present in the input image.

Strategy

The problem is split up into two stages: 1. detection and bounding box localization of nuclei present in the image, 2. segmentation of the nuclei present in each bounding box.

For localization a region proposal network (RPN) will be built along the lines of the approach used in Faster-RCNN[1]. Localized detection and bounding box predictions will be implemented as one efficient deep convolutional network.

Once we have the localized instances with bounding boxes, the bounding box crops of the image will be passed through a FCN (fully convolutional network) segmentation network with architecture inspired by U-Net[2] to generate instance masks.

Evaluation Metrics

The solution will be two independent systems, one for localization and another of segmentation. The two parts of the systems have different objectives and have different loss metrics. They will be evaluated seperately.

1. RPN (region proposal network) : This will be a multiloss network with a simple cross entropy classification loss for object detection at each quantized image location and a smooth L1 regression loss for bounding box co-ordinates.

2. Segmentation network : The segmentation network takes bounding box crops from the RPN and tries to predict an image mask. The loss metric for the segmentation network is cross entropy over each pixel between predicted mask and ground truth mask.

Also, I make use of another simple classification network purely trained to identify nuclei from an arbitrary background to filter out the regions proposed by the RPN before segmentation. This is a simple binary classifier with cross entropy loss over it’s predictions.

II. Analysis

Data Exploration

The source of the dataset used for this project is from the 2018 data science bowl[3] competition as stated previously. The dataset consists of 670 png images of cell cultures taken by microscope.

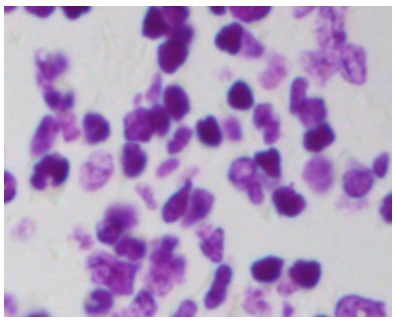

Fig. 1. A sample image from the dataset

These are all 3 channel color images with pixel values ranging from 0-255. The images vary in width, height from around 200 to above 500 pixels and consist of various cell types, imaging techniques, illumination, resolution, scale/zoom, setup, etc. All in all, the data set contains ~29,400 mask-labeled nuclei. Which comes to an average of 44 nuclei per image. Each image is provided with seperate binary image mask for every nucleus in the image

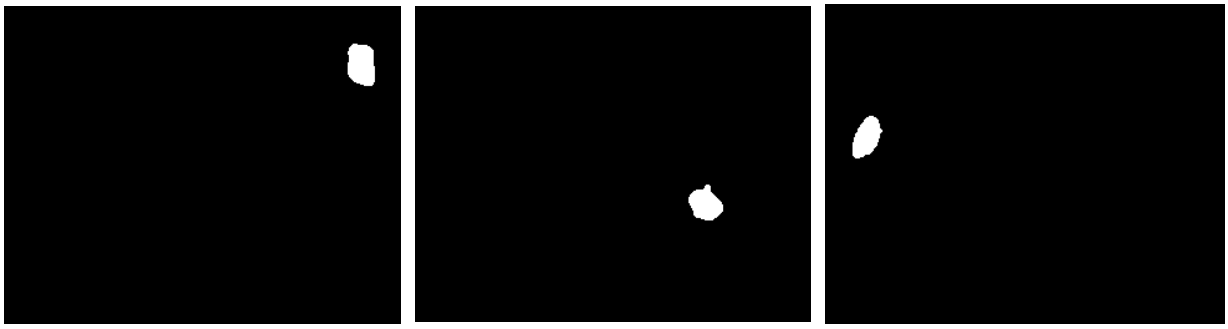

Fig. 2. Binary masks representing nuclei in the input image

The data set will be split into training and validation sets. Around 10% of images will be kept as part of the validation set.

