Data Science and AI

Capstone Project:

Cancer Histopathology

Image Classification

By:

Pozniak Dhani

1. PERSONAL INFORMATION

Name: Dhani louis Shaya Pozniak
Email: dhani@pozitsolutions.com.au
GitHub: https://github.com/pozengineer/dataScienceImageClassify/
Web Application: https://lit-harbor-25638.herokuapp.com/

A highly motivated and hardworking individual, who is seeking a challenging role where I can expand my skills and knowledge. My education in Data Science at University of Technology, Full-Stack Web Development at the University of Sydney, and self-taught knowledge and experience, combined with my interest, passion and enjoyment in coding, programming, Cloud Computing and Virtualization, will enable me to learn, contribute and excel in this industry. My ability to learn new coding languages and programs enables me to understand coding concepts quickly which can then be transferred and utilized in developing user-friendly web-based applications using html, css, javaScript, bootstrap, jQuery, Python and many more coding languages and concepts.

The skills and experience I have will be beneficial in collecting, maintaining, analysing and presenting data which will assist with the translation of business requirements into insights and reports. My passion and knowledge in Data Science enables me to undertake any of the following tasks:

- Perform exploratory data analysis (EDA) using library such as numpy/ pandas and produce graphs utilising matplotlib and Tableau.
- Able to use Python in conjunction with relational databases such as SQLite, MongoDB.
- Use API calls in order to retrieve data using google platforms such as big query.
- Predictive machine learning algorithms such as logistic regression, support vector machine, naive bayes, random forest and gradient boosting.
- Work with natural language processing and image classification.
- Work with deep learning algorithms such as neural networks.
- Familiar with S3 and Sagemaker in AWS.

Table of Contents

2. BACKGROUND INFORMATION

Histopathology is the investigation of cells and tissues at a microscopic level by a histopathologist. These medical experts are responsible for determining appropriate actions/ treatments for patients, depending on the nature and severity of the disease. Digital histopathology has evolved significantly in modern medicine. It requires special processing techniques that enable machines to create digital histopathological images. These images visualize the status of biological structure which can be used to diagnose diseases like cancer. Specialized cameras with a microscope are use to acquire these images. The specimen is embedded in wax and stained with dyes to separate cellular components so that the architecture and constituent of the sample can be studied for diagnosis. The preparation process consists of five stages:

- Fixation: Formaldehyde or glutaraldehyde solutions are added to the samples to prevent autolysis and putrefaction.
- Processing: This is a two-part process consisting of Dehydration and Clearing. Dehydration is where water is extracted from the tissue and substituted with alcohol that solidifies it. Clearing involves a solvent for the embedding paraffin and dehydrating agent being used to remove the dehydrator.
- Tissue Embedding: Wax is used as a medium provide external support to the tissue and allow thin sectioning when solidified.
- Sectioning: The details of the microstructure characterization of the cells are visualized by generating superfine slices of the tissue samples.
- Staining: H & E is used to increase the contrast to the tissue sample and highlight specific features. The sample is then mounted on a glass slide.

After the sample has been processed, the diagnostician compares the histopathological image with those of healthy tissue samples in order to identify differences or abnormalities. The preparation process aids the diagnostician in analysis and classification of the tissue sample.

3. MACHINE LEARNING IN HISTOPATHOLOGY

Machine Learning (ML) can be viewed as an artificial intelligence (AI) that is not specifically programmed to but utilises algorithms to use historical data allowing software applications to predict accurate new output values which can be used in decision making for enterprises for future endeavours. The ML platform wars will intensify as it continues to become an integral part of business operations and as AI becomes more practical in the enterprise environment. To produce an algorithm that can perform a task at an optimal level requires extensive training. Programming a machine that can apply context learned from one task to future, different tasks require research and techniques into deep learning and AI that will create flexible models and will lead to increasing focus in developing general applications.

These digitized images have assisted diagnosticians in the visualization and diagnosis of tissue samples. This has introduced a new era in computer-aided diagnosis (CAD) and has become the new challenge for machine algorithms. These ML models preserve performance while decreasing the workload for pathologists so that they can focus on other tasks. CAD models that are able to precisely and confidently label classes and minimize reading interpretation run times are essential for businesses to integrate them into their current system processes. The processes for a CAD model in large patient populations would include receiving a batch of raw images, pre-processing them, and then produce an array of diagnosis labels. Areas where CAD models can be applied in the histopathological industry include:

- Cancer detection in the supplied tissue.
- Disease characterization through automatic grading that quantifies the malignancy level.
- Discovery and separation of cell/ nuclei/ gland regions from images by segmentation.
- Multi-class classification of various subtypes of a specific cancer.
  (Shehata, M., 2021. https://encyclopedia.pub/entry/8696)

Increased storage/ cloud capabilities has made it easier to store large numbers of digitized images enabling fast remote analysis and secure pathology information. Automating diagnostics is becoming increasingly important as evident from the recent global pandemic. Implementing these ML models will offload some of the pressure for Doctors and diagnosticians by reducing redundancy, and improving accuracy and capacity. Convolutional Neural Networks (CNN) has become the most popular ML method for image classification.

Large sizes in digitized images create a challenge when it comes to storage capacity, pre-processing the data so that it is compatible with deep learning algorithms. A common method to deal with this issue is normalizing the image. This results in a trade-off where data is lost which affects the performance of the ML model. Open- source availability of histopathological images is limited due to privacy policies that enforce confidentiality and protection of patient's data which presents another challenge for deep learning algorithms, as they require large datasets to implement train and test sets for fitting and evaluating purposes. Data augmentation and image data generator are tools used to apply transformations to the image such as rotation, shifting, scaling, introducing noise/ blur, and colour value variations which multiplies the dataset dealing with the limited amount of data available.

4. DATA PREPROCESSIING

Jupyter Notebook:

- cancerHistopath01_dataPreprocessing.ipynb
- cancerHistopath02_EDA.ipynb

Dataset:

- Borkowski, "Lung and Colon Cancer Histopathological Images", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/lung-and-colon-cancer-histopathological-images. [Accessed: 30- Jul- 2022].

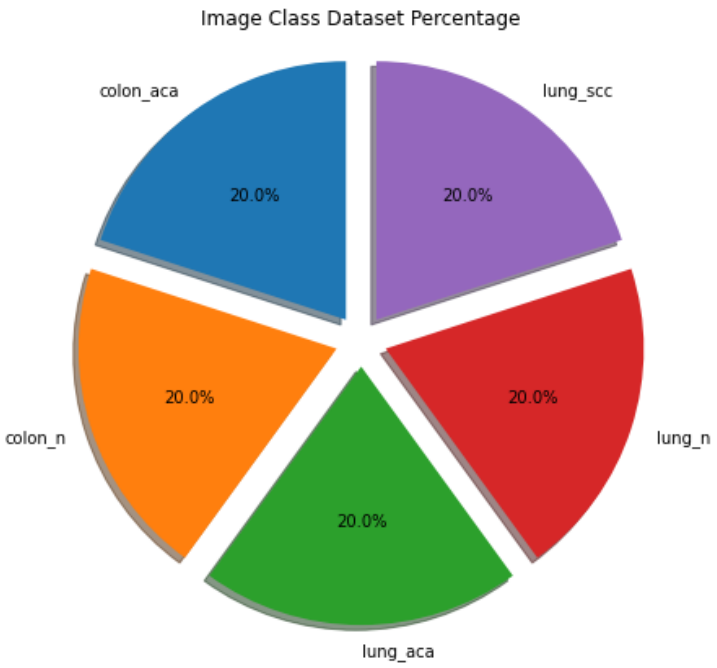
Sources:

- Borkowski AA, Bui MM, Thomas LB, Wilson CP, DeLand LA, Mastorides SM. Lung and Colon Cancer Histopathological Image Dataset (LC25000). arXiv:1912.12142v1 [eess.IV], 2019

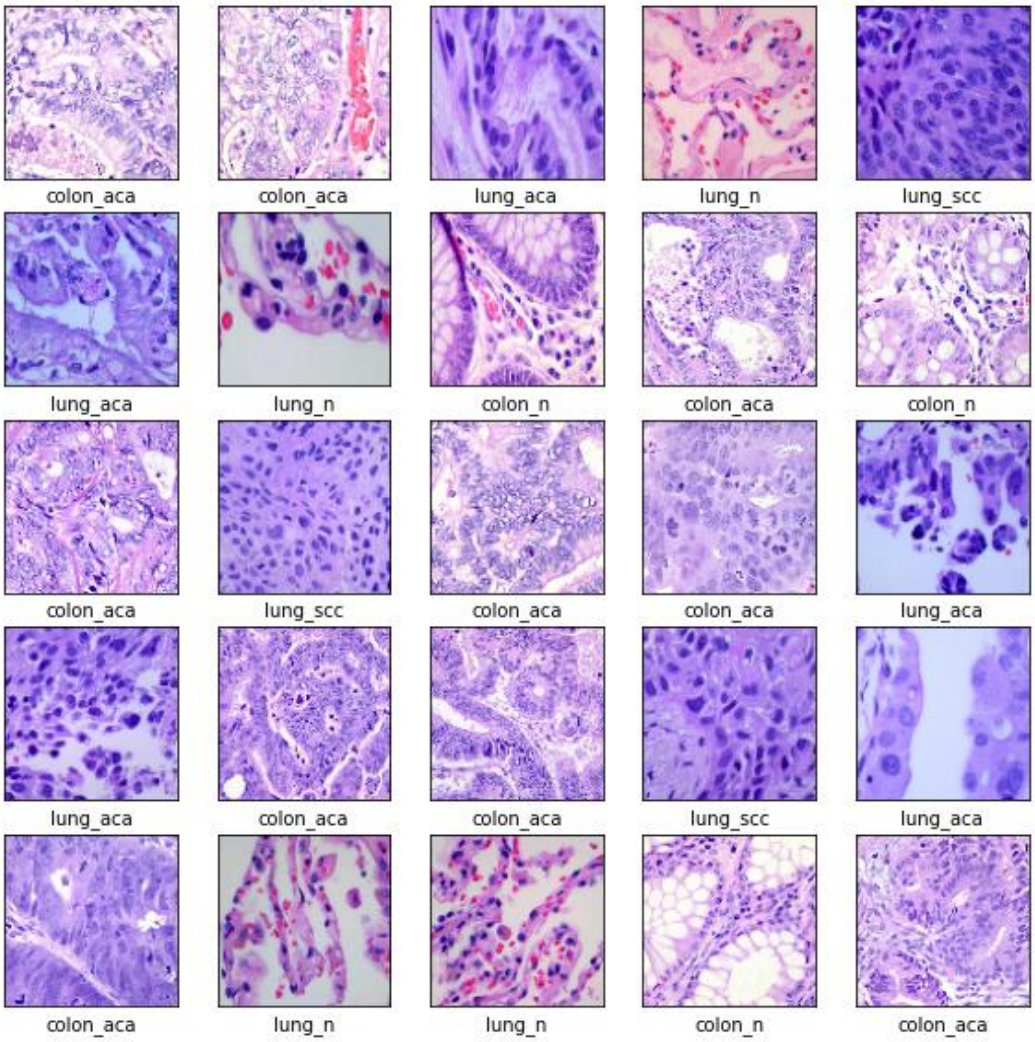
Image Labels:

- Lung benign tissue: 3
- Lung adenocarcinoma: 2
- Lung squamous cell carcinoma: 4
- Colon adenocarcinoma: 0
- Colon benign tissue: 1

The dataset is made up of 25000 images with 5 classes. Initial analysis as shown in the figure below illustrates a balanced dataset which will have a positive effect on the performance of the machine/ deep learning algorithms.



Images were resized from 768x768 to 96x96 which will be beneficial with storage capacity. Also, with the limited resources available, it allows us to process, fit and evaluate the data and algorithms quicker with minimal lagging. A sample of images from the dataset were loaded successfully as shown below.



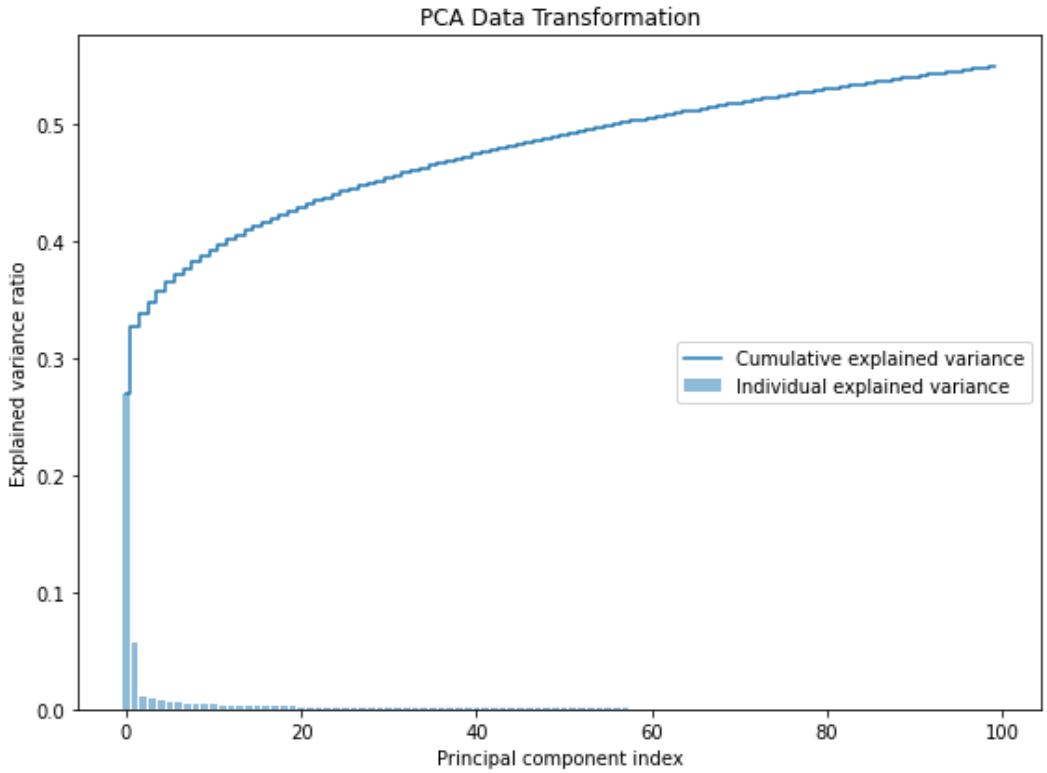
As mentioned before, after the sample has been processed, the diagnostician compares the histopathological image with those of healthy tissue samples in order to identify differences or abnormalities. Similarly with machine algorithms, after they have been trained, will take the new image and compare it with the patterns and features it has learned during its training phase which will enable it to accurately match the image with the respective class.

