Springboard Data Science Intensive Program

Capstone Project 1:

Cervical Cancer Screening Using Cervix Images Dataset

By: Ivy Huong Nguyen, Ph.D.

September 2018

Table of Contents

I. INTRODUCTION

Cervical cancer is classified as an easy-to-prevent cancer if caught in its precancerous stage. According to the American Cancer Society, cervical cancer is one of the most successfully treatable cancers if detected early. However, one of the most problematic issue in treating patients with this type of cancer is the ability to identify an appropriate treatment that works effectively and accordingly to the patient's physiological needs. In rural parts of the world, many women who are susceptible to cervical cancer are receiving treatment that will not work due to the position of their cervix. In order to solve this problem, it would be necessary to develop an algorithm that could help health providers identify the type of cervix that a patient has based on the images of their cervix. In this capstone project, we will demonstrate the details of our process that we used in wrangling and analyzing the available cervix images dataset, and finally using the training dataset to conduct a model selection.

This capstone project is inspired by a Kaggle competition posted and hosted by Intel and MobileODT. The main goal of this project is to develop a new algorithm that can effectively identify the type of cervix a patient has based on images. This workflow will help health providers give proper cervical cancer treatment referral to their patients

The client for this problem is health providers as well as the patients that are at high risk for cervical cancers. The health providers can use this algorithm to have an effective real-time determination to provide an appropriate cervical cancer treatment for their patients. That way, patients don't have to face high-cost treatment as well as ineffective treatments. This will help both health providers and patients reduce their time in giving and receiving cervical cancer treatment.

II. DATA COLLECTION AND WRANGLING SUMMARY

The data that is used in this project is provided by Intel and MobileODT via their posted Kaggle competition. The data wrangling step was initially started as follows:

Step 1: Generate a pandas dataframe that includes the image directory paths and the type of cervix image for the training dataset

Step 2: Visualize some sample images of each cervix type of the training dataset to get a sense how different the types are

Step 3: Create a new dataframe that include the following columns for the training dataset:

- Image directory
- Cervix type
- Image height
- Image width
- Image channels

Step 4: Write the dataframe generated in step 3 to a csv file (training_images_size.csv)

Step 5: Create a dataframe that includes the following columns for the testing dataset:

- Image directory
- Image height

- Image width
- Image channels

Step 5: Convert the generated dataframe in step 5 to a csv file for later use (testing_images_size.csv)

III. EXPLORATORY DATA ANALYSIS:

The exploratory data analysis was performed under different categories as follows:

a. *Exploratory data analysis based on image sizes*

Under the first category, I found out that there are 1481 images with 3 different cervix types in the training dataset and 503 images in the testing dataset. Additionally, I also noticed that all testing and training images have really high resolution and thus should be resized before using them for any further analysis or constructing an algorithm. The image size was finally reduced to 224x224x3 for use in the model selection process. This size of image was chosen to accommodate the default input size of the VGG16 model and the ResNet50 model available in the Keras applications.

b. *Exploratory data analysis based on the Exchangeable Image File Format (EXIF data) of all images*

