Machine Learning Report

Sam Forbes, ID: 1629094, March 2018

Abstract

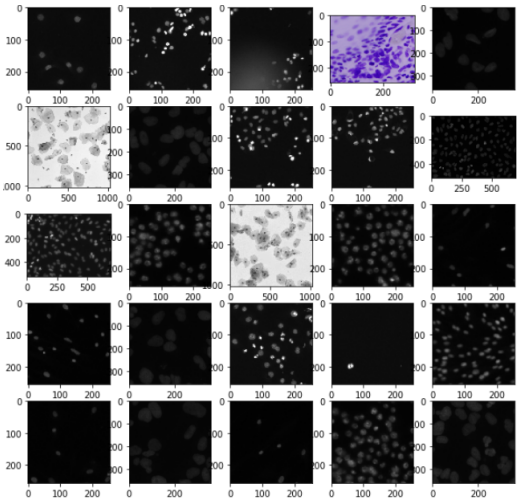
The task was to predict nuclei in images. Models were trained on a training set and then a test set was used for predictions. An error metric was evaluated on the predictions and the corresponding actual nuclei. These actual nuclei were in the form of masks where the background has one pixel value and the nuclei another. The masks of the individual nuclei could be combined to form a mask containing all nuclei for an image. The error metric used was intersection over union (IoU). This corresponds to the proportion that the predicted nuclei overlaps with the actual nuclei masks. 0 would be no prediction of nuclei, 1 would be complete prediction of nuclei. The task was part of the kaggle 2018 Data Science Bowl to find nuclei in divergent images to advance medical discovery. When submitting the predictions for the images in the test set, kaggle computed an IoU metric with the corresponding masks which it kept hidden. The best IoU metric I recieved when submitting to kaggle was 0.254 with Otsu image thresholding. My next best was a convolutional neural network using the U-Net model which gave around 0.19. A multi layer perceptron got an IoU metric of 0.13. According to other kaggle users the U-Net model should have given a higher IoU of at least 0.27 indicating an issue in my implemenation.

Data Augmentation, Feature Engineering, and Segmentation

Task 1

The training data was a set of 670 images containing nuclei. The images had a lot of variation. Most were shades of grey, some had colour, usually shades of pink and purple, some had many nuclei, some had very few, see Figure 1. The variation in the size and shape of the individual nuclei was also high. There was a mean of 44 nuclei per image with a minimum number of 2 and a maximum of 376. The mean size of nuclei was 560 with a minimum size of 69 and maximum size of 1243 (pixels^2), see Figure 2. The image sizes were in different categories with most having a pixel size of $(256, 256, 3)$ where the first two entries are the dimensions of the image and the last entry relates to red, green and blue image channels for colour, see Figure 2. There were also nuclei which appeared to be overlapping so it was hard in some images to tell the exact number of nuclei. The masks of the individual nuclei in an image did not overlap according to kaggle.

Figure 1: Sample of 25 Images of Nuclei from 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

(a) Number of Images of Each Size in Training Set

	mask_counts	nuclei_size_mean	nuclei_size_min	nuclei_size_max	nuclei_size_std
count	670.000000	670.000000	670.000000	670.000000	670.000000
mean	44.971642	560.483462	69.270149	1243.029851	306.627435
std	47.962530	634.752932	197.402043	1346.599675	359.936976
min	2.000000	26.000000	21.000000	26.000000	0.000000
25%	16.250000	150.521825	24.000000	287.250000	65.940617
50%	28.000000	265.434343	33.000000	634.500000	148.111038
75%	55.000000	674.434370	63.000000	1733.000000	394.974610
max	376.000000	7244.071429	4367.000000	11037.000000	1965.426189

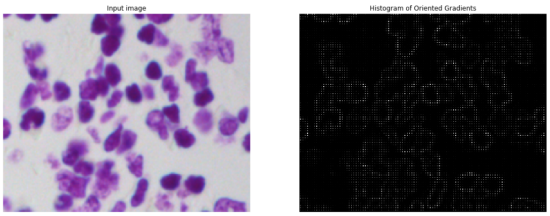
(b) Min, Max, Quartiles, Mean and Standard Deviation of Nuclei Size in Training Set