5. PCA DATA TRANSFORMATION

Jupyter Notebook:

- cancerHistopath03_PCA.ipynb

After flattening the image array, PCA was able to reduce the number of features from 27649 to 100. This represented 55% of the spread of data which means that half the data was lost in the transformation but this was necessary to prevent overfitting when fitting this dataset to machine learning algorithms such as Logistic Regression.



6. MACHINE LEARNING ALGORITHMS

Jupyter Notebook:

- cancerHistopath04_simpleModels.ipynb
- cancerHistopath13_functions.ipynb

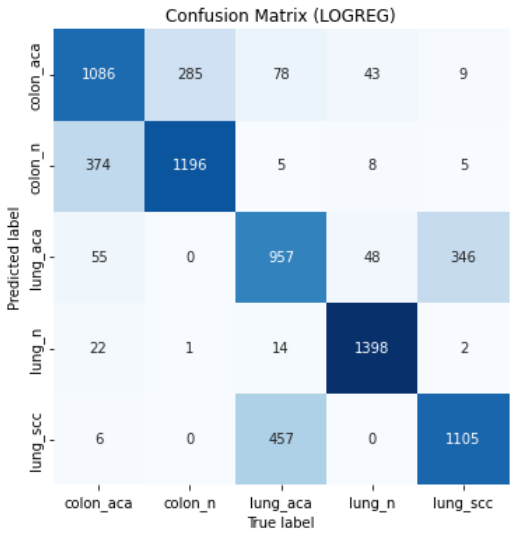
6.1. Logistic Regression

Hyper-parameter: C=1

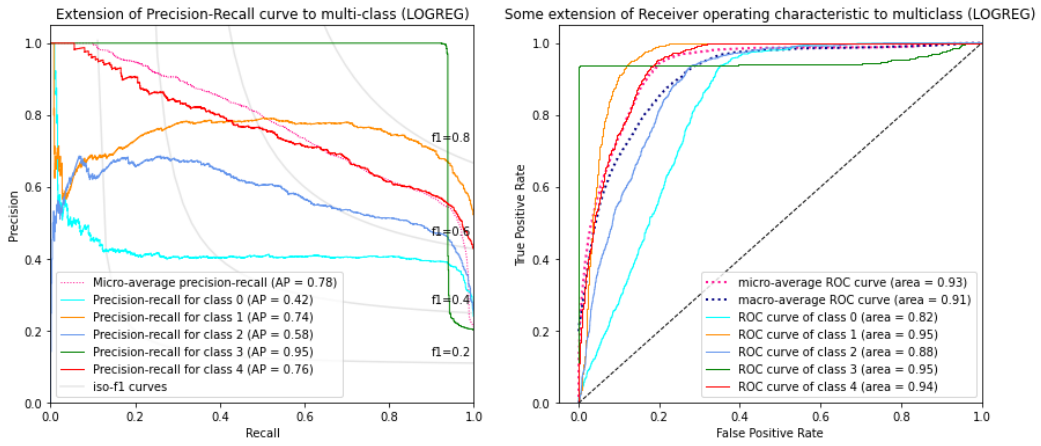
Classification Summary Report

```
Accuracy  : 0.7656 [TP / N] Proportion of predicted labels that match the true
labels.
Precision : 0.7670 [TP / (TP + FP)] Not to label a negative sample as positive.
Recall    : 0.7663 [TP / (TP + FN)] Find all the positive samples.
f1-score  : 0.7660 [2 * (Precision * Recall)/ (Precision + Recall)]
ROC AUC   : 0.9454
```

Confusion Matrix



Precision/ Recall/ Roc-Curve



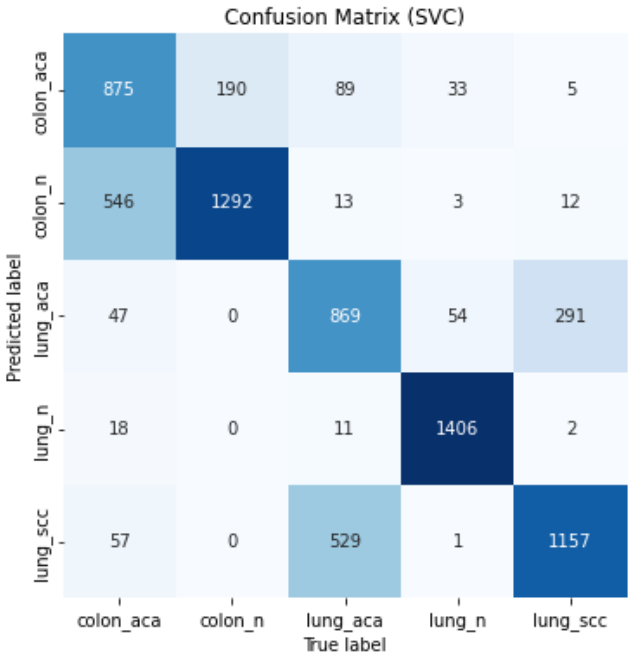
6.2. Support Vector Machine

Hyper-parameter: C=1

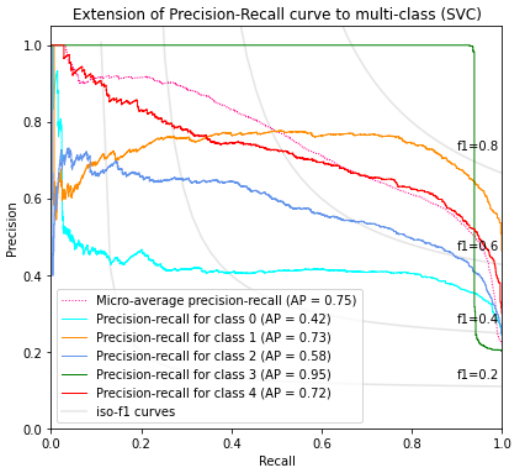
Classification Summary Report

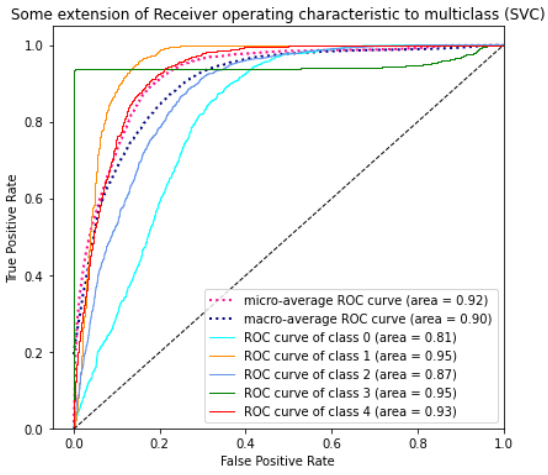
```
Accuracy  : 0.7465 [TP / N] Proportion of predicted labels that match the true
labels.
Precision : 0.7515 [TP / (TP + FP)] Not to label a negative sample as positive.
Recall    : 0.7484 [TP / (TP + FN)] Find all the positive samples.
f1-score  : 0.7435 [2 * (Precision * Recall)/ (Precision + Recall)]
ROC AUC   : 0.9267
```