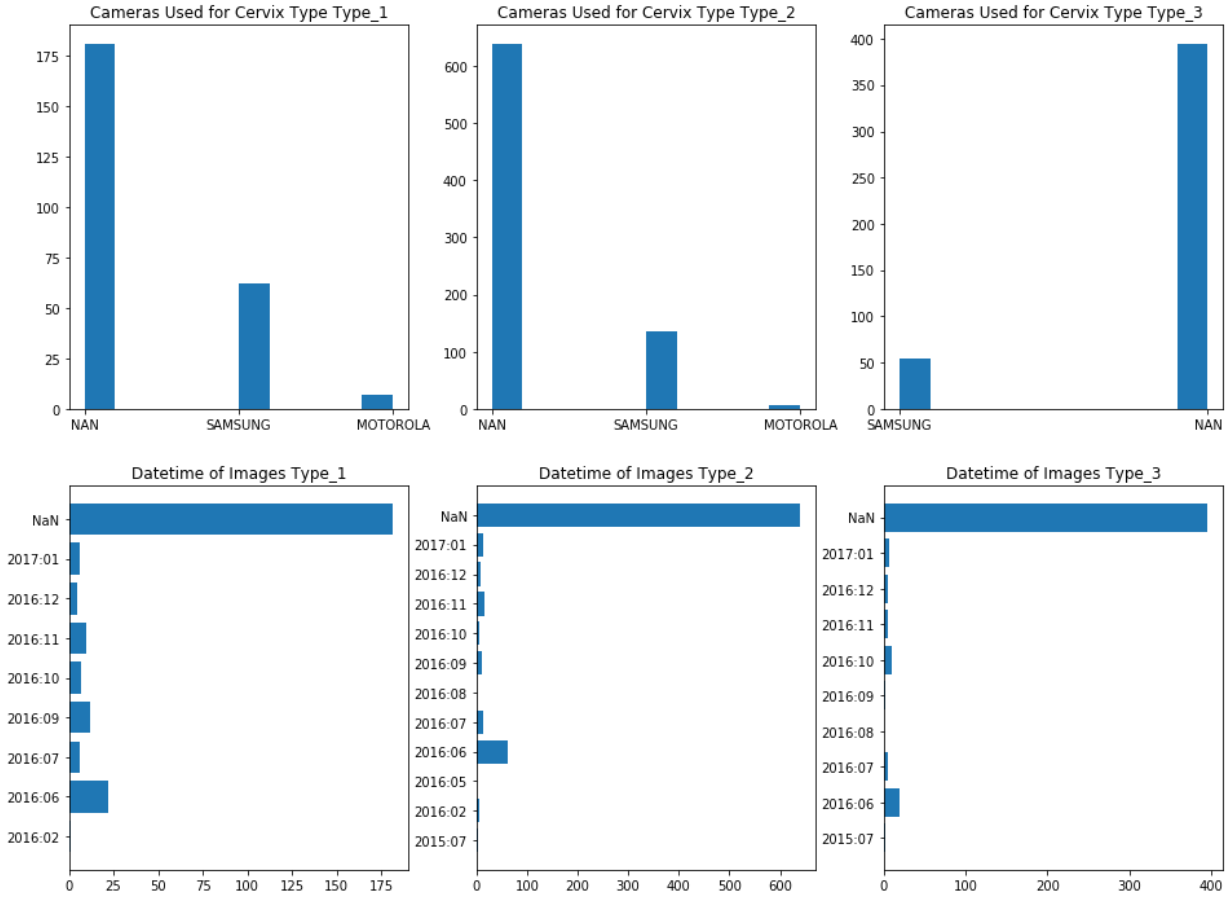


Figure 1. EDA using available EXIF data to explore the camera models that were used to take the images and the datetime that the images were taken.

The Exchangeable Image File Format (EXIF) is a way to store data related to an image such as the date and time that the image was taken, camera model, camera settings such as ISO speed, shutter speed, aperture, white balance, etc.….In this category of the EDA process, I chose to explore two components of the EXIF data that are available to certain images: datetime and camera model. I noticed that Samsung and Motorola are the two main models used to take most of the images in both the training and the testing dataset. There is also a large portion of images that have 'NaN' as their camera model, which is an interesting fact for images that were taken for medical purposes. If the 'NaN' data portion is disregarded, June of 2016 is the time that most images were taken.

c. Inferential statistics to determine if there is any image resolution difference between different cameras.

Since there are at least two different camera models that were used to take most of the images, I wanted to test the hypothesis whether there is any image resolution difference between Samsung and Motorola. The inferential statistics was conducted using two different methods: frequentist approach and bootstrap hypothesis testing. The null hypothesis for this inferential statistics is that there is no difference in the resolution mean between the images taken by Samsung and those taken by Motorola. The alternative hypothesis claims that that there is a significant difference in the resolution mean between the images taken Samsung and those taken by Motorola. I decided to conduct the analysis with 95% confidence or a significant probability α of 0.05.

In the frequentist approach, I chose to conduct the statistical analysis using a two-tailed t test because of the sample size of the Motorola sub-dataset (<30). Under this approach, I found a statistical t value of -1.106 that has the corresponding p-value of 0.27. This p-value is much greater than $\alpha = 0.05$ and thus I failed to reject the null hypothesis.

In the bootstrap hypothesis testing approach, I computed 10,000 bootstrap replicates after shifting the resolution mean of both the Samsung-subdataset and the Motorola-subdataset to be the same. This approach also gave a p-value that is greater than 0.05 and thus the null hypothesis is not rejected.

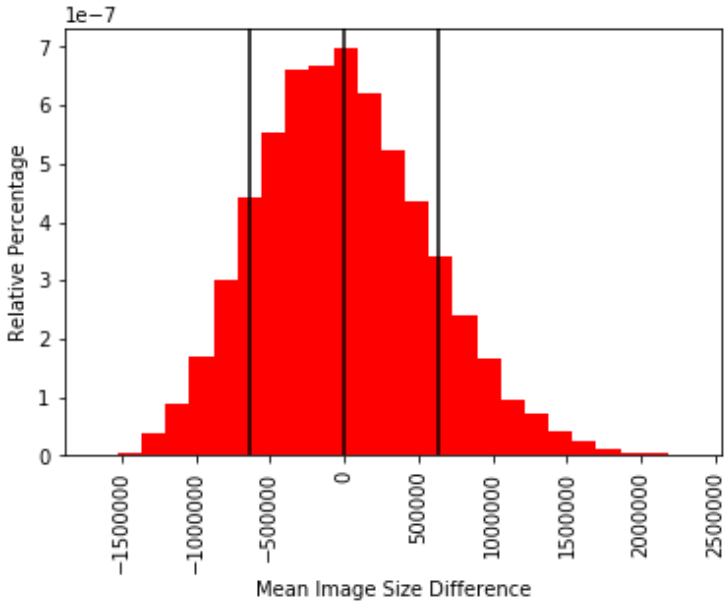


Figure 2. Bootstrap replicates probability distribution where the three black lines represents the mean difference in image sizes (middle), upper bound (right), and lower bound (left).

By using two different inferential statistical approaches, I was able to confirm that there is no difference in the image resolution taken by Samsung and Motorola, which should be the case in medical research practice.

d. Descriptive Statistics for both the Training and the Testing dataset

Descriptive statistical analysis was performed using grayscale images to find the average image of each cervix type in the training dataset and the average image of the testing dataset. The min and max images were also determined by using vector norms as a criteria for both dataset.