Exploratory visualization

A few important statistics are needed to decide on determining certain network parameters. Statistics showing variation in input image sizes and mask sizes will give us some insight into how deep and large the network and feature maps are going to be, how wide the field of perception will be and also it will help us in coming up with appropriate kernel sizes so as to capture the details of the image well in the network. Also, these statistics are need to determine how much of the training data can be fit into the GPU for parellization.

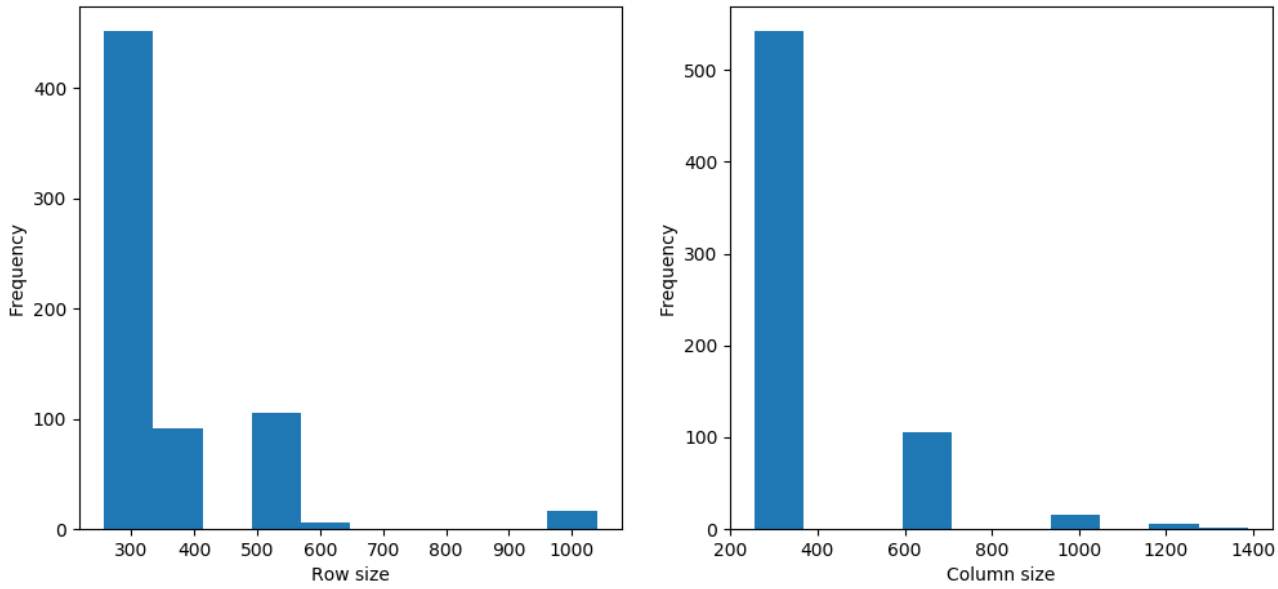

Fig. 3. Histogram showing column and row sizes in pixel length of input images

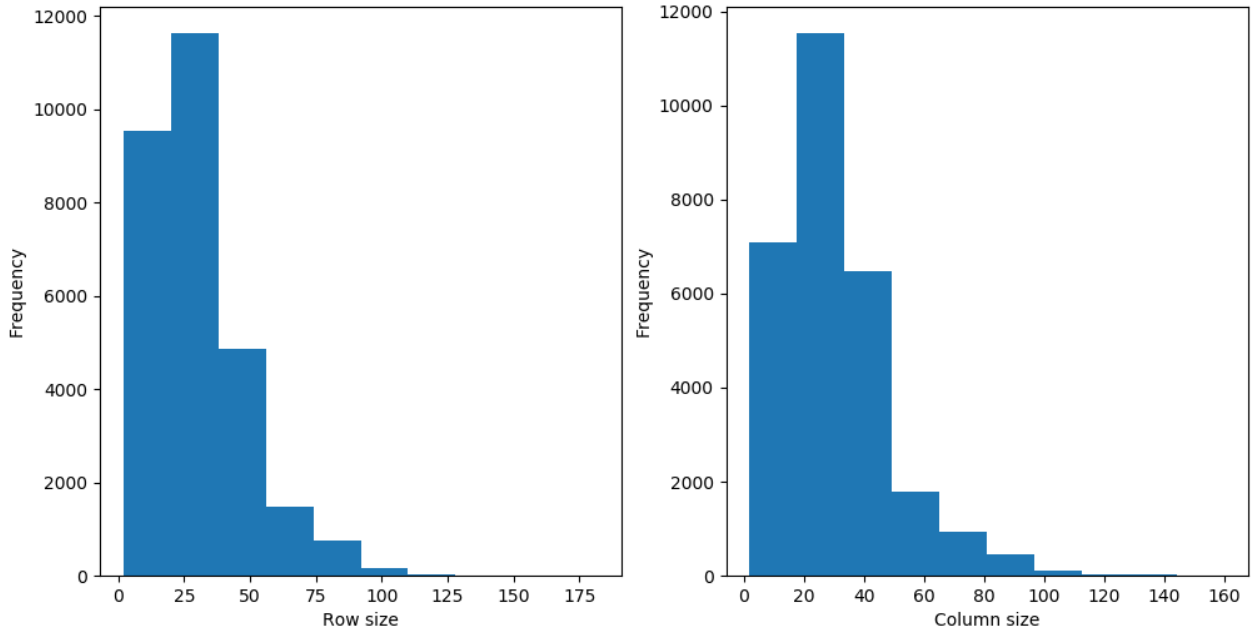

Fig. 4. Histogram showing column and row sizes in pixel length of nucleus masks

These visualizations how most input images have width and height around 300 pixels and object masks have width and height of around 30 pixels. These are just rough heuristics that I will base my choice of network parameters. Since most object masks have side length of around 30px, I will make my stand alone classifier take object crops resized to 32px. As we can see that side length of object masks has a maximum around 150, I will design the RPN network to have a perception field around 150px. Object size distribution will also be usefull in selecting the scales and aspect ratios of anchors to be used by the RPN.

Algorithms and Techniques

As mentioned earlier, the solution will be of two parts, an RPN and a Segmentation network.

The RPN places potential bounding boxes called anchors with different scales at each strided location on the input image. The stride length depends on how much the convolutional network down sizes the spatial resolution of the feature maps w.r.t input image as discussed in [1]. The classifier head of the RPN predicts nucleus detection scores for each anchor.

The bounding box predictions by the RPN is offsets to center co-ordinates and height, width of anchors. These predicted offsets are applied to anchors which are classified as a positive detections by the classifier head and a smooth L1 loss calculated between predicted bounding box co-ordinates and nearest ground truth bounding box co-ordinates.

Once we have a set of region proposals from the RPN, we apply a simple algorithm called non maximum suppression (NMS) to remove highly overlapping region proposals. The algorithm sorts the region proposals in the order of their object scores from the classifier head. In a set of region proposals with overlap over a certain threshold, the region with the highest object score is chosen while removing the others from the set of proposed regions.

After NMS, we filter the remaining regions with a stand alone nucleus detector and pass the selected regions through a FCN segmentation network.

Benchmark

For the purposes of setting a baseline benchmark to compare the solution's performance, the networks' evaluation before training will be used. Here are the resutls of the solution system on validation dataset before training:

Sub-System	Validation Loss
RPN classifier head	0.6865, Cross-entropy
RPN bounding box regression head	0.0277, Smooth-L1
Stand alone binary classifier	0.7350, Cross-entropy
Segmentation network	0.8074, Cross-entropy

As we can see, all of the solution networks showing high validation loss since the networks are untrained. This will serve as a good baseline for comparision with the trained models.

