# CS342 - Machine Learning, Assignment Two

U1500912

March 14, 2018

# 1 Abstract

This coursework required us to follow a set of tasks involved with the 2018 Kaggle 2018 Data Science Bowl. This assignment has demonstrated how important spatial reasoning is for image processing tasks. Models such as the MLP are unable to get an accuracy higher than most feature engineering methods, as they are unable to represent the problem in an accurate manner. Conversely models that can do this are able to get particularly good accuracy only when provided with a large enough starting data set. While Machine Learning models are very powerful they cannot learn from nothing and require a large set of initial training data, whether that be real or generated.

## 2 Task 1 - Data Exploration

This task required the generation of image masks, in order to identify cell nuclei. The first challenge of this task is to analyse the style and formatting that the data provided holds. An initial exploration of the data shows that the images are all stored as RGB and consequently comprised of 3 separate channels. These images then display a range of other characteristics. Most importantly of these is the way the cells are displayed, in some cases the cells appear as the darker patches, whereas in others they appear as light on a dark background as seen 2. Furthermore there is a reasonable variation in the image sizes, most notable of these being the lowest dimensionality is 256 by 256 4. In addition to this it is important to calculate the number of masks and consequently cells that we expect to see in each image 5, by analysing this we can observe whether the chosen model we are using is outputting the appropriate amount of data.

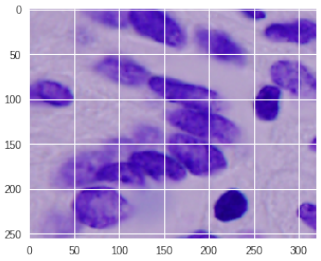


Figure 1: Colour Image

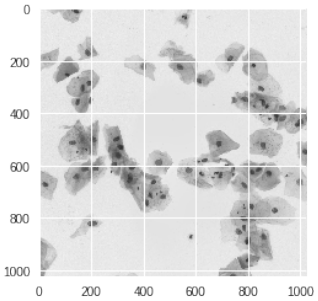


Figure 2: Dark Nuclei

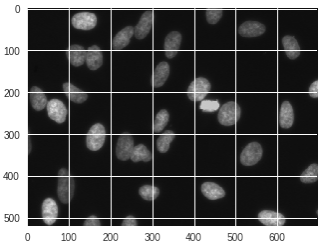


Figure 3: Light Nuclei

Due to the nature of machine learning models it will be important that the data is regularised before any data is passed into the model before passed in to it for training. This is as inconsistencies between the way in which nuclei are displayed will confuse the model and significantly reduce its accuracy. In addition to this the pre-processing will have to be applied to the test data so that the model is provided with data in the format it expects.

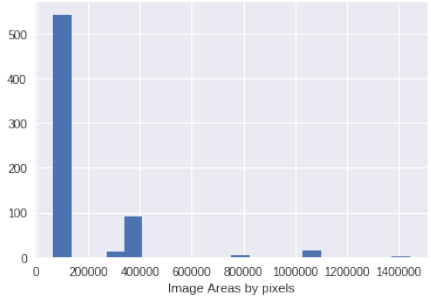


Figure 4: Images by pixel count

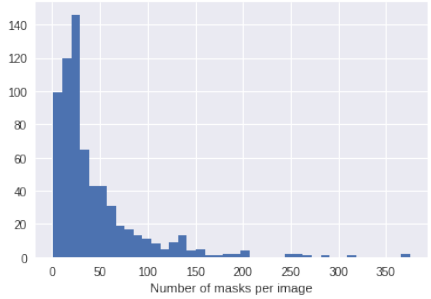


Figure 5: Masks per Image

This standardisation can take several forms. In order to deal with the image size discrepancies we will first want to convert all images to 256 by 256 format before making predictions and learning, the result produced will then need to resized relatively. The background-foreground separation issue can be solved through a variety of feature engineering or segmentation approaches. These methods will attempt to highlight or otherwise identify areas that contain nuclei. However while this will be effective at teaching an MLP in order to identify the location of nuclei it will have an adverse affect on a CNN. This is due to the fact that these methods will reduce the amount of data that the CNN and receives, as such it is better off being passed images with greater depth. As an alternative, the data can be normalised before being passed into the CNN by calculating the z-score. These standardised images can then be passed into the CNN in order to improve its learning.

Finally we can observe that the data set that we are provided with is actually relatively small. As a consequent of this we can assume that data augmentation will be an affective approach to improving model accuracy. This is as the rotation and flipping of images will effectively create new data on which models are able to train.

# 3 Task 2 - Feature Engineering and Segmentation

In this task we were required to implement a selection of Featuring or Data Segmentation approaches that we believed would be successful in aiding the analysis of the provided Dataset. Figure 6 displays the different results that can be obtained through the following methods.