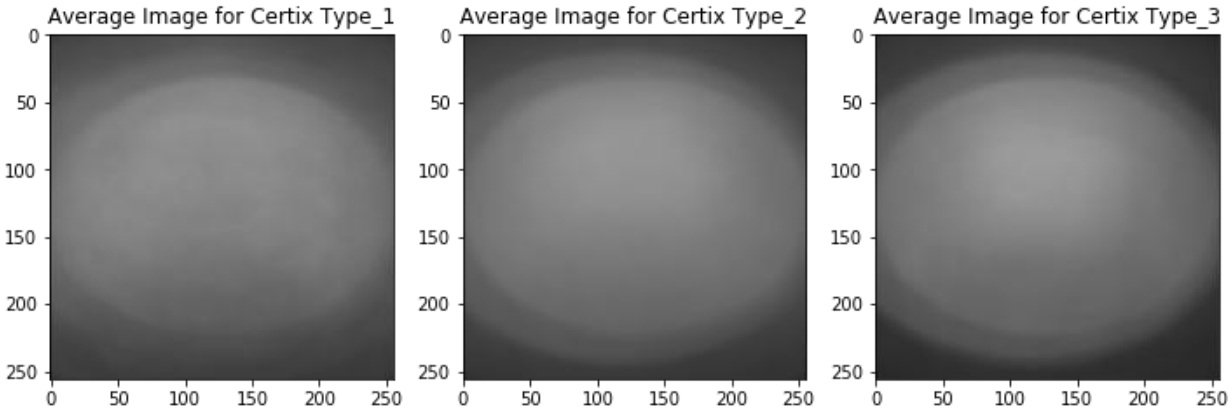


Figure 3. The average images for each cervix type of the training dataset.

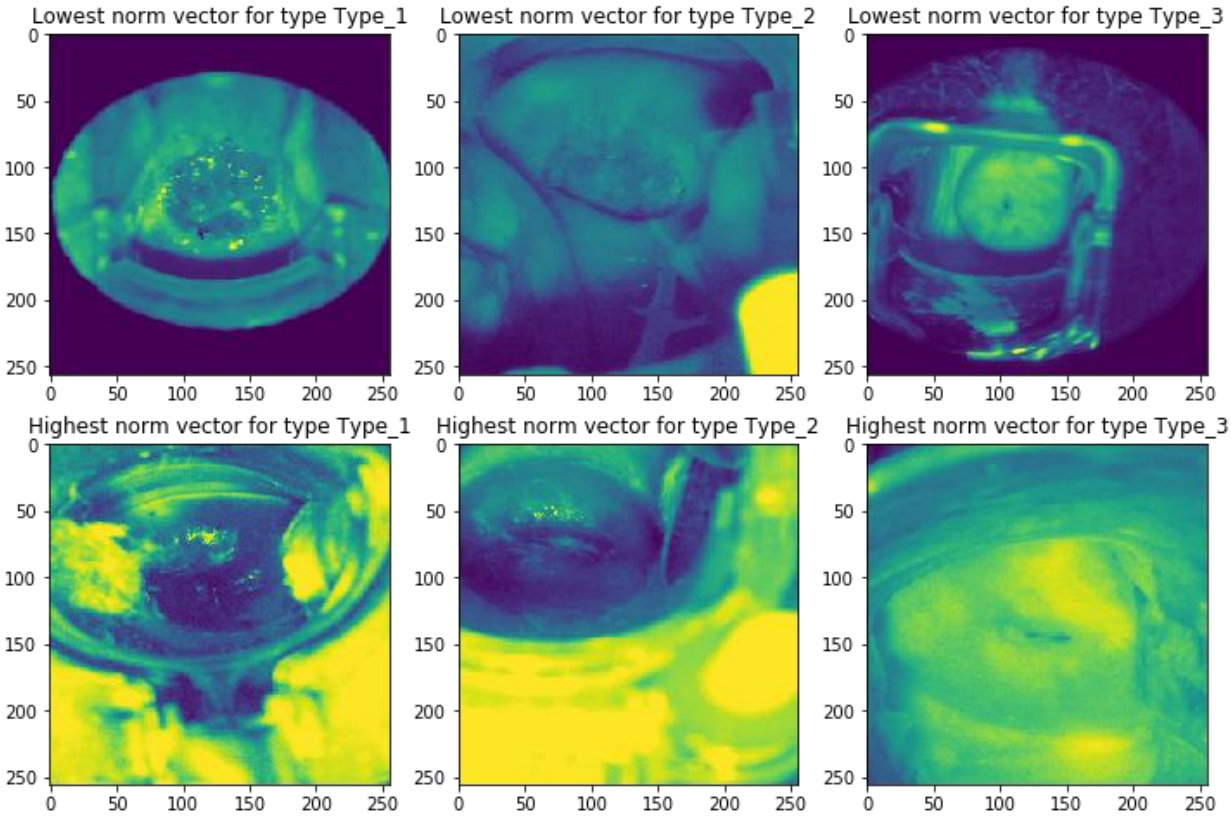


Figure 4. The corresponding min and max images based on vector norms for each cervix type of the training dataset.

It appears that the average image of cervix type 1 has the lightest contrast in comparison to the average image of the other types based on grayscale images. This fact is true for both the training and the testing dataset.

e. <u>Explore the Structural Similarity Index (SSIM) of a cervix type.</u>

In order to compare whether two different images have any similarity in their structures, I useed SSIM as an evaluation. Structural similarity index (SSIM) is calculated based on the following formula:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where:

- μ_x is the average of x
- μ_y is the average of y
- σ_x^2 is the variance of x
- σ_y^2 is the variance of y
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are the two variables to stabilize the division with weak denominator
- L is the dynamic range of the pixel-values

- $k_1 = 0.01, k_2 = 0.03$ by default

SSIM keeps track of the structural information of the images and thus can be used to compare the similarity in structures of a given set of images data. SSIM value can vary between -1 and 1, where 1 indicates perfect similarity. SSIM can be calculated via implementing the scikit-image package of Python. In this section, only the SSIM of cervix type 1 from the training dataset was determined due to the time limit.