Confusion Matrix



Precision/ Recall/ Roc-Curve



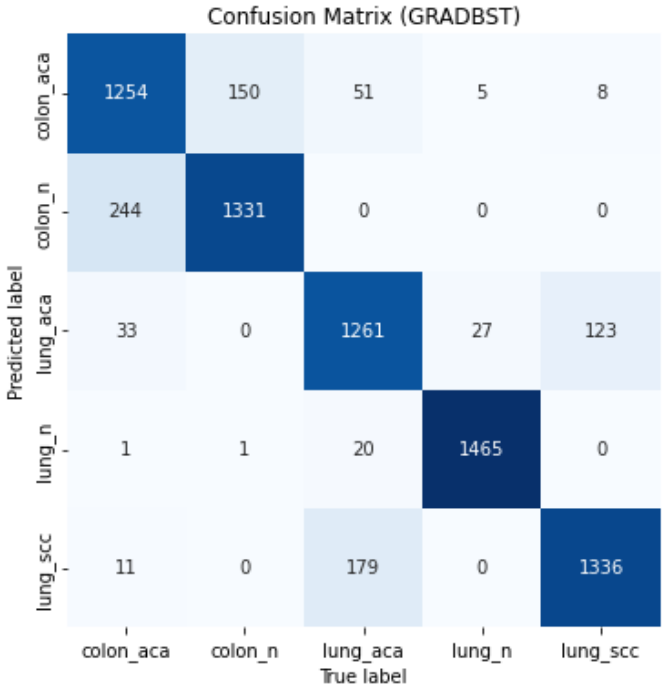


6.3. Gradient Boosting

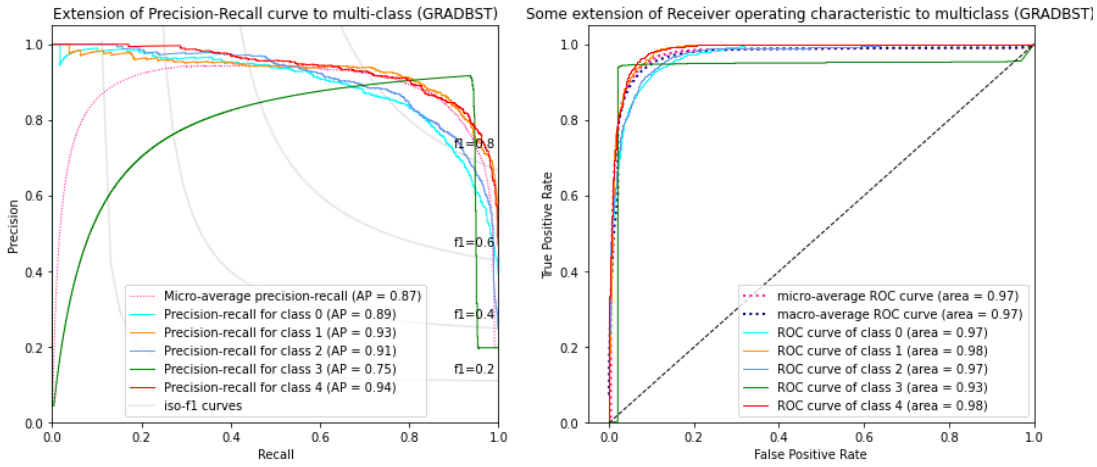
Classification Summary Report

```
Accuracy  : 0.8863 [TP / N] Proportion of predicted labels that match the true
labels.
Precision : 0.8867 [TP / (TP + FP)] Not to label a negative sample as positive.
Recall    : 0.8869 [TP / (TP + FN)] Find all the positive samples.
f1-score  : 0.8864 [2 * (Precision * Recall)/ (Precision + Recall)]
ROC AUC   : 0.9839
```