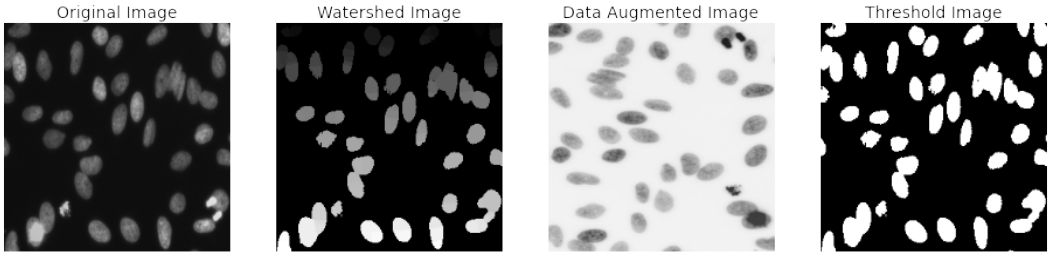


Figure 6: Feature Engineering

## 3.1 Threshold

Thresholding provides a consistent method to attempt to divide the nuclei from the background. On a pixel by pixel basis this task is a classification problem, wherein a pixel either is representative of part of a nuclei and should be white or should be ignored and set to black. There are a several variations of thresholding provided in the skimage.filters package. In its simplest for thresholding could be performed by taking the mean or median value of the total image, then assuming all values above that to be nuclei and below background. For this task I determined that OTSU would be the most effective. OTSU splits the image into a bimodal distribution wherein there are two distinct peaks of values representing the peak background and foreground intensities. It then finds the optimal separation value in order to distinguish the nuclei from the background Greensted [2010].

## 3.2 Watershed

Watershed is another possible image processing technique suitable for this problem. It functions by treating the image as a set of elevations, wherein the highest RGB pixel values essentially act as the peaks. After defining a value for the background, acting as the floor, the image is 'flooded' from the peaks. Once this is done any flooded areas that contain only a single peak are shown as single nuclei. In situations where there are multiple peaks in a single flooded area the gradient can be examined in order to determine whether there should be a split into two distinct nuclei.

## 3.3 Data Augmentation

Although unable to produce any results alone, data augmentation is the most important aspect of pre-processing for this problem. The data set provided to train on in this task is reasonably small. As a consequent of this it is difficult for the model to achieve a high accuracy. In order to compensate for this we can apply data augmentation, this methodology allows us to create additional data to train the model on. By applying the same transformations to the masks that you do to the image data you are able to ensure that the 'new' data you produce can be correctly learned from by your model. I achieved this by taking the image data provided and then using an image of rotation 90°, 180° and 270° respectively. This effectively quadruples the size of my dataset. It would be possible to exploit this yet further by using a series of flips, slices and scaling transformations to the data and its corresponding masks to generate as much extra data as possible.

# 4 Task 3

For this task I uploaded both a CSV files containing the predictions of a Watershed as well as a Threshold function. Receiving a score of 0.091 and 0.257 respectively. My threshold code is based heavily from Bailey [2018]'s work.

# 5 Task 4 - MLP

While not totally suited to this problem, MLPs do provide some level of accuracy. MLPs consist of a number of hidden layers sandwiched between an input and output layer, data is here passed from one end through to the other. As the MLP is fully connected, every node in each layer is connected to every other in the following layer with a certain weight. This weight is adjusted during the training phase in order to produce the most accurate model. When attempting to make a prediction the image data is passed through this optimal model in order to determine the mask.

The MLP treats this task as a classification problem by working through the image pixel by pixel, it attempts to use the rest of the image in order to predict whether the pixel being inspected is a 0 (background) or a 1 (nuclei). In doing so the MLP doesn't take into consideration the previous predicted values, or provide additional weighting to nearby pixel values. This makes the MLP poorly suited to this style of problem as the predictions it makes are not locally correlated. That is to say the location of 'nuclei pixels' should be influenced by the location of other nearby nuclei. The MLP instead attempts to view the image as whole and gives all other pixels equal weighting when predicting the value of its current inspection.

The MLP I eventually generated is reasonably simple, consisting of four layers with. I achieved the best results when squeezing the node count in the middle layers, this was mainly due to the overfitting. In addition to this I further added a few dropout layers into the model so as to ensure that it can be generalised further. Following testing I determined that the 'elu' activation model was the most appropriate for the layers to take as it produced the highest accuracy, however the final output needs to be 'sigmoid' so as to produce a value between 0 and 1 that will be the probability of classifying the pixel as a nuclei.

The lack of image normalisation is a significant issue in this model as there is such variation in the data. As shown in Figure 3 some images show lighter nuclei on a dark background and others dark nuclei on a lighter background. This only acts to confuse the model as it is torn between whether areas of dark or of light correspond to nuclei location.