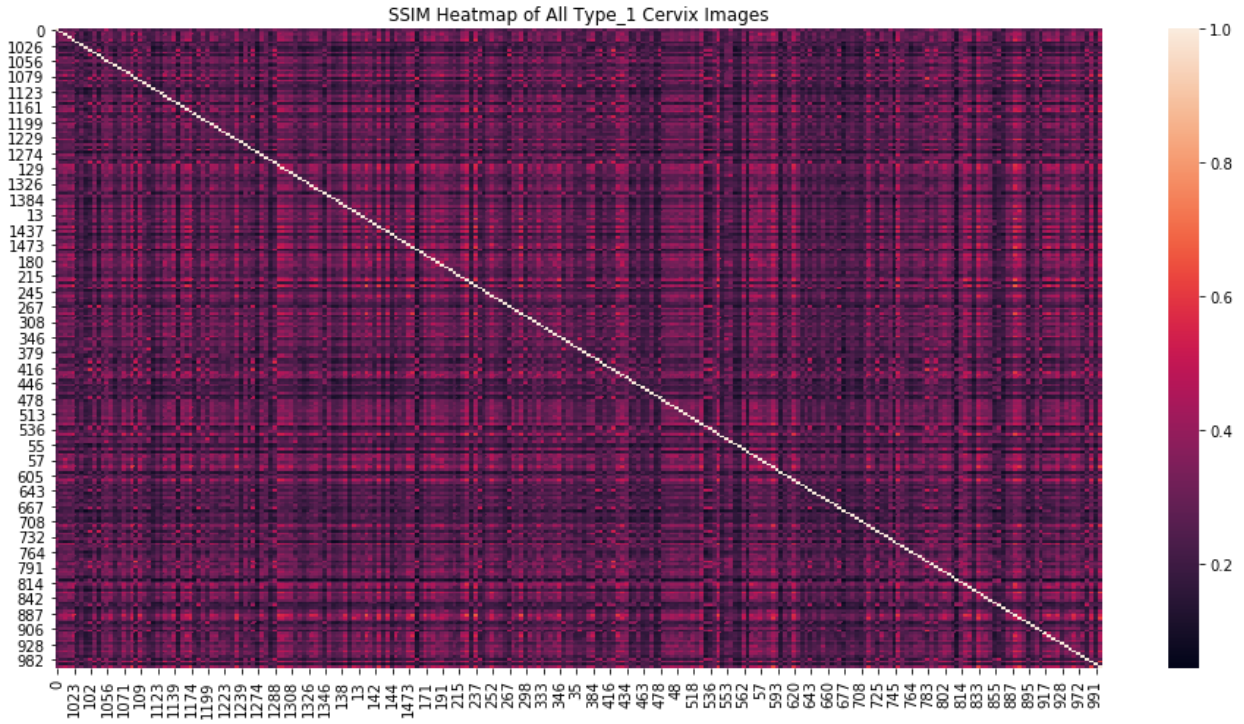


Figure 5. SSIM of type 1 cervix images where the lighter the color is, the more similar the images are to each together.

IV. MODEL SELECTION

The model selection process comprises of 2 stages: a) comparing the performance of three different models with different complexity in the number of layers, b) optimizing the best model by adding layers, changing activation methods, changing different combination of optimizer and loss function, and finally changing the number of filters.

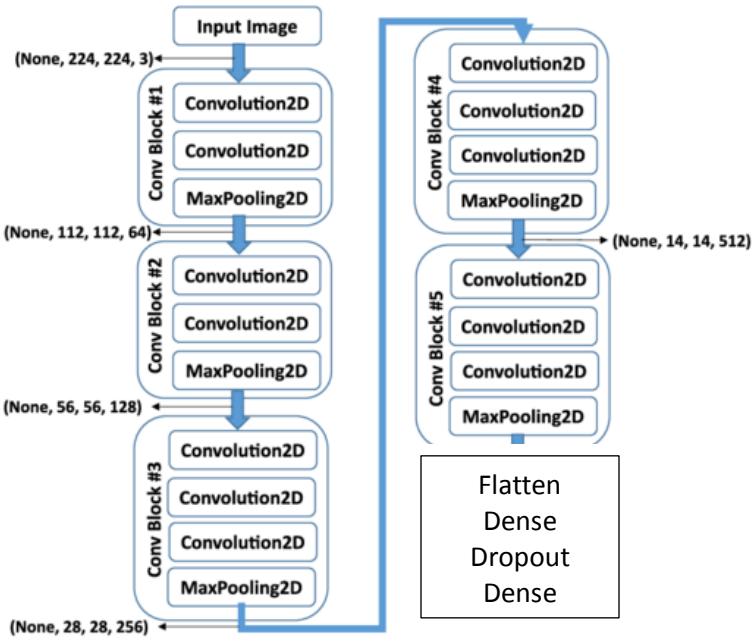
a) Comparing Model Performance

In order to determine whether using a more complex model would be a better choice for my dataset, I chose to compare the performance of three different models including:

1. A simple convolutional neural network model with 3 blocks of layers including a convolutional layer followed by an activation, a maxpooling2D layer and a dropout layer. This model is finished up with a block of flatten layer and dense layer. This model will be preferred to as Model 1 throughout this report. The architecture of this model is summarized as follows:

Layer (type)	Input Shape	Output Shape
Block1_Conv2D (Conv2D)	(None, 224, 224, 3)	(None, 222, 222, 32)
Block1_Activation ('relu')	(None, 222, 222, 32)	(None, 222, 222, 32)
Block1_MaxPooling2D	(None, 222, 222, 32)	(None, 111, 111, 32)
Block1_Dropout	(None, 111, 111, 32)	(None, 111, 111, 32)
Block2_Conv2D (Conv2D)	(None, 111, 111, 32)	(None, 109, 109, 64)
Block2_Activation ('relu')	(None, 109, 109, 64)	(None, 109, 109, 64)
Block2_MaxPooling2D	(None, 109, 109, 64)	(None, 54, 54, 64)
Block2_Dropout	(None, 54, 54, 64)	(None, 54, 54, 64)
Block3_Conv2D (Conv2D)	(None, 54, 54, 64)	(None, 52, 52, 128)
Block3_Activation ('relu')	(None, 52, 52, 128)	(None, 52, 52, 128)
Block3_MaxPooling2D	(None, 52, 52, 128)	(None, 26, 26, 128)
Block3_Dropout	(None, 26, 26, 128)	(None, 26, 26, 128)
Block4_Flatten	(None, 26, 26, 128)	(None, 86528)
Block4_Dense ('relu')	(None, 86528)	(None, 256)
Block4_Dropout	(None, 256)	(None, 256)
Block4_Dense ('softmax')	(None, 256)	(None, 3)

2. The second model is built from the trained VGG16 model where the weights are loaded from training the VGG16 model with ImageNet dataset. The added layers include the same layers of the last block of model 1. This model is preferred as model 2 throughout in this report. The architecture of this model is summarized as follows:



3. The third model is built from the trained ResNet50 model where the weights are loaded from training this model with ImageNet dataset. The added layers include the same layers of the last block of model 1. This model is preferred to as Model 3 in this report.

When comparing the performance of the three models, it appears that Model 2 and model 3 take much longer time to train each epoch than model 1 does. In terms of training and validation accuracy, model 2 and model 3 gave no difference or a slight improvement in accuracy in comparison to model 1. Based on their performance and the training time, I chose to use model 1 for the rest of my model optimization. A summary of the performance of the three models is listed below. All numeric data was extracted from epoch 1 of each model. In addition, I also noticed that overfitting became a prominent problem when the number of epochs is greater than 25 for model 1. Hence, the following optimization steps were set to have epochs=25 and batchsize=16 unless the number of filters is changed.

Table 1. Summary performance of three models based on epoch 1.

Model	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	Training Time/Epoch	Early_Stopping was used?
Model 1	0.5169	0.5118	1.0503	1.0171	206s	No
Model 2	0.5270	0.5118	7.4928	7.8691	1432s	Yes
Model 3	0.5211	0.5118	7.5800	7.8691	837s	Yes

b) Optimizing Model 1

i. Tuning the number of layers for model 1

In order to determine whether increasing the number of layers of model 1 would help improving its performance, I decided to optimize this aspect of model 1. I first defined a function that could help me add an X number of the additional block of layers including: 1 convolutional layer followed by an activation, 1 maxpooling2D layer, and a dropout layer. The result of this step is shown in Figure 6.

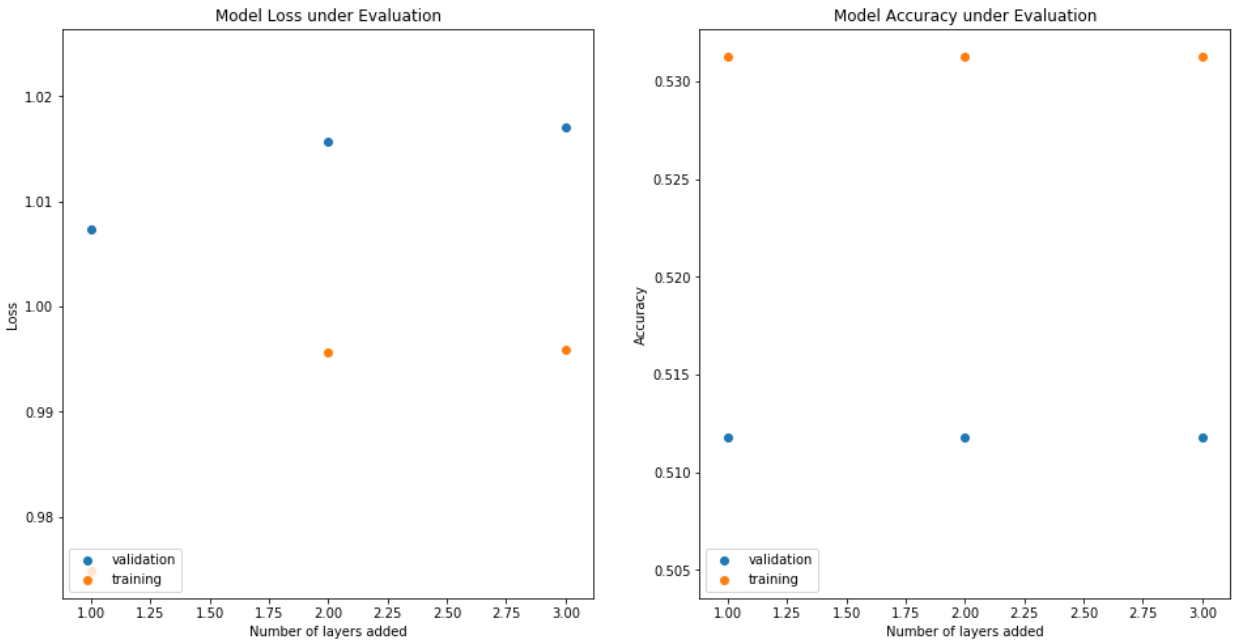


Figure 6. Model 1 performance as more layers are added

According to Figure 6, the accuracy of both the training and the validation dataset did not change as the number of layers increased. Therefore, increasing the number of layers of model 1 would not help improve its performance in this case.

ii. Tuning model 1 using a different activation method

Since increasing the number of layers of model 1 did not help its performance, I kept the original architecture of model 1 and modified different aspect of its structure. The second aspect of model 1 that I wanted to optimize was the activation method used in block 1, block 2, and block 3. I tested the performance of model 1 by changing the activation method in these blocks to the following: elu, softplus, sigmoid, and linear. The result of this optimization step is shown in Figure 7.