Confusion Matrix

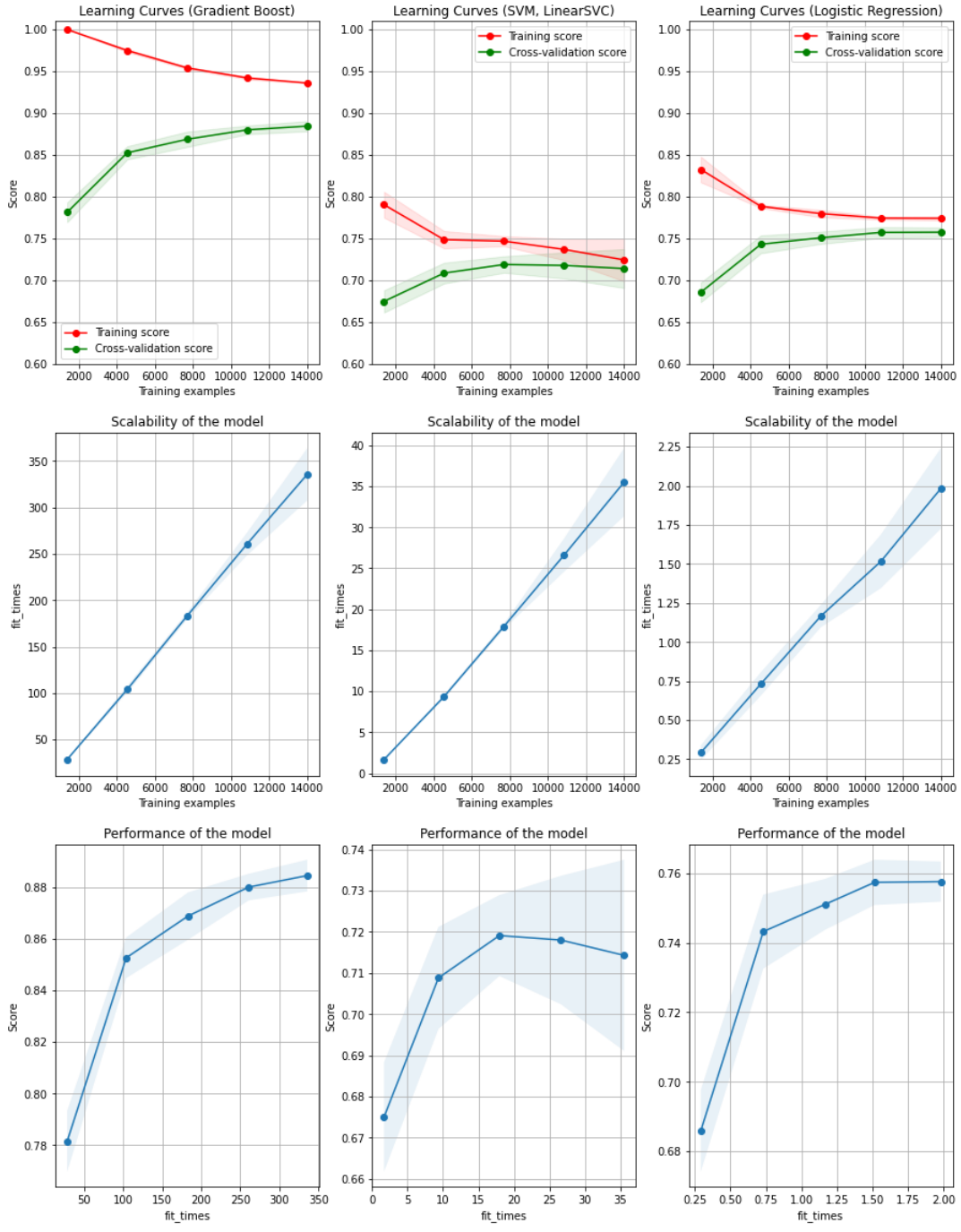


Precision/ Recall/ Roc-Curve



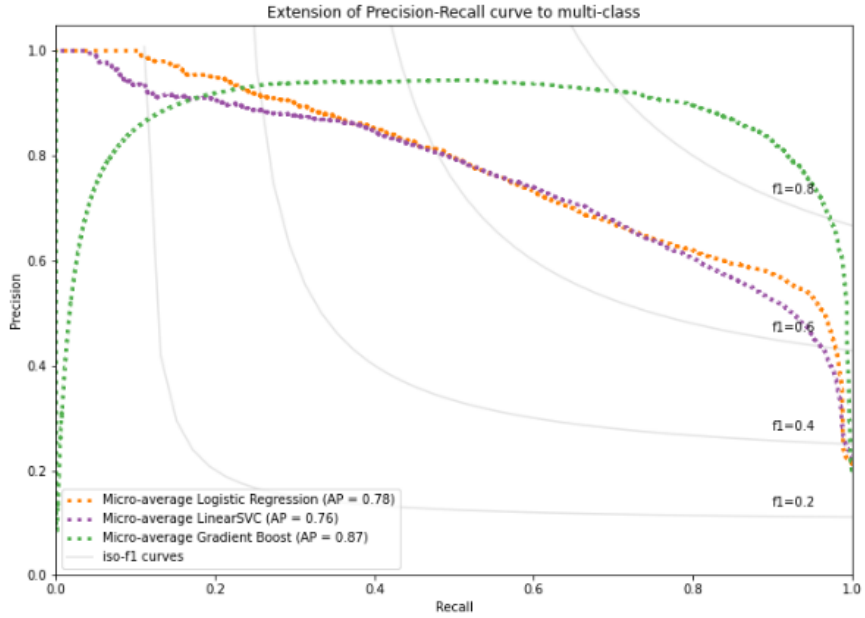
6.4. Learning Curve

The machine learning algorithms performed well considering half the data was lost during the PCA transformation but it was necessary to prevent overfitting. Convolutional Neural Networks (CNN) are the preferred algorithm when it comes to image classification

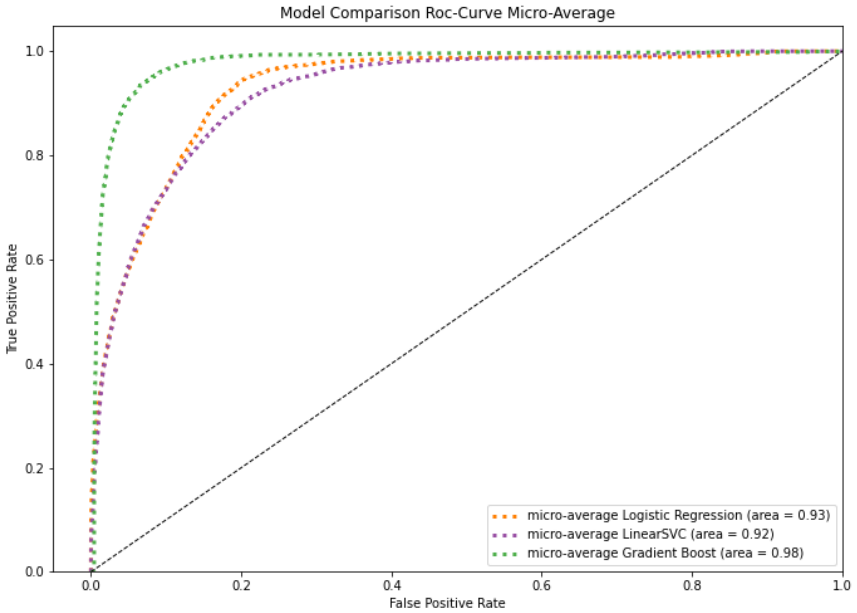


6.5. Precision/ Recall Model Comparison

Gradient Boosting was the superior machine learning algorithm and able to achieve 88% accuracy with half the data lost. This is not ideal but interesting to know that simple models can be considered in image classification projects.



6.6. Roc-Curve Model Comparison



7. PRE-TRAINED CNN MODELS

Jupyter Notebook:

- cancerHistopath05_preTrainedModels.ipynb

Before experimenting with custom CNN architecture, it was advised to explore models that have already been trained for image classification in order to become familiar with the hidden layers, sequence and structure used. The following models were selected for analysis:

- VGG16
- VGG19
- Xception
- InceptionV3

When creating the train/ test set for the models, the image arrays should not be normalized otherwise it will affect the performance of the pre-trained models in a negative way and will produce unexpected results.

There might be a bug with the Xception model because after awhile it kept on restarting the computer after one epoch run so take caution when implementing this model. Due to limited resources and computer processing capacities, the runtimes for the pre-trained models were long and as a result had to fit the models using 10 epochs. Refer to the appendix for details of the architecture for the following pre-trained models:

- VGG19
- Xception
- InceptionV3

8. CUSTOM-BUILT MODELS

Jupyter Notebook:

- cancerHistopath08_cnnMarkModels.ipynb

After analysing the pre-trained models, it was observed that they consist of many hidden layers and are high in complexity. The aim was to build multiple CNN models that were smaller in size and be able to process and evaluate the test set faster. There were six variations built. The initial models cnnMk01 and cnnMk02 were trained on pre-normalized image arrays similar to the pre-trained models. The remaining four models were trained on normalized image arrays.

There was an initial problem obtaining positive results due to loading the images in BGR. This led to the models being trained on BGR images, so when predictions were made on RGB images, the performance of the model was decreased. In the future, check what type of data is being used to train the deep learning algorithms as this type of mistake can significantly affect the performance of the models. Techniques such as BatchNormalization, Dropout, MaxPooling, and Dense Layers were used in the architecture of the custom-built models.

A large amount of time was focused on trial and error in adjusting the hidden layers to achieve maximum performance. The sequence of the hidden layers and the position of BatchNormalization played a major role in the performance of the model. Refer to the appendix for details of the architecture for the following custom-built models:

- cnnMk01
- cnnMk02
- cnnMk03
- cnnMk04
- cnnMk05

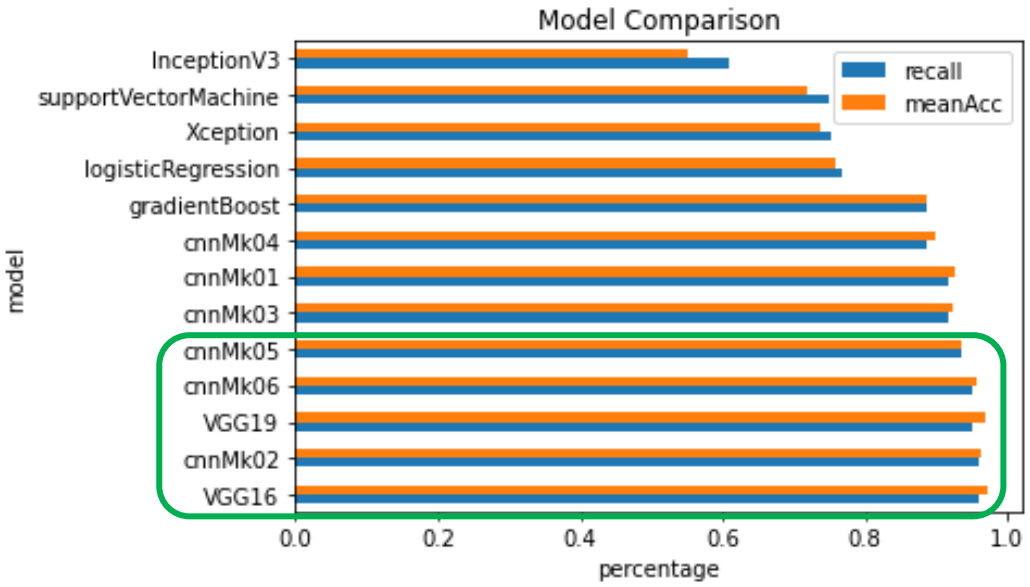
9. MODEL COMPARISON

Jupyter Notebook:

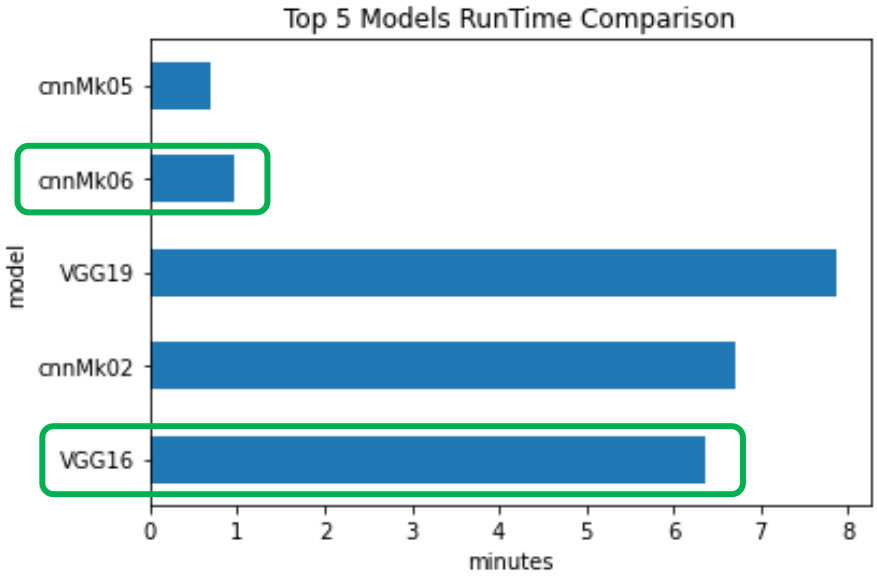
- cancerHistopath10_modelComparison.ipynb

Recall (false negatives), the proportion of actual positives identified correctly, was the metric used to compare the models. As illustrated in the graph below the top 5 models are:

- VGG16
- cnnMk02
- VGG19
- cnnMk06
- cnnMk05



The runtimes were considered in order to select the top pretrained and custom-built models to determine whether a custom-built model could perform better than a pre-trained model. Runtime for 7500 unseen images processed and classified.



	precision	recall	meanAcc	runTime_min
VGG16	0.96	0.96	0.97	6.36
cnnMk02	0.96	0.96	0.96	6.72
VGG19	0.95	0.95	0.97	7.86
cnnMk06	0.95	0.95	0.96	0.98
cnnMk05	0.94	0.93	0.93	0.70
cnnMk03	0.92	0.92	0.92	1.06
cnnMk01	0.92	0.92	0.93	0.88
cnnMk04	0.90	0.89	0.90	1.24
gradientBoost	0.89	0.89	0.89	4.86
logisticRegression	0.77	0.77	0.76	0.02
Xception	0.81	0.75	0.74	4.01
supportVectorMachine	0.75	0.75	0.72	1.58
InceptionV3	0.74	0.61	0.55	36.54

10. PRE-TRAINED VS. CUSTOM-BUILT

Jupyter Notebook:

- cancerHistopath05_preTrainedModels.ipynb
- cancerHistopath09_cnnMarkModels.ipynb
- cancerHistopath09_cnnEvalModels.ipynb

10.1. VGG16

- 21 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 6 minutes with 96% accuracy

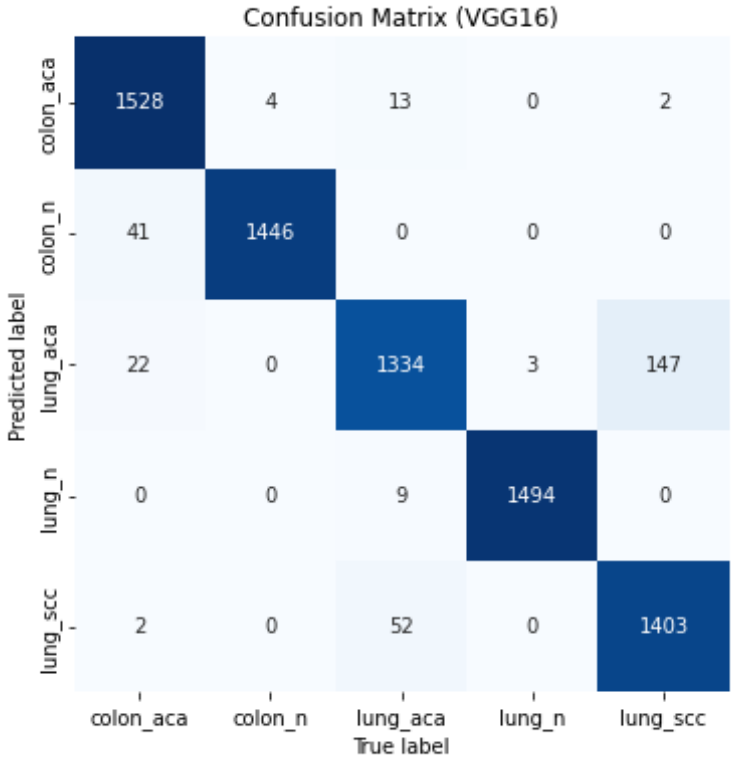
Model Architecture

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 96, 96, 3)]       0

 block1_conv1 (Conv2D)       (None, 96, 96, 64)        1792

 block1_conv2 (Conv2D)       (None, 96, 96, 64)        36928

 block1_pool (MaxPooling2D)  (None, 48, 48, 64)        0

 block2_conv1 (Conv2D)       (None, 48, 48, 128)       73856

 block2_conv2 (Conv2D)       (None, 48, 48, 128)       147584

 block2_pool (MaxPooling2D)  (None, 24, 24, 128)       0

 block3_conv1 (Conv2D)       (None, 24, 24, 256)       295168

 block3_conv2 (Conv2D)       (None, 24, 24, 256)       590080

 block3_conv3 (Conv2D)       (None, 24, 24, 256)       590080

 block3_pool (MaxPooling2D)  (None, 12, 12, 256)       0

 block4_conv1 (Conv2D)       (None, 12, 12, 512)       1180160

 block4_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 6, 6, 512)         0

 block5_conv1 (Conv2D)       (None, 6, 6, 512)         2359808

 block5_conv2 (Conv2D)       (None, 6, 6, 512)         2359808
```

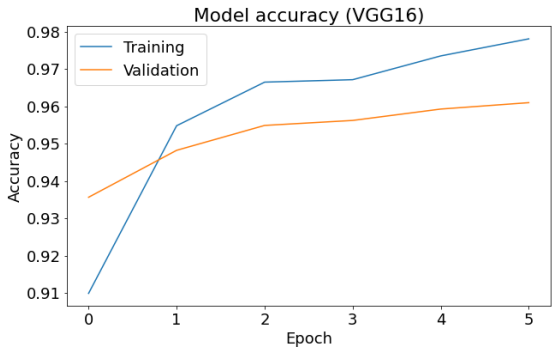
```
 block5_conv3 (Conv2D)        (None, 6, 6, 512)         2359808