The loss for RPN bounding box regression is an order magnitude lower than other cross entropy losses this is because of the nature of Smooth-L1 loss and also becuase during training and validation only those anchor bounding boxes that have a high overlap with ground truth bounding boxes are chosen for regression. These anchors are already quite close to ground truth and thus resulting in low apparent loss. During application, the anchors are chosen based on scores from the classifier head of RPN and later modified by the bounding box regressor to better envelop the object.

III. Methodology

Data Preprocessing

Data preprocessing will be discussed in context of the three subsystems of the solution as all of them are trained on the same data but separately with slightly different preprocessing steps.

1. Region Proposal Network: RPN operates on the full resolution input image without any resizing as it needs to preserve spatial information to produce localized outputs. One common transformation applied to input images for all subsystems is converting the 3 channel color images to grayscale. Further the pixel values are normalized in the range 0-1.

2. Stand alone binary classifier: The objective of this network is to perform detection for a small image to decide whether or not the image contains a nucleus. For this purpose, all the nucleus instances in the input images of the dataset are croped using object masks provided in the dataset. These nucleus crops are resized to 32 x 32 pixels, converted to grayscale and normalized in the range 0-1. These crops serve as positive examples for the classifier. For negative examples, images from CIFAR10 data set is used since they are of similar resolution, scale and size.

3. Segmentation network: The data fed into segmentation network is similar to that of the binary classifier, nucleus crops resized to 32 x 32 pixels, converted to grayscale, normalized between 0 and 1. For output targets, a 32 x 32 pixels resized crops of the object masks around the position of the nucleus. These target images are also graycaled and normalized to 0-1.

The choice of the size for nucleus and target image crops was determined by observing from the analysis section that most nuclei have side lengths of around 32 pixels and the nuclei are mostly well rounded in shape.

Implementation

1. Region proposal network: This network's architecture is based on the region proposal network from [1]. RPN starts with a common backbone network to extract features for both the classifier head and bounding box regressor head. This backbone network is based on the architechture of the standalone binary nucleus detector. This architecture is inspired by the residual architecture of RESNETs [4]. The backbone is 9 layers deep with relu activations and batch normalization. The first layer has a kernel size of 5x5 and the rest of convolution layers have kernel size of 3x3.