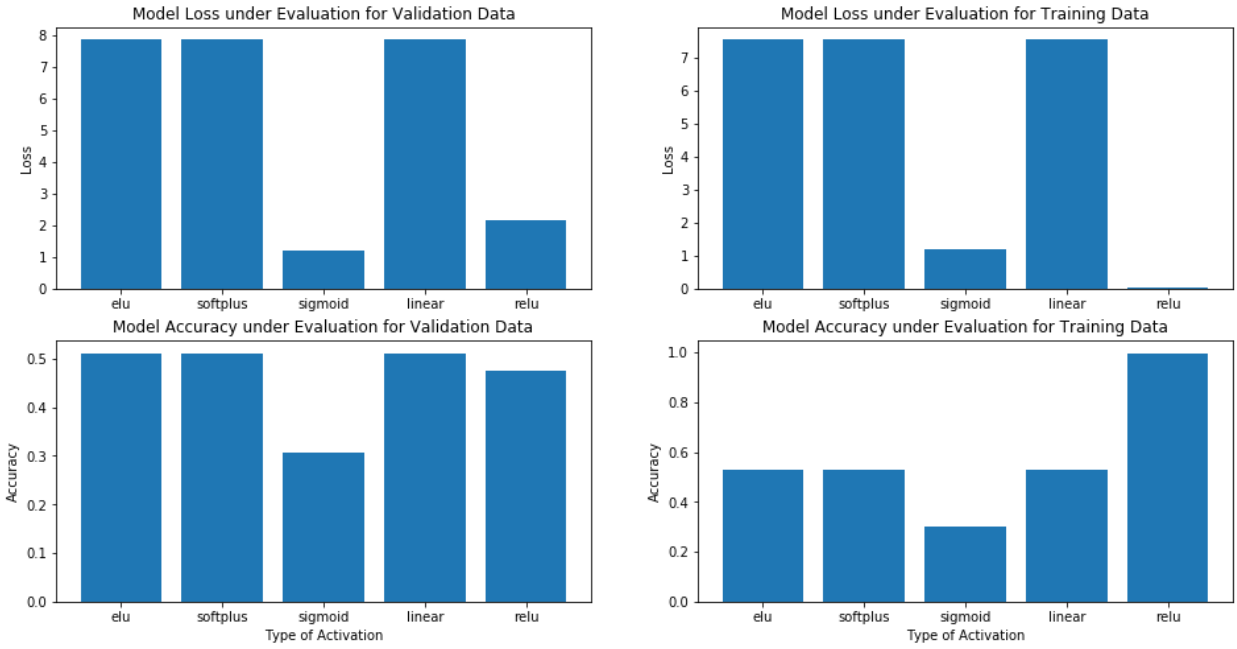


Figure 7. Model 1 loss and accuracy for both the training and the validation dataset under different activation method.

According to Figure 7, ‘relu’ appears to the best activation method out of the 5 tested activation methods. This fact is seen through ‘relu’ accuracy in both the training and the validation dataset. While ‘elu’, ‘softplus’ and ‘linear’ also give us very high accuracy in the validation dataset, their losses are much greater than ‘relu’, which make them not as ideal as ‘relu’ activation method.

iii. Testing the performance of model 1 by reducing the number of filters

The third aspect of model 1 that I tested is its number of filters. The number of filters represent the output shape which is also the input of the next layer. The number of filters can also be considered as feature detector. In this case, I chose to reduce the number of filters to determine whether that would help improving the performance of model 1.

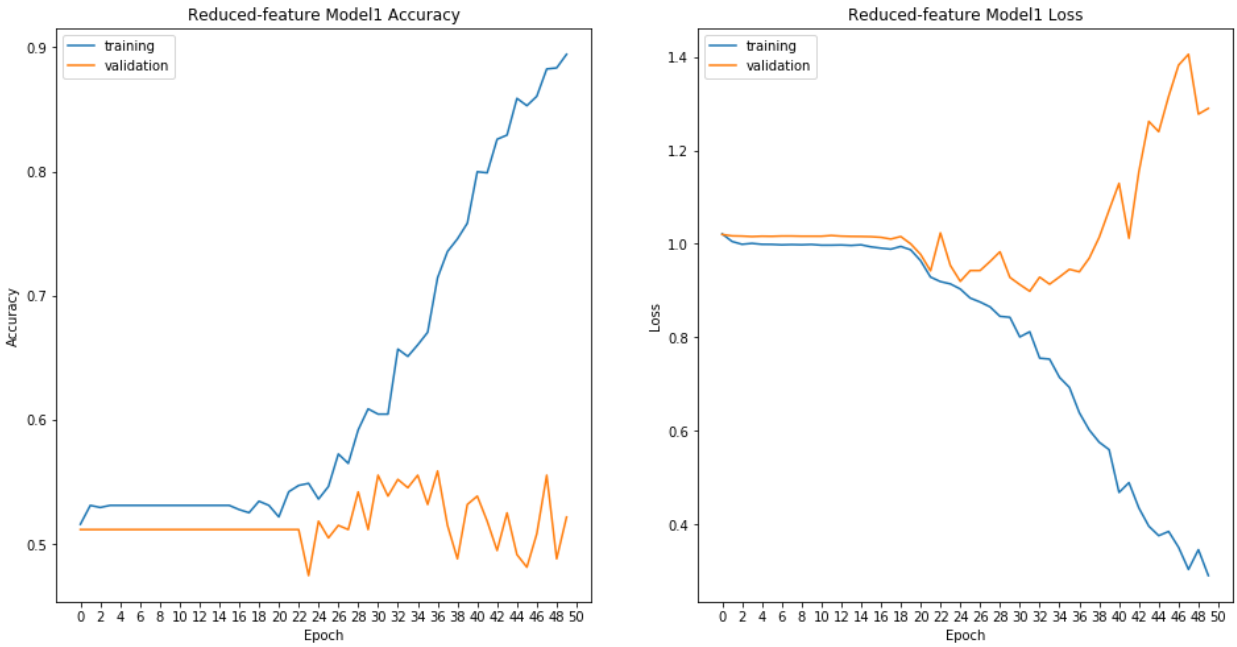


Figure 8. The performance of model 1 after reducing the number of filters over 50 epochs.