Figure 2: Some Statistics of Training Images

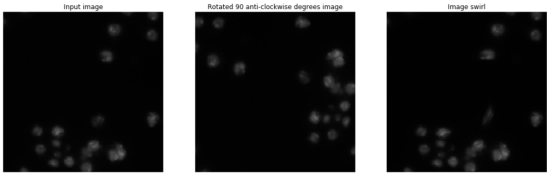
Task 2

Three different types of segmentation were applied to an image. I chose to apply histogram of gradients, data augmentation and watershed augmentation. The histogram of gradients did not appear to be very effective. There were many types of data augmentation that could be applied including rotation, swirling and shearing. I applied rotation and swirling to an example image. See Figure 3 for the segmentations applied on a few random images.

(a) Histogram of Gradients



(b) Data Augmentation: Rotation and Swirl



(c) Watershed Segmentation

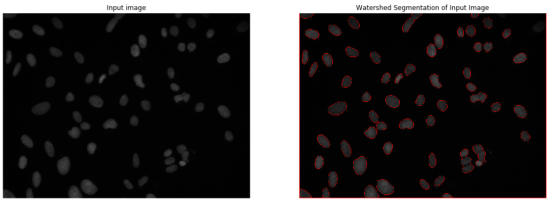


Figure 3: Image Segmentation

Task 3

I applied thresholding to predict the masks of the actual nuclei. I used a threshold of the mean pixel size in each image which gave an IoU of about 0.09. An Otsu threshold worked a lot better and gave an IoU of 0.254 after sumission to kaggle. Otsu thresholding works by choosing the threshold that minimises the spread of the two classes of thresholded pixels, see ref **1.**

Machine Learning Models: Artificial Neural Networks and Deep Learning

Task 4

I used a multi-layer perceptron (MLP) with the scikit learn Multi Layer Perceptron Classifier to fit a neural network to the 670 training images and the corresponding masks. I used one hidden layer of 100 neurons. The input and output layers were both 256 (size of resized images). I could only fit images ignoring colour (the third column in the dimension of images) as the MLP classifier function did not allow a 3 dimensional array to be fitted with the masks. After fitting the classifier I predicted on images in the training set. I compared these predicted images to the corresponding images of masks to make sure the model predicted to a reasonable accuracy. Predicting images in the training set and using cross validation I found that adding extra layers did not improve the accuracy and too many layers made the accuracy worse, likely due to overfitting.

The chosen number of hidden layers and neurons in each layer for the best MLP can not be known exactly beforehand. However one or two hidden layers and a neuron count between the input and output layer seems to be reasonable, see ref **8.** Therefore using 256 neurons may have been better than 100 but I did not test this.

A lot more different variations in the number of layers and neurons could have been tested, however as an MLP is fully connected it would be computationally expensive to compute too many neurons or layers to all 670 images and their masks.

The highest IoU score I recieved when submitting to kaggle was 0.131. This was the MLP with 100 neurons when trained on the first 300 images. Training on the full set of 670 images actually gave a lower score.

Task 5

I used the same MLP with one hidden layer of 100 neurons to fit the original images together with another 300 augmented images and corresponding masks of original and augmented images. The augmented images were augmentations of a random selection of the original 670 images using ImageDataGenerator in the keras python module. The masks of images that were augmented were correspondingly augmented. The augmentations used were a random rotation between 0 and 50 degrees, a random shear range up to 0.5, a random zoom, width shift and height shift range of up to 0.2 and a reflect fill mode. The zoom zooms in on the images, the shifts realign objects in the image from the centre and the reflect fill mode fills points outside the boundaries of the nuclei with a reflection. These augmentations were chosen as they did not alter the original images in an unrealistic way. Again having two layers did not change the accuracy to a significant degree when using ten fold cross validation. The prediction gave an IoU score of 0.132, virtually unchanged from Task 4. To keep the the IoU progression graph in Figure 6 simple I left this value off the graph as I submitted much later than when I did the previous MLP and it adds little information.

