Nuclei Detection - Classifying cervical cytology smears using a supervised learning approach

Brittaney Everitt
Queen's University
Kingston, Ontario, Canada K7L 2N8
15be6@queensu.ca

Reproducibility Summary

Scope of Reproducibility

In this project, I aimed to reproduce the results of the deep learning method reported by Phoulady and Mouton [1] to classify patches of cervical cytology smears as a nucleus (+1) or not (-1). The goal was to implement a model, as close to the model written in the paper, in order to achieve a precision around 0.861, recall around 0.895 and F-1 score around 0.878 on the testing dataset. I aimed to pre-process and generate around 654,948 image patches from the training set where about 1.71% of the patches have a positive label. I compared the results of my implementation to the results of the baseline segmentation method reported in the paper to determine if the deep learning approach for this detection task "outperforms" and is "significantly better" than the baseline segmentation method. I determined how many nuclei were not detected and missed in the output of the model.

Methodology

The cervical cytology smears were split up into a train and test set and then each image was pre-processed into patches of 75x75 pixels. The authors used a convolutional neural network architecture to classify each image patch and the code was not publicly available.

Results

Average precision, recall and F1-score of the test set were 0.88, 0.82 and 0.85 respectively, which was similar to results reported in the paper. 335,340 training image patches were created with 1.76% of patches labelled as positive. The percent of positively labelled patches was consistent with results reported in the paper but the number of patches created was about half the number reported in the paper. The implemented method's result was better than the baseline segmentation method's result reported in the paper. However, the ratio of the increase was not as high as reported in the paper. About 10.51% of nuclei were missed and not detected in the test set using the trained model output, which was consistent with the results reported in the paper.

What was easy

Loading in the data and separating the data into training and testing sets were easy. Setting up the deep learning model in Keras and Tensorflow, calculating the result metrics and post-processing of the image patches were straightforward.

What was difficult

The most difficult task was deciding on the implementation and techniques used to pre-process the images into patches. Determining hyper-parameters to use in the CNN architecture was difficult as the majority of them were unknown.

Communication with original authors

An email was sent to the authors which asked questions about the pre-processing, model and training. At the time of writing, there was no response from the authors.

33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

1 Introduction

Cytology is a method used to diagnose diseases by looking at single cells and small clusters of cells. This technique is key to diagnosing some types of cancer. Routine pap smears are one of the most common screening methods to detect for early stages of cancer and can be digitized to produce cervical cytology images. Analyzing the changes in the nucleus, specifically its size and shape, is used when determining if the cells and tissue are normal or diseased. Nuclei segmentation and detection can assist specialists with this task, where nuclei are first automatically identified in the image, followed by an automatic segmentation around the shape of each nucleus.

Based on previous research, cytology images are generally easier to train using deep neural networks and convolutional neural networks (CNN) because of their high-quality magnification level, compared to other types of pathology images such as histopathology. However, accurate segmentation and classification algorithms are still challenging in this field. There are not many papers on nucleus segmentation of cervical cytology images. The most commonly used deep supervised models applied to analyze cervical cancer are CNN models [2]. Whenever deep learning architectures are applied in health care, it is important to reproduce the method to test for robustness and safety as it can directly impact people's lives.

2 Scope of reproducibility

In this study, I implemented the deep learning method used by Phoulady and Mouton [1] which determines the probability that a patch in a cytology image is a nucleus or not. The main goal was to compare the implementation results to the results reported in the paper. The paper presents different methods, such as a baseline segmentation method. I only focused on reproducing and mirroring the deep learning method developed in the paper.

- Claim 1: The precision of the method was 0.861, the recall was 0.895 and the F measure was 0.878 [1]. All results reported were for the testing dataset
- Claim 2: 654,948 extracted image patches in the training set, with 1.71% positive labels (within 15 pixels of a manually marked point) [1]
- Claim 3: The CNN method "easily outperformed" the other methods that did not use deep learning and "achieved significantly better results than the other methods" (about a 6% improvement over the baseline method) [1]. The F measure was more than 5% better than the baseline method [1]
- Clam 4: The CNN method only missed about 10% of the nuclei (most missed nuclei were close to the boundary and/or were overlapping nuclei) [1]

3 Methodology

The author's code for their baseline segmentation method is available, however it does not use deep learning. The code for the deep learning method of the paper by Phoulady and Mouton is not publicly available. There is also no visual representation of the model, only a written description of the model architecture. Therefore, my approach is to re-implement the model based on the concise description in the paper. There is relatively little documentation from the paper on how the pre-processing was conducted or the environment the models were trained in. Therefore, to create the image patches I used MATLAB, and to train the models I used Tensorflow, and Keras libraries.