My finally submitted MLP then received a score of 0.13

# 6 Task 5 - MLP with Data Pre-processing

In my prior testing I found the the MLP was incredibly accurate when trained and tested on the same data. This suggests that given a large enough data set the MLP will produce significantly better results as it is increasingly likely to have seen a similar image previously. Consequently I expected Data Augmentation to provide a significant boost to the MLP score. By applying simple rotations to the images in order to quadruple the size of the dataset it was possible to increase the MLP score to 0.189.

Applying thresholding to the MLP and Watershed, while improving the accuracy, does little to accommodate for the difficulty the MLP has with solving this task. The normalisation of this data makes the MLP more likely to predict pixels correctly as the information contained within the image is significantly reduced. Applying a Watershed to the training and testing data lead to a score of 0.143 and Thresholding with a score of 0.156.

The fact that the MLP combined with thresholding produces a lower accuracy than the thresholding itself is unsurprising. Thresholding alone does a reasonable job of creating a set of masks, the MLP is then provided an opportunity to edit these masks, however it is not given any additional data. This is requires the MLP to evaluate the accuracy of the generated data based on that alone. While this may be possible from a human perspective if one nuclei should clearly be split into two, due to the pixel by pixel analysis the MLP uses it will only ever reduce the accuracy of the mask.

# 7 Task 6 - CNN with Raw Pixel Data

Similarly to MLPS, CNNs are created as a fully connected set of hidden layers as well as an input and output layer 7. However in contrast CNNs are ideally suited to solve a problem of this nature. While the MLP works on a pixel by pixels basis, the CNN works on a sliding window approach. This means that the CNN is very context dependant and is able to classify nuclei using a spatially aware metric.

CNN layers can take one of several types. Convolutional layers are designed to emulate the response of a human neuron to visual stimulus, in this layer each neuron is assigned a small part of the image to observe and acts as a learnable filter. This area then iterates across the entire image in order to generate an activation map,

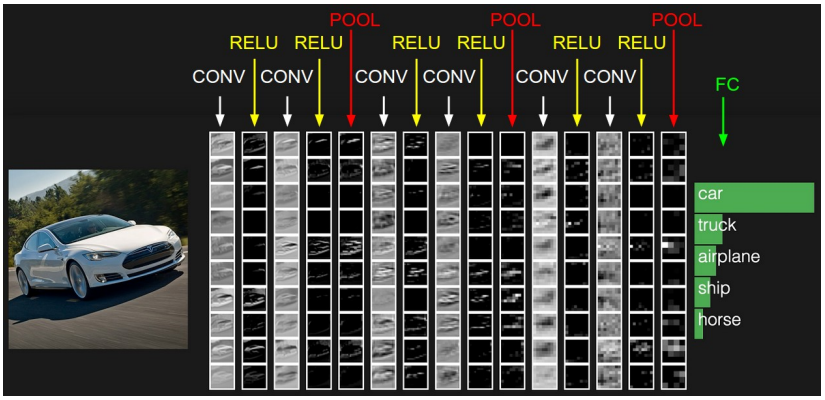


Figure 7: Standard CNN layers CNN [2018]

this displays the response of the filter to each part of the image CNN [2018]. As the learning process continues the model is then able to determine which filters are most accurately able to describe which segments of the image, for example edges of the nuclei. These nodes are then given a higher weighting and more influence over the result the CNN produces.

Dropout layers are used in the same manner as MLPs wherein a random proportion of the prediction data is discarded, this is done to avoid overfitting. To a similar end Pooling layers can be applied. Here each of the image windows is downsampled allowing the result to become more generic8. During the production of my CNN I chose to use MaxPooling whereby the maximum pixel value in the gird is taken as the downsampled value.

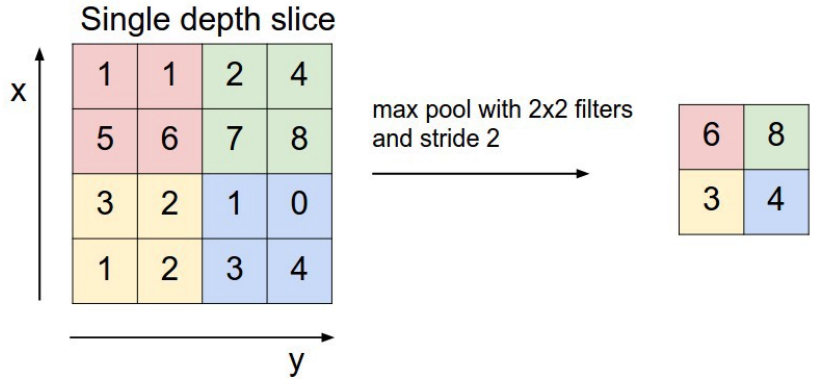


Figure 8: Max Pooling Abdelfattah [2017]

I began the design of my CNN with the creation of a series of Convolutional layers that start with fewer nodes before expanding into more complex layers and cascading down before the output. Following this I then added several large dropouts and MaxPooling after the more complicated layers. I believed that this would allow the Network to initially explore a large range of filters and distribute the node weighting accordingly, but then reduce the overfitting that this would result in by enforcing some generalisation with the MaxPooling layers, followed by large dropouts.