Task 6

An initial convolutional neural network (CNN) was fitted to the images and corresponding masks using sequential from keras. This had 7 convolutional 2D layers of sizes 8, 8, 16, 16, 32 and 1. Padding was used in each layer. The final layer had a sigmoid activation. The predictions for this CNN gave a poor IoU score of 0.099 after submitting to kaggle. I also tried a similar model with more convolutional 2D layers with greater sizes: 8, 8, 16, 16, 32, 32, 64, 64, 128, 128, 32, 32, 16, 16 ,8, 8, 1. This did not predict the training images well so I did not submit. I think both these models did not work well as I did not run on many epochs and there was likely to be overfitting due to an absence of max pooling. Also in the second model I used dilation rate in some of the layers. This extends the field of view which may not be beneficial in this problem to detect nuclei.

In theory a CNN is better to use than MLP for image recognition as it takes into account that pixels close to each other tend to have more of a relationship than pixels further away. It does this by

'convolving' through separate regions of the images and is more efficient than an MLP which creates weights for every pixel in an image in every fully connected neuron, ref **10.**

A variation of the U-Net model, see ref **6.** & **9.**, shared by other kaggle users was deemed to give good results. The model has convolutional 2D layers going from 8, 16, 32, 64, 128, 256 and then back down to 8 and a final layer of size 1 with sigmoid activation. Max pooling was used after each convolutional 2D layer was implemented as the layer size went up and convolutional 2D transpose layers, convolutional 2D layers, concatenations and dropouts were used as the size of the layers went down. The optimizer used was 'adam', a stochastic gradient descent algorithm.

After training on more and more epochs (> 50) the IoU score went up to 0.18 after an initial score of 0.129 on submission to kaggle. I then used this same U-Net model with another 600 augmented images (using the same method as in Task 5) and this gave an IoU submission score of 0.19 with 17 epochs.

The way I implemented the model was to put all the pixels of the training images and corresponding masks into two arrays after being resized. The images were resized to $(256, 256, 3)$ and masks were resized to $(256, 256, 1)$ (the most common size of images). When predicting on original and augmented images, the augmented images and masks were concatenated to the end of these two respective arrays. These two arrays were fitted to the U-Net model with the number of epochs specified as 50, batch size as 16 and cross validation on 0.1 of the images. Binary cross entropy loss and an IoU metric was used on each epoch train and test. The mean IoU metric used from a kaggle kernel, ref. **6.**, was not reliable and gave a value too high. The binary cross entropy loss went down to around 0.07 and decreased slightly erratically through the epochs starting at about 0.45 on the first epoch.

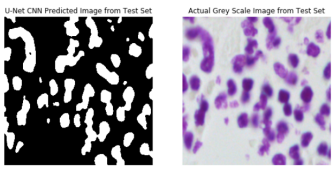
The fitted model gave prediction probabilities when predicting an image. These values were then thresholded at 0.5 to give predicted masks.

To submit to kaggle the model was used to predict masks of the 65 images in the test set, thresholded at 0.5, then the individual predicted nuclei masks were converted to Run Line Encoding (RLE) code (codes for predicted masks in a succinct way). The RLEs for each mask were put into a data frame as one column and the corresponding image IDs as another. For this I used functions from kernels on kaggle with some slight adaptations, ref. **2.** & **3.**

The submissions to kaggle for Tasks 3, 4 and 5 were done in a similar way however no thresholding was needed as predictions were in two binary classes rather than probabilities.

Figure 4 shows predictions of two images from the test set using the U-Net model fitted on the training set and 600 extra augmented images and corresponding masks. You can see how the irregular shaped nuclei in (a) are not predicted very well as nuclei close together are predicted as one nuclei. More regular nuclei that are mostly spaced evenly apart in (b) are predicted a lot better.