3.1 Dataset

A novel, relatively challenging dataset is used in the paper. The dataset is the Cervix93 Cytology dataset that is "representative of routine cervical cytology images from Pap smears" [1] due to the variability in the nuclei. There are 93 real extended depth of field images (at 40x magnification) of size $1280x960$ pixels and 2705 manually annotated nuclei. Each image is a grayscale image. Each image has one of three grade labels: negative, LSIL – low grade squamous intraepithelial lesion (abnormalities but not cancerous) and HSIL high grade squamous intraepithelial lesion which could be cancerous. The manually marked nuclei coordinates are recorded in a csv file for each image in the dataset and are publicly available. Nuceli on the boundary of the smear were marked as a nuclei "if more than half of it was located in the frame and if the center of it was roughly about 10 pixels from the boundary" [1]. The link to the dataset Github is: `https://github.com/parham-ap/cytology_dataset`

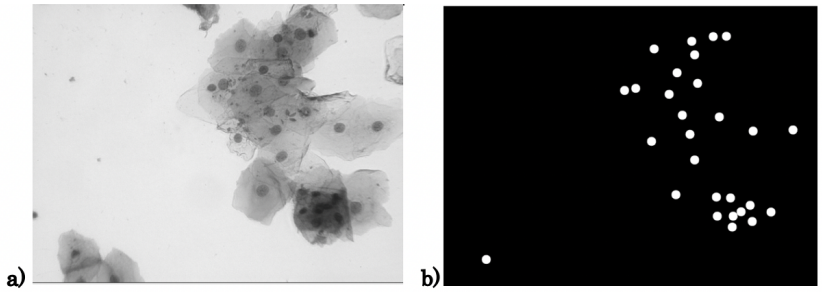

Figure 1: Image pre-processing mask generation. a) A full slide cytology image. b) A generated binary nuclei mask for the corresponding cytology image.

3.2 Image Pre-processing

The images (1-93) were divided into training and testing datasets based on the labels in the label.csv file. Images that were labelled with a 0 were put into the training set and images that were labelled with 1 were put into the testing set. There were 69 whole images in the training set and 24 whole images in the testing set. In total there were 2705 nuclei, 1983 nuclei in the training set and 722 nuclei in the testing set [1].

In order to use the images in the deep learning model proposed by Phoulady and Mouton, image patches were created following the pre-processing method described in Figure 3. I assumed that the top left corner was pixel (1,1), the first column was the horizontal "i" axis and the second column is the vertical "j" axis. This is the commonly used documentation for image axes. Establishing image axes was important because manually marked nuclei coordinates corresponding to pixels in each image was documented in a csv file by Phoulady and Mouton [1]. Initial exploration of these images and their corresponding manual nuclei coordinates was conducted to confirm initial alignment.

Binary nuclei masks were created for each full slide image, in both the test and train sets, to determine if an image patch would be labelled as positive or negative. The distance measure used to determine if an image patch was positive, stated as "within 15 pixels of a manually marked nucleus point" [1] or negative, outside 15 pixels, was not mentioned in the paper. Possible distance measure options include the L1/L2 norm, Hamming or Euclidean. I chose to use the Euclidean distance because it is a commonly used distance measure in image processing. For each pixel in each full slide image, the Euclidean distance was calculated to every manually marked nuclei in the image. If any of the calculated Euclidean distances for a single pixel were within 15 pixels or less from any manually marked nuclei, the pixel value was changed to 1. If the pixel was not within 15 pixels of any manually marked nuclei, the pixel value was changed to 0. An example of a generated nuclei mask for a single full slide image, following this distance measure can be seen in Figure 1.

Each image patch in the training set could have been constructed from overlapping or non-overlapping 15-pixel intervals. I chose the overlapping option as it seemed like the most likely approach given the large number of image patches reported by the authors. Image patches, of size $75x75$ pixels, were created by overlapping 15 pixel intervals across the full slide image (to the right), and then down 15 pixels and then across again. Image patches that were not $75x75$ pixels were removed. 4860 image patches were created for each full slide image in the training set.