The architecture of model 1 after reducing the number of filters (reduced-filters model 1 version 1) is summarized as follows:

Layer (type)	**Input Shape**	**Output Shape**
Block1_Conv2D (Conv2D)	(None, 224, 224, 3)	(None, 222, 222, 8)
Block1_Activation ('relu')	(None, 222, 222, 8)	(None, 222, 222, 8)
Block1_MaxPooling2D	(None, 222, 222, 8)	(None, 111, 111, 8)
Block1_Dropout	(None, 111, 111, 8)	(None, 111, 111, 8)
Block2_Conv2D (Conv2D)	(None, 111, 111, 8)	(None, 109, 109, 16)
Block2_Activation ('relu')	(None, 109, 109, 16)	(None, 109, 109, 16)
Block2_MaxPooling2D	(None, 109, 109, 16)	(None, 54, 54, 16)
Block2_Dropout	(None, 54, 54, 16)	(None, 54, 54, 16)
Block3_Conv2D (Conv2D)	(None, 54, 54, 16)	(None, 52, 52, 32)
Block3_Activation ('relu')	(None, 52, 52, 32)	(None, 52, 52, 32)
Block3_MaxPooling2D	(None, 52, 52, 32)	(None, 26, 26, 32)
Block3_Dropout	(None, 26, 26, 32)	(None, 26, 26, 32)
Block4_Flatten	(None, 26, 26, 32)	(None, 21632)
Block4_Dense ('relu')	(None, 21632)	(None, 256)
Block4_Dropout	(None, 256)	(None, 256)
Block4_Dense ('softmax')	(None, 256)	(None, 3)

As seen in Figure 8, reducing the number of filters seems to help the performance of model 1 slightly. However, the training time is reduced significantly, which makes this new model 1 a better candidate for the next optimization step.

iv. Tuning the performance of model 1 by using different optimizer/loss function combination

The reduced-filters model 1 was further subjected to next optimization step, which involves changing the optimizer and the loss function in the compiling step. I chose to optimize 9 different optimizer/loss function that includes the following loss functions: mean_squared_error, categorical_crossentropy, binary_crossentropy, and the following optimizers: sgd, adagrad, adam. The best combination of optimizer and loss function is binary_crossentropy and sgd.

v. Further reducing the number of filters in model 1

Lastly, I chose to further reduce the number of filters in model 1 and repeated the optimization step in the iv part to determine if the model performance would improve and if the best optimizer/loss function combo would change. The new reduced-filters model 1 (reduced-filters model 1 version 2) has the following architecture:

Layer (type)	**Input Shape**	**Output Shape**
Block1_Conv2D (Conv2D)	(None, 224, 224, 3)	(None, 222, 222, 7)
Block1_Activation (‘relu’)	(None, 222, 222, 7)	(None, 222, 222, 7)
Block1_MaxPooling2D	(None, 222, 222, 7)	(None, 111, 111, 7)
Block1_Dropout	(None, 111, 111, 7)	(None, 111, 111, 7)
Block2_Conv2D (Conv2D)	(None, 111, 111, 7)	(None, 109, 109, 7)
Block2_Activation (‘relu’)	(None, 109, 109, 4)	(None, 109, 109, 4)
Block2_MaxPooling2D	(None, 109, 109, 4)	(None, 54, 54, 4)
Block2_Dropout	(None, 54, 54, 4)	(None, 54, 54, 4)
Block3_Conv2D (Conv2D)	(None, 54, 54, 4)	(None, 52, 52, 4)
Block3_Activation (‘relu’)	(None, 52, 52, 5)	(None, 52, 52, 5)
Block3_MaxPooling2D	(None, 52, 52, 5)	(None, 26, 26, 5)
Block3_Dropout	(None, 26, 26, 5)	(None, 26, 26, 5)
Block4_Flatten	(None, 26, 26, 5)	(None, 3380)
Block4_Dense (‘relu’)	(None, 3380)	(None, 256)
Block4_Dropout	(None, 256)	(None, 256)
Block4_Dense (‘softmax’)	(None, 256)	(None, 3)