(a) Bad Prediction of Masks



(b) Good Prediction of Masks

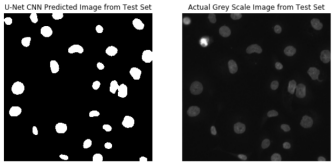


Figure 4: Image Prediction on Two Images from Test Set with U-Net

Progression Graph and Discussion

Task 7

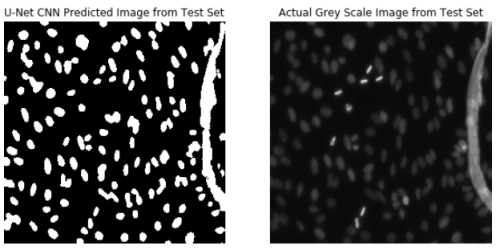
My progress in terms of IoU score from kaggle submissions went from 0.086 with mean pixel size thresholding straight to my best score of 0.254 with Otsu thresholding. It then went back down to 0.131 with an MLP on the first 300 images which was near enough the highest I recieved with MLP. I then fitted CNNs. The initial CNN with no max-pool layers got a score of 0.099. A U-Net model was then fitted and recieved 0.129 after 10 epochs, this increased to 0.161 with another 10 epochs and 0.178 after another 20 epochs. I managed to increase this to 0.193 with another 600 augmented images added using keras datagen and got a similar value with 2500 extra augmented images. Unfortunately this was still well below the Otsu threshold and the value of 0.27 which a kernel on kaggle I had adapted claimed to recieve, see ref **6.** Figure 6 shows my progression graph which ends at March 6th as submissions after this date did not improve the IoU score and were mainly tests with the same U-Net model with more epochs or on more augmentations.

If I had more time I would try to understand why my implementation of the U-Net model gave such a low score in comparison with kernels on kaggle. I would also try to increase my understanding of what was happening in the layers of MLPs and CNNs and how to choose them for image recognition of nuclei. I would look into research on more CNN models for image recognition and try to test them. I would do more implementation on smaller sets of images. I wasted a lot of time running ineffective models for many hours on the entire training set. Also I would try to understand why the IoU metric I was using from kaggle gave unreliably high values far off the actual kaggle submission value.

In terms of preprocessing the data I would try different resizings of the images. Perhaps I would separate images into their respective sizes and then train a model on each size and then combine models, but this may not be possible. I would experiment with more types of augmentations and more augmented images and possibly figure out how to separate nuclei that were joined before training a model on them.

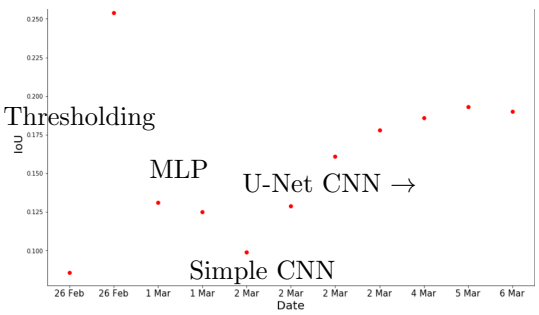
Overlapping nuclei and resizing were probably the main factors that reduced accuracy. Also there were some images in the test set that caused large errors due to (apparent) stains in the image predicted as nuclei, see Figure 5.

Figure 5: Image Stain Predicted as Nucleus



It seems like the hardest to predict nuclei were small ones and irregular shaped ones so I would look into this. Finally finding more images and masks and using them to form a larger training set may improve the score.

Figure 6: IoU Progression



References

1. http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html
2. https://www.kaggle.com/stkbailey/teaching-notebook-for-total-imaging-newbies
3. https://www.kaggle.com/kmader/nuclei-overview-to-submission
4. https://www.kaggle.com/jerrythomas/exploratory-analysis
5. https://www.kaggle.com/pudae81/data-visualization-and-analysis
6. https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277
7. https://www.kaggle.com/c0conuts/unet-imagedatagenerator-lb-0-336
8. https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw
9. Ronneberger O, Fischer P, and Brox T: U-Net: Convolutional Networks for Biomedical Image Segmentation (2015), arXiv:1505.04597
10. http://cs231n.github.io/convolutional-networks/