Two different methods could have been used to label each image patch as positive or negative following the description in Figure 3. Distance could be calculated from the centre of each patch or from any pixel in the patch. Originally I chose to calculate the distance from any pixel in the patch. In other words, if a single pixel in the patch had a pixel value of 1 in the corresponding nuclei mask, the patch was labelled as +1. If all pixels had a value of 0 in the nuclei mask, therefore no nuclei were in the patch, the patch was labelled as -1. This meant that one of the pixels in the patch was within 15 pixels of a manually marked nuclei. This method achieved 20.43% of positively labelled patches which was more than the expected number of 1.71% of patches labeled as positive as described in the paper [1]. I decided to try the other method, only labelling a patch based on the centre pixel value located at coordinate (37,73). In other words, the patch was only labelled as positive if the center of the patch was within 15 pixels of the manually marked nuclei coordinate. Image patches were extracted from the corresponding binary nuclei mask at the same time. If the pixel coordinate at (37,37), from the image patch, had a corresponding value of 1 in the binary nuclei mask patch, the cytology image patch was labelled 1. If the pixel coordinate at (37,37), from the image patch, had a corresponding value of 0 in the binary nuclei mask patch, the cytology image patch was labelled -1. Image patches generated using the described method can be seen in Figure 2.

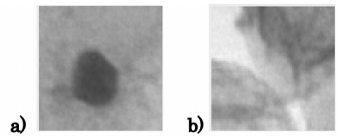

Figure 2: Sample cytology image patches. a) A patch labelled as +1 (a nuclei). b) A patch labelled as -1 (not a nuclei).

Image patches of size 75x75 pixels were extracted from the grayscale frames in the training dataset. Patches were sampled uniformly at 15-pixel intervals and each of them was labeled as positive if it was within 15 pixels of a manually marked point; otherwise, it was labeled as negative. The network was trained with a constant learning rate of 0.001 in 100 epochs. Overall, 654,948 were extracted and used to train the network with only 1.71% of patches labeled as positive.

The trained CNN was used to classify patches extracted from the frames in the test set. The patches were extracted at shorter interval compared to the training dataset, namely at 3 pixels apart. The hit map was generated, dilated, and thresholded at cut off point of 0.5. Small regions that did not represent the nuclei (regions smaller than 100 pixels) were removed.

Figure 3: The image pre-processing description from the paper [1]

A similar pre-processing method was be applied to the full slide images in the testing set. The method in the paper used 3 pixel intervals instead of 15 pixel intervals to generate the testing patches [1]. However, this method created millions of testing images which caused the computer to crash multiple times. Therefore, instead of using 3 pixel intervals, 15 pixel intervals were used to generate the testing patches following the same pre-processing method as the training image patches.

All image patches were created in MATLAB, saved, and then loaded into a Jupyter notebook in Google colab. Results were assessed visually by looking at the images, and quantitatively by determining if about 654,948 image patches were extracted in the dataset and if about 1.71% of patches were labeled as positive.

3.3 Model description

A deep CNN architecture was implemented following the description in Figure 4. From the description in Figure 4, the most likely model architecture was constructed and can be seen in Figure 5. The model was not pre-trained. The input to the model was grayscale images of size $75x75$, with corresponding +1 or -1 labels which were one hot encoded. The output of the model was a predicted label for each patch. There was no mention of a validation set in the paper, therefore only training and testing sets are used to train and evaluate the model.

The first 2D convolution used the Keras layer "Conv2D" with 32 filters, $3x3$ kernel size, strides of 1, no padding and a ReLu activation function. 2D layers were used because the input are 2D grayscale images with only 1 channel. There was no indication in the paper if non-linear activation functions, such as the ReLU, are specified at each 2D convolution layer. It is common that each convolution layer has an activation function, usually ReLU since it is the best and most commonly used activation function. An activation function was necessary to make the model nonlinear because the classification is a nonlinear task. Therefore, I chose to use the ReLu activation function at each convolution layer.