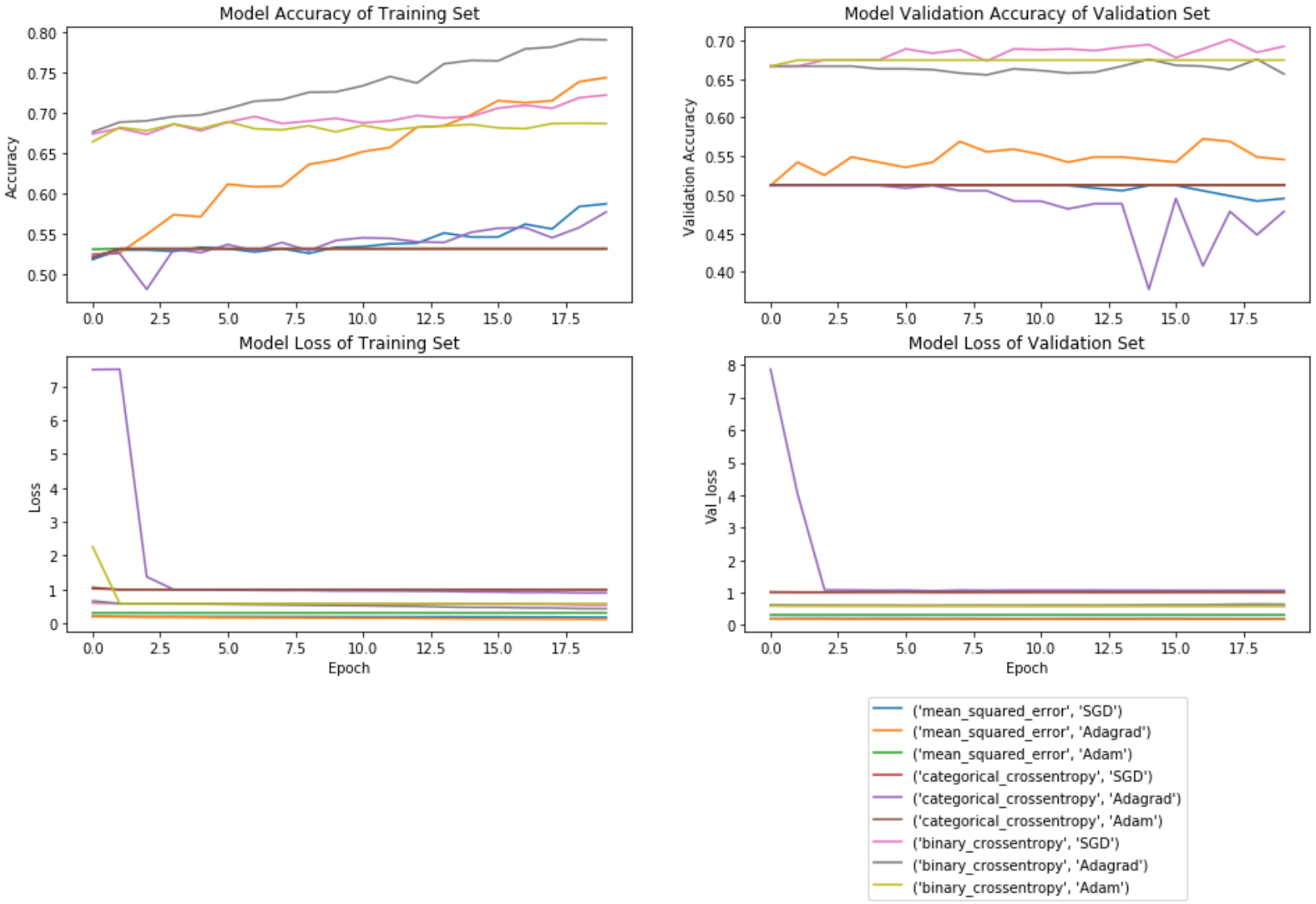


Figure 9. Performance of model 1 after further reducing the number of filters using different optimizer/loss function combination.

After further reducing the number of filters, the best combination of optimizer and loss function is still binary_crossentropy and sgd.

V. PREDICTION

In terms of prediction, I chose to use the two model 1 with a lower number of filters in the previous step. I also chose to use SGD and binary_crossentropy as my optimizer and loss function with 'relu' as the activation method in block 1, block 2, and block 3 of model 1.

The following results were obtained after using each model to predict the cervix type of the testing dataset:

Model 1	Accuracy score
Reduced-filtered model 1 version 1	24.9%
Reduced-filtered model 1 version 2	47.7%

VI. CONCLUSION

The best model that was found in this study in the following architecture:

Layer (type)	**Input Shape**	**Output Shape**
Block1_Conv2D (Conv2D)	(None, 224, 224, 3)	(None, 222, 222, 7)
Block1_Activation ('relu')	(None, 222, 222, 7)	(None, 222, 222, 7)
Block1_MaxPooling2D	(None, 222, 222, 7)	(None, 111, 111, 7)
Block1_Dropout	(None, 111, 111, 7)	(None, 111, 111, 7)
Block2_Conv2D (Conv2D)	(None, 111, 111, 7)	(None, 109, 109, 7)
Block2_Activation ('relu')	(None, 109, 109, 4)	(None, 109, 109, 4)
Block2_MaxPooling2D	(None, 109, 109, 4)	(None, 54, 54, 4)
Block2_Dropout	(None, 54, 54, 4)	(None, 54, 54, 4)
Block3_Conv2D (Conv2D)	(None, 54, 54, 4)	(None, 52, 52, 4)
Block3_Activation ('relu')	(None, 52, 52, 5)	(None, 52, 52, 5)
Block3_MaxPooling2D	(None, 52, 52, 5)	(None, 26, 26, 5)
Block3_Dropout	(None, 26, 26, 5)	(None, 26, 26, 5)
Block4_Flatten	(None, 26, 26, 5)	(None, 3380)
Block4_Dense ('relu')	(None, 3380)	(None, 256)
Block4_Dropout	(None, 256)	(None, 256)
Block4_Dense ('softmax')	(None, 256)	(None, 3)

The best optimizer and loss function that works well for this model is binary_crossentropy and stochastic gradient descent. Rectified Linear Unit (relu) shows to be the best activation method to be used for block 1, block 2, and block 3 of model 1. This model was able to predict accurately up to 47.7% of the testing dataset.

The process in training a model in deep learning requires a lot of time and thus in order to improve or optimize any model performance, we would need a better GPU and larger storage memory capacity.