 block5_pool (MaxPooling2D)   (None, 3, 3, 512)         0

 flatten_2 (Flatten)          (None, 4608)              0

 dense_4 (Dense)              (None, 5)                 23045

=================================================================
Total params: 14,737,733
Trainable params: 23,045
Non-trainable params: 14,714,688
```

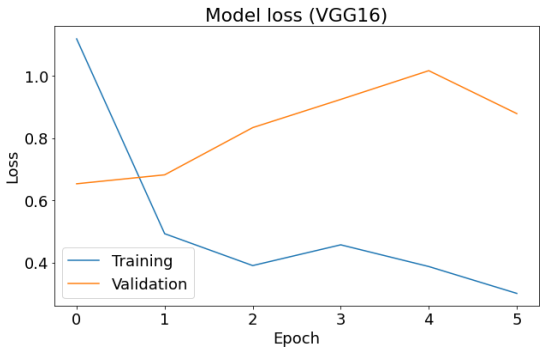
Classification Summary Report

	precision	recall	f1-score	support
0	0.96	0.99	0.97	1547
1	1.00	0.97	0.98	1487
2	0.95	0.89	0.92	1506
3	1.00	0.99	1.00	1503
4	0.90	0.96	0.93	1457
accuracy			0.96	7500
macro avg	0.96	0.96	0.96	7500
weighted avg	0.96	0.96	0.96	7500



Visualization of Cost





10.2. cnnMk06

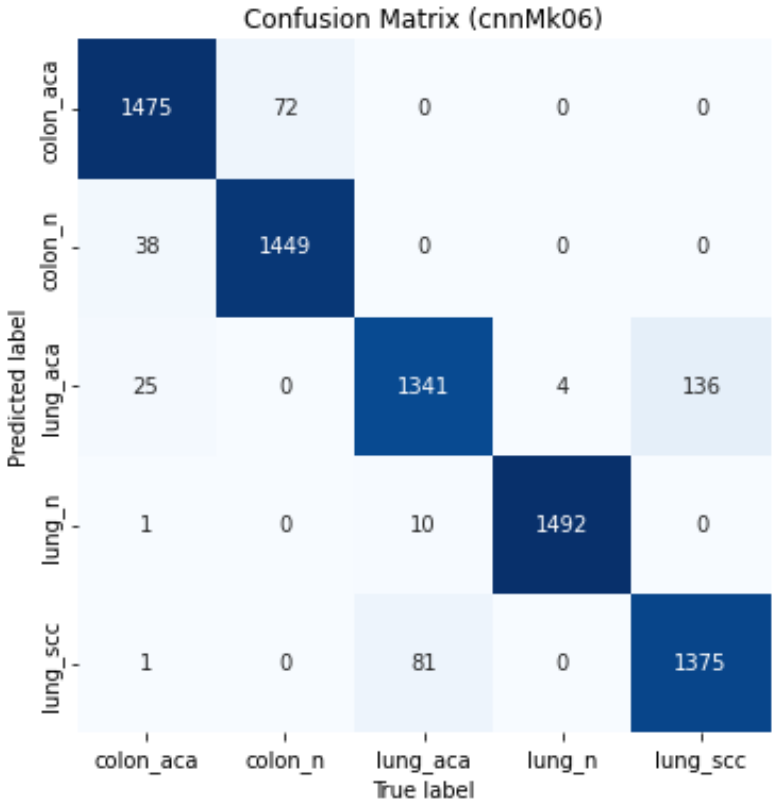
- 8 Hidden Layers
- 15 epochs
- Classified 7500 unseen images in 1 minute with 95% accuracy

Model Architecture

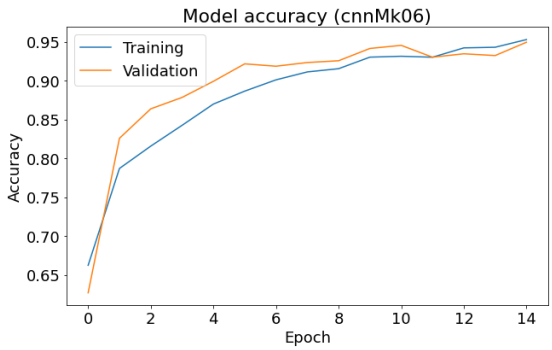
```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_57 (Conv2D)          (None, 94, 94, 64)        1792

 max_pooling2d_38 (MaxPoolin  (None, 18, 18, 64)       0
 g2D)

 conv2d_58 (Conv2D)          (None, 16, 16, 32)        18464

 max_pooling2d_39 (MaxPoolin  (None, 8, 8, 32)         0
 g2D)