The first max pooling layer used the Keras layer "MaxPool2D" with a pool size of $2x2$, strides of 2 and no padding. The second convolution layer used 64 filters, $3x3$ kernel size, strides of 1, no padding and a ReLu activation function. Then the second max pooling layer used a pool size of $2x2$, strides of 2 and no padding. The tensor was then flattened to a 1D tensor to be used in the fully connected neural network. The first fully connected layer had 128 nodes and used the ReLu activation function. The second fully connected layer had 64 nodes and again used the ReLu activation function. The output fully connected layer had 2 output nodes, because there are 2 classes, with a softmax activation function. I assumed that the softmax loss layer applied to the output layer described in the paper 4 meant a softmax activation function on the output layer and a crossentropy loss function for the model.

The exact number of nodes in each layer and other parameters in the convolution, max pooling, and fully connected layers were unknown as no code was provided and they were not explained in the paper. Therefore, I used the most common parameter values at each layer. It is common practice in deep convolution neural networks to use filter sizes of 16, 32, 64, 128 etc. and to increase the number of filters as the network gets deeper in order to capture more high

We trained and tested a Convolutional Neural Network (CNN) to detect cervical nuclei in the dataset. The designed CNN architecture has two convolutional layers (CL) and two fully connected layers (FCL). Each CL was followed by max pooling layers (MPL) and each FCL was followed by ReLu activation function (RL). Finally, the output layer (OL) with two neurons was followed by a soft max loss layer (SFL) to generate the probabilities of each patch being a nucleus or not.

Figure 4: The CNN architecture description from the paper [1]

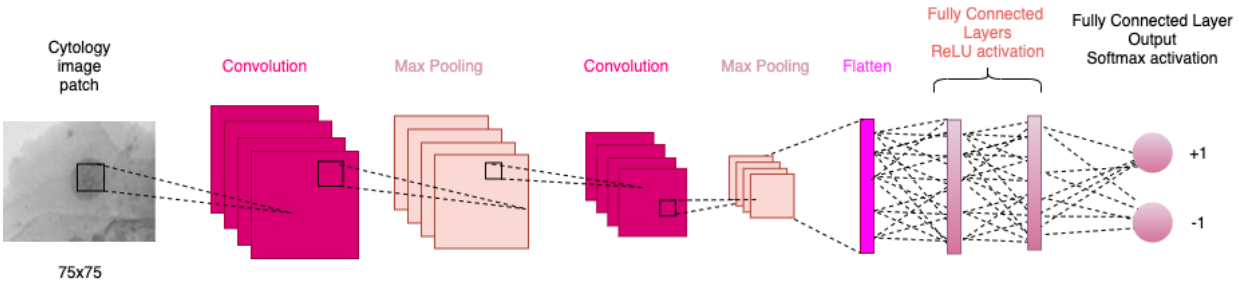

Figure 5: Proposed model

level patterns from the extracted features. Therefore, I chose filter sizes of 32 and then 64 as the images are small and I wanted to extract a large number of features. Small kernel sizes of $3x3$ were used because the nuclei in the cytology images are small features. The images are small so strides of 1 were used to not skip any pixels. Max pooling parameters with pools size of $2x2$ and stride 2 were used as these are common parameters and the images are already small so they do not need to be shrunk down in size. It is common to decrease the number of hidden nodes gradually between the flattened layer and the final output layer, therefore 128 and then 64 nodes were chosen at each fully connected layer respectively.

The model was trained using a learning rate of 0.001 as described in the paper as seen in Figure 3. The paper did not explain the optimizer used to train the model. I chose to use the Adam optimizer as it is the most common and generally considered the best optimizer. It is likely that Phoulady and Mouton used this optimizer in their approach. I assumed that the softmax loss layer applied to the output layer described in the paper 4 meant a cross-entropy loss function was used to train the model. Because the output of the model is two classes, I chose to use the categorical crossentropy loss function in the Keras library.

Phoulady and Mouton used 100 epochs to train the model as described in their paper as seen in Figure 3. After experimentation, it was found that 5 epochs was enough to train the model with a batch size of 128. This batch size was used because there are a large number of training images and a larger batch size can speed up the training time of the model.