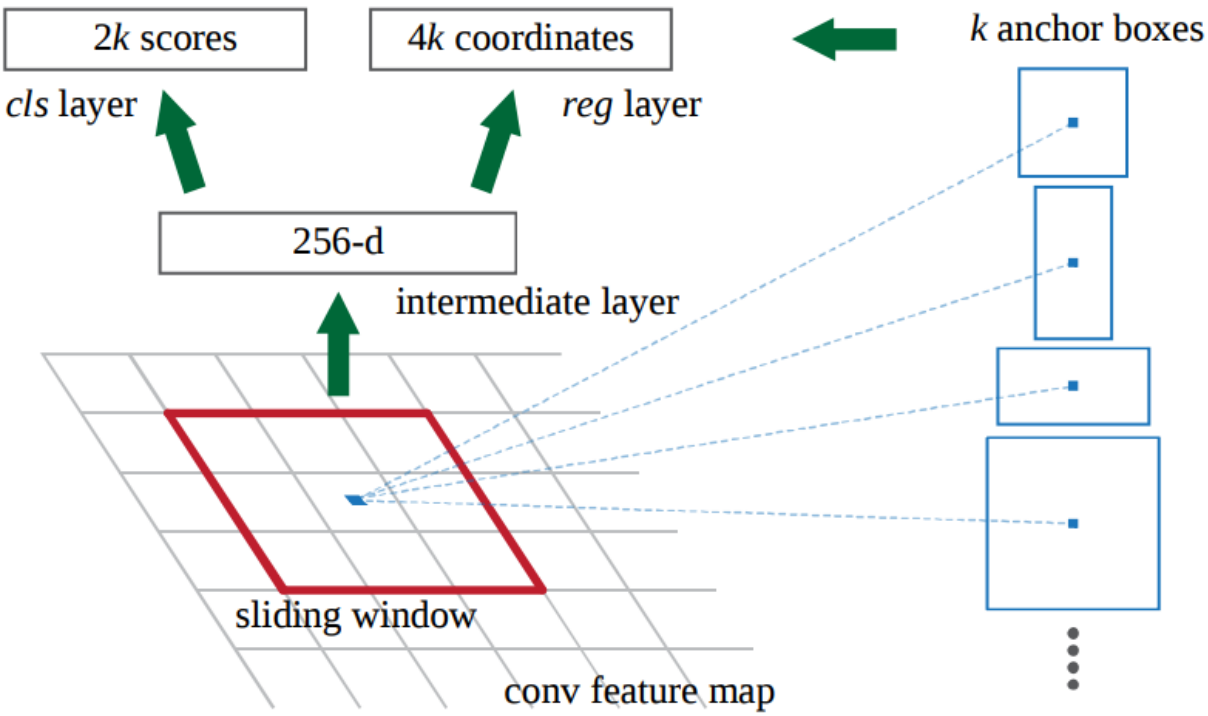

Fig. 5. Classifier and bounding box regression heads

On top of this feature map, 3x3 kernels are used to perform convolution (stride 1) and produce a 256 or 128 channel output (in my implementation, 128 channels are used). This will be fed into two different 1x1 convolution paths. One of the path will produce classification scores and the other will produce bounding box regression. The number of output channels of the two heads of the RPN is parameterized on *k*, the number of different anchors used. Anchors are potential bounding boxes at each location on the backbone feature map. I have used various anchor sizes and aspect ratios starting from 8x8 to 128x128. The anchor sizes used can be found in the *constants.py* file. The *k* anchors are placed at every location on the backbone feature map but are in turn projected on to the

original input image. Since there will be spatial compression from the input to backbone feature map, the anchors will be placed at strided locations and the stride length depends on the compression ratio. In my backbone implementation, anchors are projected on the input image at every 8 pixels.

The classifier head predicts binary classification for every anchor at each spatial location thus resulting in 2*k* outputs at every location. This is achieved as a 1x1 convolution with 2*k* channel output. The bounding box regressor head produces corrections to center co-ordinates, width and height of each anchor. Thus at every spatial location on the feature map, the regressor head produces 4*k* output. This is implemented as a 1x1 convolution with 4*k* channel output.

2. Stand alone binary calssifier: This is a stand alone network for purposes of filtering proposals from the RPN. This network architecture is exactly same as the backbone network with a fully connected layer to produce classification output.

3. Segmentation network: This network's architecture is based on U-Net[2]. This is a 8 layer deep fully convolutional network with relu activations and batch normalization which preserves spatial resolution with appropriate padding. All the layers in the network have kernel size of 9x9.

Once we have region proposals from RPN, we perform non maximum supression with over lap IOU threshold of 0.15 and further filter out the region proposals using the stand alone classifier. Then the proposed regions are croped from the original image, resized to 32 x 32 and passed through the segmentation network to get the instance masks.

Training

During training, there is quite a bit of work generating groud truth data for RPN from the training dataset. For the classifier head, we need to generate a 2*k* x *W* x *H* tensor representing classification targets for every anchor placed over the backbone feature map. *W*, *H* are width and height of backbone feature map. The target tensor is generated in the following way. From every object for an input image in the training dataset, we generate bounding boxes around the object mask and encode the co-ordinates in a tensor. A similar tensor is formed to represent the co-ordinates of all the anchors. We calculate intersection over union (IOU) overlaps between every ground truth box and every anchor. Any anchor with IOU with any ground trugh box greater than 0.5, we consider them as positive targets and anchors with IOU less than 0.1 with all ground truth box are considered negatives. Rest of the anchors are ignored. Classification target tensor is constructed based on which anchors are considered positive and negative based on their IOU with ground truth boxes.

In a similar fashion, those anchors considered positve, their co-ordinates and height, width is used to construct a 4*k* x *W* x *H* tensor as the target for bounding box regression.