Through the use of callbacks while training the model I was able to save a copy of the best performing epoch. This allowed the model to continue training in hopes of achieving a better accuracy. However in order to avoid overfitting a validation set of 0.1 was set, finally I used a patience value of 5 in order to prevent the model from running when it was no longer improving.

When compiling the model I decided to use a optimizer of ‘adam’ as this was recommended by Jev during the lecture as well as a loss function of binary cross entropy as this is most accurate to the nature of the task, being representative of the number of bits that we will require should the data be encoded incorrectly.

The CNN I eventually submitted and provide with this report received a score of 0.247.

# 8 Task 7 - Progression Graph and Best Submission

My final submission for the Kaggle competition gave me a final score of 0.33, I created with the aid of the U-Net Kernel that is available on the site with a few variations. The biggest difference in scoring between my own submission and the kernel was the introduction of data augmentation. In addition to this I have changed the sizing of some of the layers so that it better reflects my original intention of an expanding and then collapsing set of nodes with a slightly higher dropout rate.

As can be seen from my graph displayed in Figure 9 I initially struggled to create a functioning MLP using the sklearn classifier, however upon switching to Keras I was able to quickly make progression. I understood quickly that the MLP was poorly suited to the problem and as such decided to halt on my first successful submission and simply apply the various feature engineering techniques to the model.

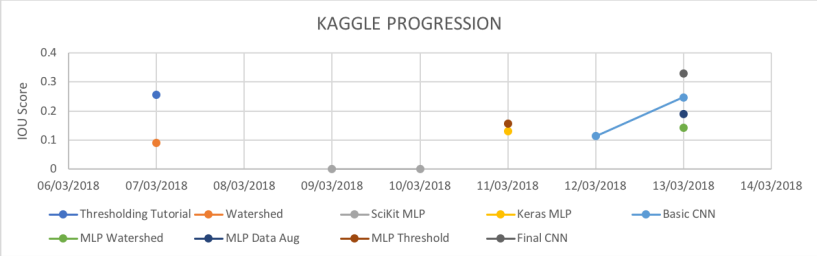


Figure 9: Kaggle Submission Progression

While I initially attempted to create a CNN from scratch the U-Net kernel provided a higher-scoring model that I based my final submission upon Savik [2018]. The model provided in the kernel was able to improve on my previous score for a number of reasons, most obviously is the number of layers implemented. The kernel solution implements 5 more layers with a reduced Node size in each, my initial belief was that this would lead to over-fitting, as multiple filters iterated on top of each other would simply produce the corresponding results to suit the training data. Instead I attempted to use a high node count with fewer layers on my original network so that the results would remain generalised with the best 'general' filters being provided the highest weighting. However, the over-fitting caused by the high layer number was simply manageable through the use of MaxPooling as well as dropout.

The are some simple changes to this model that could increase the score simply with access to greater computing power. Most obvious of these is the Data Augmentation, given more RAM it would be possible to increase the data set four fold again. Each image can be flipped (only a single time to reduce duplicate data) and additionally scaled and shifted. This increased computing power would also allow for a greater number of layers and nodes, while this could lead to overfitting I believe that the accuracy is still low enough that there is leeway before this will become an issue.

Outside of this the model still has issues with adjacent nuclei becoming merged into one, this is the largest cause of accuracy loss in the model currently. This could be resolved through some post processing such as binary opening wherein we attempt to shrink the masks and observe if any separations occur. The masks can the be expanded out, ensuring to keep the separations marked as individual nuclei. In a similar vein to this accuracy is loss with the misclassification of noise as nuclei, this could be avoided with a post processing technique which would erase masks that fall below a certain size. This size could be determined through an observation of the smallest nuclei present in the original dataset.

## Bibliography

Convolutional neural networks (cnns / convnets). http://cs231n.github.io/convolutional-networks/, 2018.

Abdellatif Abdelfattah. Image classification using deep neural networks. https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4, 2017.

Stephen Bailey. Teaching notebook for total imaging newbies. https://www.kaggle.com/stkbailey/teaching-notebook-for-total-imaging-newbies, 2018.

Dr. Andrew Greensted. Otsu thresholding. http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html, 2010.

Kjetil Amdal Savik. Keras u net starter. https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277, 2018.