3.4 Post-Processing

The paper by Phoulady and Mouton did not describe the post-processing method. The generation of the hit map described in the paper was unclear. The calculation of the number of missed nuclei by the model was also not described in the paper. The following is a description of how I generated the hit map and calculated the number of total missed nuclei by the model. The output of the deep learning model, a CSV file containing the predicted label of each image patch in the testing set, was saved and processed in a MATLAB script. A hit map image was created for each cytology whole slide testing image. The hit map image had predicted nuclei markers, from the output of the deep learning model, and ground truth nuclei coordinates. The predicted nuclei markers for each patch was generated by reversing the method described in section 3.2. The center of each image patch was coordinate (37,37) and patch labels were created based on this center coordinate. Each $75x75$ pixel patch in each whole slide image was generated. If the center of the patch had a predicted label of +1, the corresponding coordinate in the whole slide image was labelled as a hit and added to an array. Arrays of all of the predicted hit coordinates were saved and then superimposed onto a whole slide image along with the ground truth nuclei coordinates. This was repeated for each whole slide image in the testing set. The number of missed nuclei was generated by manually counting the number of ground truth nuclei that were not hit by any predicted

hit coordinates in each image. The percentage of nuclei missed by the machine learning model was calculated using a MATLAB script and involved determining the total number of manually marked nuclei in the test set.

3.5 Experimental setup and code

All image pre-processing and post-processing code was written in MATLAB because it is a user-friendly environment for image processing. Environment and GPU/CPU that the paper used to train the deep learning model was not mentioned in the paper. I chose to train the models in Jupyter notebook using Google Colab environment. A local Jupyter notebook was set up and then connected to Google Colab by selecting the local runtime option in Colab. Keras and Tensorflow libraries were used to build the convolution neural network model. The sklearn library was used to display statistical results of the test dataset such as precision, recall and F measure. The implemented code of the image pre-processing, model and post-processing as well as detailed instructions on how to set up the environment can be found here: `https://github.com/Brittaney-Everitt/Deep_Learning_Final`

3.6 Additional experiments

The model was additionally trained using 15 epochs and 35 epochs. Results for each experiment can be seen in section 4.2.1.

The hit map whole slide testing images that were generated following the method described in section 3.4 were manually analyzed to determine where the model was not detecting nuclei. These results are presented in section 4.2.2.

3.7 Computational requirements

MATLAB version R2020b Update 2 (or a more recent version) is needed to run the image pre-processing and post-processing. Python 3 is required to run the code. A MacBook Pro with macOS Catalina, version 10.15.6, and a 2.3 GHz 8-Core Intel Core i9 processor (CPU) was used to implement this project. Additionally, the computational time to create a single whole slide binary nuclei mask was about 10 minutes. The creation of all image patches (4860 patches) for one whole slide image was about 2 minutes. The computational time of each epoch during training was about 12 minutes. The image patches in the train set require 1.37 GB of space. The image patches in the test set require 477.8 MB of space.

4 Results

4.1 Results reproducing original paper

This section is a concise presentation of the results of the reproducibility study when the model was trained using 5 epochs.

4.1.1 Result 1 - Model statistics

This result references claim 1 in section 2. These results were reported when the loss after the fifth epoch was 0.0266 and began to plateau. Results were calculated using the sklearn *classification report* method. Average results were calculated using the *macro* option of the *precision score, recall score and f1 score* methods in sklearn which calculates the unweighted mean of each label for each metric. The trained model had an average precision of 0.88, recall of 0.82 and F1-score of 0.85 on the testing dataset. The precision, recall and F1-score for the -1 class was either 0.99 or 0.100 for the testing dataset. The positively labelled patch class, +1, had lower results. The precision was 0.76, the recall was 0.65 and the F1-score was 0.70.

4.1.2 Result 2 - Image patches

This result references claim 2 in section 2. The number of generated image patches for the training set, following the method described in section 3.2 was 335,340. 5906 image patches in the training set were labelled as +1 which resulted in about 1.76% positive labels (within 15 pixels of a manually marked point).

4.1.3 Result 3 - Model comparison

This result references claim 3 in section 2. The baseline segmentation developed by Phoulady and Mouton achieved the following results on the test set: precision of 0.805, recall of 0.838 and F1-score of 0.82. If the average results

Table 1: Model results after training for 5, 15 and 35 epochs

Number of Epochs	*Training Set*			*Testing Set*		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
5	0.88	0.84	0.86	0.88	0.82	0.85
15	0.88	0.84	0.86	0.86	0.83	0.84
35	0.95	0.95	0.95	0.86	0.83	0.84