For classification head loss, we mini batch the positive and negative anchors (with mini batch size 100) with equal distribution and calculate cross entropy with the predictions. For bounding box regression, we only consider the positive anchors to calculate regression loss as L1 loss. The bounding box co-ordinates are parameterized as in [1]. Also, the cross boundary anchors, anchors with edges beyound the boundary of the input image, are ignored during training and during testing, they are used with their cross boundary edges clipped at the input image boundary.

For the stand alone classifier and segmentation network the nature of input and target data is discussed in *Data Preprocessing* section.

For RPN and instance segmentation network, we start off with a learning rate of 0.001 and decay the learning rate by a factor of 10 every epoch and run for 4 epochs. For the stand alone classifier we start with initial learnig rate of 0.0001 and decreasy by factor of 10 every epoch for 4 epochs.

Refinement

Initially, I started the development of RPN with a shallower 6 layer backbone network for feature extraction. This backbone had an effective perception field of 92 x 92 pixels on the original image. This was turning out to be a problem as there were plenty of objects that were bigger than 92px and training was not converging to an acceptable loss value. Later, I built a deeper backbone with more pooling layers to achieve a perception field of 160x160 which resulted in much better training. Also, ignoring the cross boundary anchors saw a big boost in bounding box regression performance. This was done as suggested in [1].

The stand alone classifier started out as simple feed forward convolution network. This architecture yielded a cross entropy loss of around 0.3 but later, after introducing residual skip connections, the validation loss dropped to less than 0.05 which is a huge imporvement.

The segmentation network started out as a residual network. The masks generated had cross entropy more than 0.5. After researching to find a good architecture, I ended up choosing U-Net for it's implicity. Implementing U-Net like architecture brought down the cross entropy to around 0.3.

IV. Results

Model Evaluation and Validation

The nature of the dataset is such that it, in it's self, provides a very varied set of data points with different illuminations, shapes of target object, scale and staining techniques used. After all, the data set was designed by the creators of the dataset with the object of creating a very robust solution to the problem as part of the kaggle competition. Also, the during training of the stand alone classifier and segmentation network, the resizing operations were effectively providing randomly resized, scaled, morphed images since the image crops selected were of various sizes and aspect ratios. The model is validated only on unseen data. The training loss of the model correlates well with the loss on validation set. This is a good indicator that the model is generalizing well.

Justification

Here is a table summarizing the results of the solution:

Sub-System	Validation Loss	Improvement over benchmark
RPN classifier head	0.0983, Cross-entropy	85.6%
RPN bounding box regression head	0.0202, Smooth-L1	27.1%
Stand alone binary classifier	0.0122, Cross-entropy	98.3%
Segmentation network	0.3853, Cross-entropy	60.0%

As we can be seen, the solution performs significantly better than the benchmark in all aspects. The bounding box regressor of the RPN shows a less drastic improvement compared to other sub systems. This is because of the nature of anchor selections. Once the classifier head selects the anchors for the regressor, the anchors are already pretty well placed over the target object and approriate scale and aspect ratio of the anchor box is also determined by the classifier head and thus there is only so little modification that can be done on the selected anchors. As we see in the visualization below, even with this seemingly low improvement, the regressor does a good job bounding the target objects and also we see the anchor selections done by the classifier.