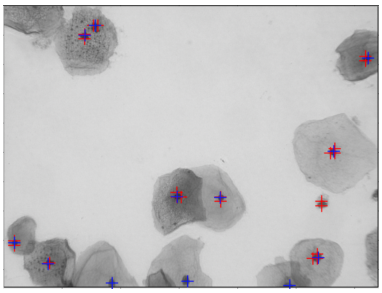

Figure 6: A hit map for one whole slide image in the test set. Blue represents ground truth nuclei and red represents nuclei patches that were predicted as positive from the output of the CNN.

from the implemented CNN model in this study are used to compare to the baseline segmentation method, than the model achieved an average precision of 0.88, recall of 0.82 and F1-score of 0.85 on the test set (as shown in a previous section). The average precision result, from the model implemented in this study, was more than a 6% improvement over the precision reported in the baseline segmentation by Phoulady and Mouton. However, the recall was about the same for both methods. The F1-score of the CNN method implemented in this study was about 3% better than the baseline method which is lower than the claim made by Phoulady and Mouton that the "F measure was more than 5% better than the baseline method" [1].

4.1.4 Result 4 - Unidentified nuclei

This result references claim 4 in section 2. The paper did not explain how this result was calculated. This result was calculated using the method explained in section 3.4. After training the model using 5 epochs, the model missed a total of 76 nuclei in the test set, resulting in about 10.51% missed nuclei. After manually inspecting the hit maps generated in the post-processing step described in section 3.4, most missed nuclei were close to the boundary of the whole slide image. Some overlapping nuclei were missed but this was difficult to determine manually as the "hit" coordinates and the ground truth nuclei coordinates were very close together.

4.2 Results beyond original paper

4.2.1 Additional Result 1 - Model Exploration

The model trained much faster than expected. After only about 5 epochs compared, loss began to plateau around 0.02. However, the paper reported that the model ran for 100 epochs. To test this claim, the model was run for a different number of epochs using the same parameters as explained in section 3.3 for each experiment. Results can be seen in Table 1.

4.2.2 Additional Result 2 - Visual Detection

Visual results of the nuclei detection from the CNN were generated in the post-processing step explained in section 3.4. It can be seen in Figure 6 that the three nuclei (blue targets) along the bottom of the whole slide image had no "hits" from the CNN. In Figure 6 there are 10 manually counted "hit" nuclei and 13 total nuclei in the whole slide image.

5 Discussion

Overall, most implementation results were within range of the results from the paper by Phoulady and Mouton. However, no experiments outperformed the results in the paper.

In reference to claim 1, Phoulady and Mouton reported that the precision of the method was 0.861, the recall was 0.895 and the F1-score was 0.878 for the testing set [1]. It was unclear from the paper how the results were calculated. If the results reported in the paper by Phoulady and Mouton [1] used an average of each metric, then the model implemented in this project reproduced similar results to the paper as seen in section 4.1.1. If results were reported only on the +1 class, which is the nuclei detection class, then the model implemented in this project did not reproduce similar results to the paper. A possible reason for the difference in results is that the paper generated image patches in the test set using 3 pixel intervals, resulting in more test image patches and more precise image patches that could potentially better localize nuclei in the center of the image patches. This could have helped increase the number of patches in the +1 class. In this study, testing patches were created using 15 pixel intervals as explained in section 3.2 resulting in fewer total image patches in the test set. As seen in section 4.1.1, the results between classes in the testing dataset were highly variable. This could be because of the imbalance in the number of images in each class. This imbalance is evident in both the training and testing sets. In the testing set, there were 114504 image patches in the -1 class and only 2136 image patches in the +1 class. Because there was no validation dataset, analysis between the training and testing results was conducted to determine if the model was overfitting or underfitting. The training dataset had slightly better results than the testing dataset for the +1 class with precision, recall and F1-score of 0.77, 0.69 and 0.73 respectively. Based on the results, there was no overfitting in the model.

In reference to claim 2, Phoulady and Mouton reported 654,948 extracted image patches in the training set with 1.71% positive labels [1]. The number of total image patches generated, as seen in section 4.1.2 was about half the number of patches reported in the paper by Phoulady and Mouton. This discrepancy was included as a question to the authors in section 5.3. However, the ratio of positively labelled patches was nearly identical, with a 0.05% difference in the results. Overall, claim 3 made by the paper was not reproducible. This is because the reported percent increase in which the CNN outperformed the baseline segmentation method (implemented by Phoulady and Mouton) was not met for the majority of the metrics. A possible reason for this could be the test set was smaller in this reproducibility study than used in the study for the baseline segmentation. Claim 4 was reproducible because the implementation results were within 0.51% of the results reported in the paper. Phoulady and Mouton reported that their "CNN method only missed about 10% of the nuclei" and that most missed nuclei were "close to the boundary of the whole slide image or were overlapping nuclei" [1]. These claims were consistent with the results found in the implemented method.

In reference to the additional experiments conducted, as seen in Table 1, the testing and training results do not change much between 5 and 15 epochs. After 35 epochs, the model began to overfit, as seen by comparing the training and testing results in Table 1. If the model was trained for 100 epochs, it would be very likely that the model would overfit. From section 4.2.2, after analyzing many test images, some patches were predicted to have a +1 label when the ground truth label of the patch was actually -1. This can be seen in Figure 6 where two predicted patches (red targets) are labelled as positive and visually correspond to an artifact in the image that is not a labelled nuclei (blue target). The classification usually corresponded with artifacts in the images or patches that resembled nuclei but were not actually nuclei. This result was determined by manually analyzing all of the test images and is also shown in the precision, recall and F1-score results.

Based on the machine learning reproducibility checklist, the main components missing from this paper that are needed to make the paper more reproducible are: a clearer description of the model; link to downloadable source code for the deep learning model; clearer description of the pre-processing step along with source code; the hyper-parameters used in the configuration of the model and results; a description of how and where experiments were run; and a clear description of how the results were calculated [3]. However, a link to a downloadable version of the dataset along with an explanation of how the samples were split for training and testing were present in the paper. [3].

5.1 What was easy

Loading the data and separating the data into training and testing sets as the authors did in the paper was relatively easy to do in MATLAB. Setting up and running the deep learning model using Keras and Tensorflow was relatively easy. Post-processing of the images in MATLAB was straightforward to implement. However, there was no description of how to calculate and generate the hit map in the paper, so the post-processing method implemented in this report might not match the method in the paper by Phoulady and Mouton. Calculating the result metrics of the model and experiments was easy because the sklearn python library was used. However, how the metrics were computed was not described in the paper. Therefore this calculation method might not match the method used by the authors.

5.2 What was difficult

Determining the exact technique of how the authors pre-processed the cervical cytology smears into image patches was challenging because the description in the paper was insufficient. Important details such as if the patches were created

using overlapping or non-overlapping pixels and the distance measure (L1/L2 norm, Euclidean distance or Hamming) and method (any pixels vs the center pixel) used to determine the label of a patch based on its distance from a manually marked pixel were missing. It was not reported in the paper if the image patches were augmented in any way which could explain the number the increased number of training patches reported in the paper. Additionally, determining if the nuclei coordinates were in the x,y coordinate system or the i,j coordinate system was difficult and there is no documentation about this in the paper or the Github dataset. As well, formatting the image patches and the labels was difficult. Originally, the files were sorted out of order and caused the model to classify all patches as -1. Files were renamed with leading 0s and sorted based on filename to ensure correct patch label correlation.

Furthermore, determining the exact model architecture and hyper-parameters used in the CNN model developed by Phoulady and Mouton was quite challenging because the hyper-parameters and details such as the activation functions used in these layers were missing in the paper. Some unknown hyper-parameters in the CNN architecture include number of nodes in each layer, batch size, kernel size, padding etc.

5.3 Communication with original authors

The authors were emailed one time and were asked three questions regarding: how they generated almost double the number of image patches in the training set, if they accounted for the unbalanced dataset during training, and if any uncommon hyperparameters were used in the model. To date, there was no reply from the authors.

References

[1] Phoulady HA, Mouton PR: A new cervical cytology dataset for nucleus detection and image classification (cervix93) and methods for cervical nucleus detection. *arXiv preprint arXiv:181109651* 2018

[2] Rahaman MM, Li C, Wu X, Yao Y, Hu Z, Jiang T, Li X, Qi S: A survey for cervical cytopathology image analysis using deep learning. *IEEE Access* 8:61687–61710, 2020

[3] Pineau J, Vincent-Lamarre P, Sinha K, Larivière V, Beygelzimer A, d'Alché Buc F, Fox E, Larochelle H: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:200312206* 2020