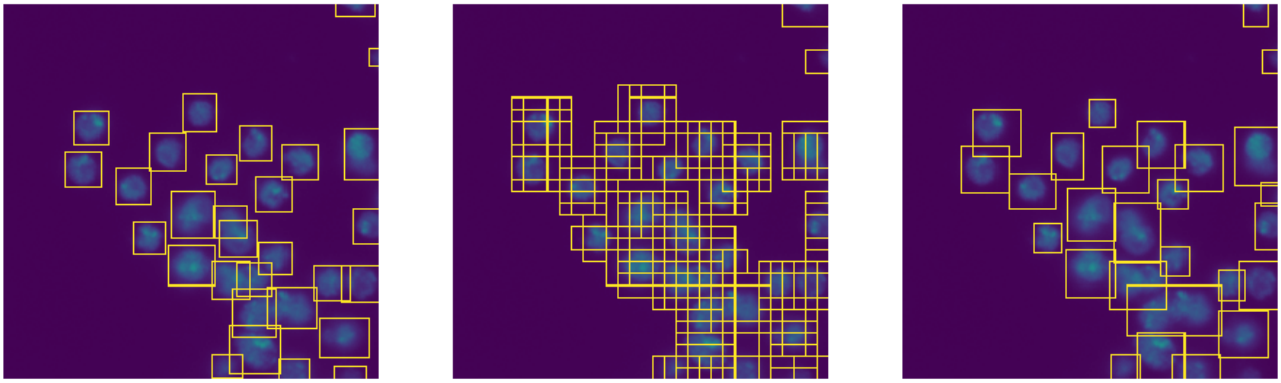

Fig. 6. (left) ground truth bounding boxes, (middle) anchor selection, (right) predictions by RPN after NMS

There is a lot of room for improvement for the segmentation network. Due to time constraint, a simple, crude imitation of U-Net was implemented.

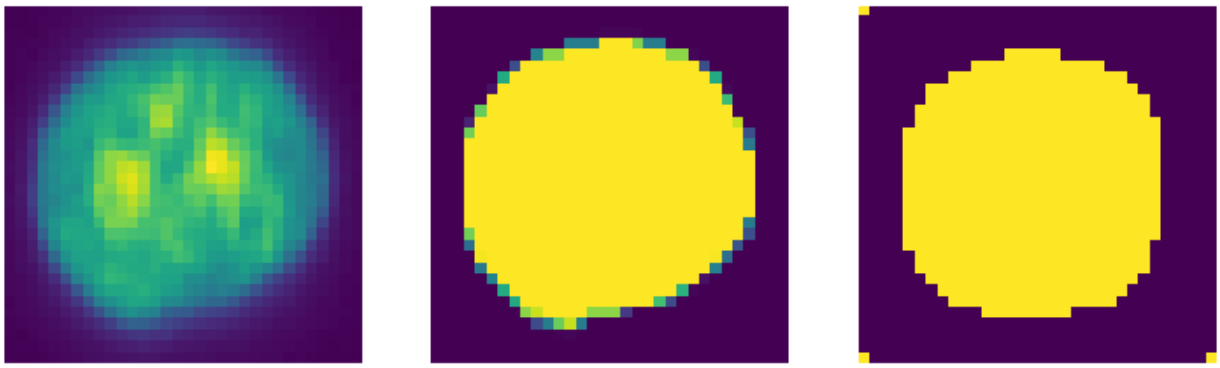

Fig. 7. (left) input object crop based on RPN proposal, (middle) ground truth mask, (right) mask prediction by segmentation network

V. Conclusion

I have trained a region proposal network to localize objects and a segmentation network to generate masks for object instances.

It was very interesting to see that a common feature extraction network could be used to minimize two different branches with very different goal and loss function.

The biggest problem faced during this project was trying to get the mask generation network to share the same feature extraction backbone as the localization network. Also, choosing the right threshold for NMS in filtering region proposals is very cruicial and data dependent. Selection of the right anchors also makes a big difference in RPN performance.

With time, there is a lot that can be improved. The segmentation network can benifit a lot by using a feature pyramid architecture and conditional random fields or a more faithful implementation of U-Net. Experimenting more with the backbone architecture, anchors and overlap thresholds can potentially improve performance of the system.

References

[0] Kaggle is a platform for hosting machine learning competitions: https://kaggle.com

[1] Faster RCNN: https://arxiv.org/abs/1506.01497

[2] U-Net : https://arxiv.org/abs/1505.04597

[3] 2018 data science bowl : https://www.kaggle.com/c/data-science-bowl-2018

[4] Deep residula learning : https://arxiv.org/abs/1512.03385