 conv2d_59 (Conv2D)          (None, 6, 6, 32)          9248

 flatten_19 (Flatten)        (None, 1152)              0

 dense_24 (Dense)            (None, 32)                36896

 dense_25 (Dense)            (None, 5)                 165

=================================================================
Total params: 66,565
Trainable params: 66,565
Non-trainable params: 0
```

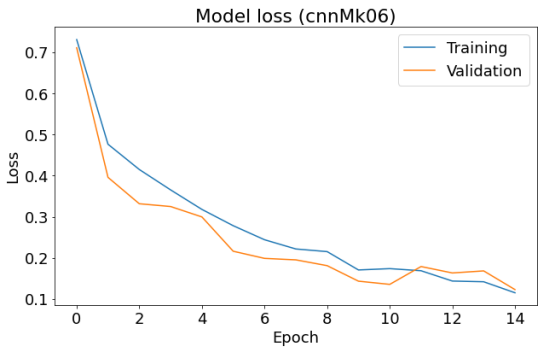
Classification Summary Report

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1547
1	0.95	0.97	0.96	1487
2	0.94	0.89	0.91	1506
3	1.00	0.99	0.99	1503
4	0.91	0.94	0.93	1457
accuracy			0.95	7500
macro avg	0.95	0.95	0.95	7500
weighted avg	0.95	0.95	0.95	7500



Visualization of Cost





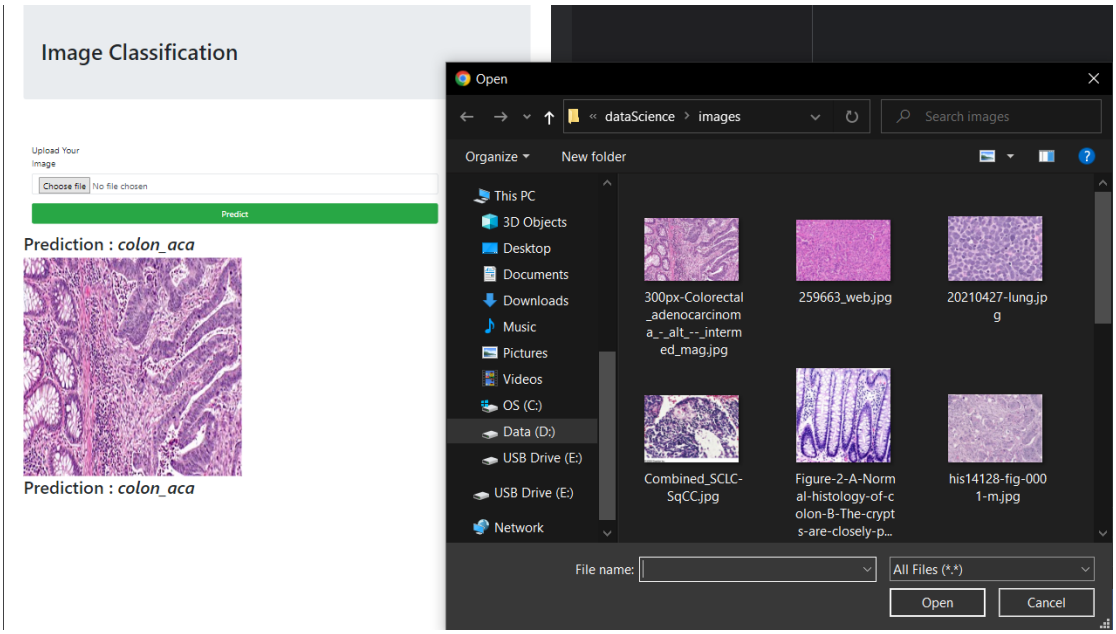
11. WEB APPLICATION

Jupyter Notebook:

- cancerHistopath06_webApp.ipynb
- cancerHistopath07_imageDataGenerator.ipynb

11.1. Deployment

GitHub: https://github.com/pozengineer/dataScienceImageClassify/
Web Application: https://lit-harbor-25638.herokuapp.com/



11.2. Considerations

User Interface and User Experience (UI/UX): These skills are used in the development of mobile applications and most of the time are skipped due to developers enabling actions instead of satisfying the end users interaction with the app and encouraging it.

Design: The application might function as designed but clients and users pay more attention to how the application looks and interacts with them on their devices. Developers need to make sure the application is appealing to the end user.

Cloud Infrastructure: Microsoft Azure and Amazon Web Services (AWS) are platforms that utilize global networks of data centres for deployment and management of web applications.

Cross-Platform Capabilities: Initially you will choose a specific platform like Android, iOS, or Windows to begin creating the application but eventually you will want your app to be able to run across all platforms. Transferring the coding from one platform to another might prove to be more complicated if not anticipated.

Data Expertise: Collecting and analysing data will be an important skill to acquire to determine the effectiveness of the app and how many users the app has attracted. This is achieved through efficient databases and repositories.

Security: The user's personal information and data is important to consider when developing an app that requires user register and login functions. You want to gain the users trust by making them feel that their data is safe and won't be used for other purposes besides what the app will use it for. You need to make sure your systems are protected from outside hackers.

(Firenze, A., 2020, Ascendle)

11.3. Tools and Technologies

In terms of designing the application, there are several tools and technologies that are being used. These python libraries include:

- Flask==2.1.0
- Gunicorn==20.1.0
- Jinja2==3.1.1
- joblib==1.1.0
- keras==2.9.0
- Keras-Preprocessing==1.1.2
- MarkupSafe==2.1.1
- Numpy==1.23.1
- Pandas==1.4.2
- Pillow==9.1.0
- Tensorflow-cpu==2.9.1

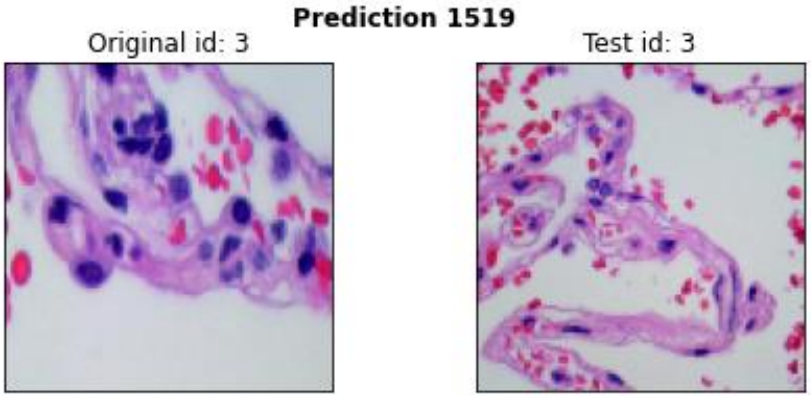
Within these Tools and Technologies, there are a number that will be key:

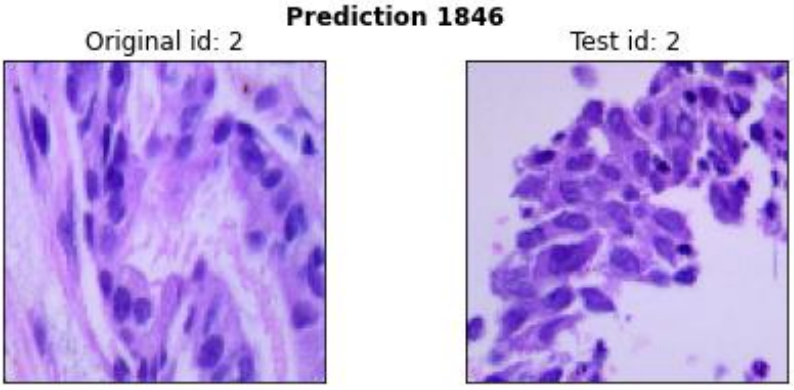
- **HTML**: HyperText Markup Language that allows the developer to describe pages. The website will use **HTML** and **CSS**, with **jQuery** as the JavaScript library, allowing the website to easily use JavaScript to add interactivity as well as content that updates in a dynamic fashion.
- **CSS**: Style Sheet Language that allows the developer to style an HTML document
- **Bootstrap**: Library of HTML and CSS files and code https://getbootstrap.com/
- **Heroku**: Is a platform as a service (PaaS) that enables developers to build, run, and operate applications in the cloud.

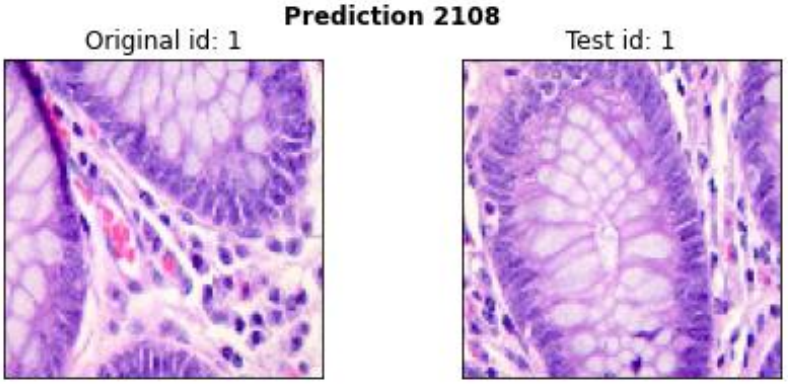
12. TEST EVALUATION RESULTS

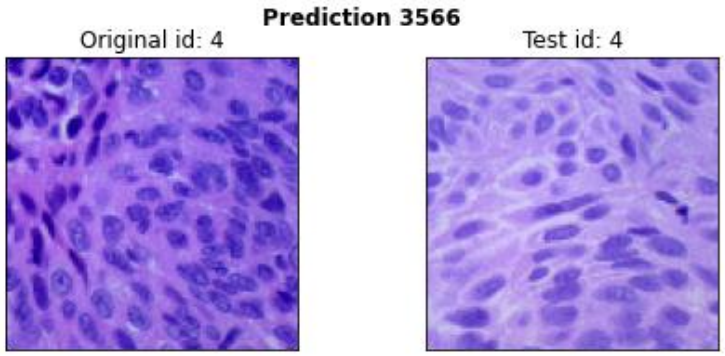
Jupyter Notebook:

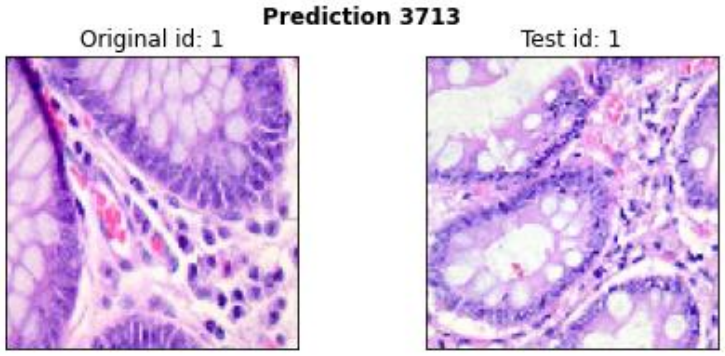
- cancerHistopath17_testEvalResults.ipynb

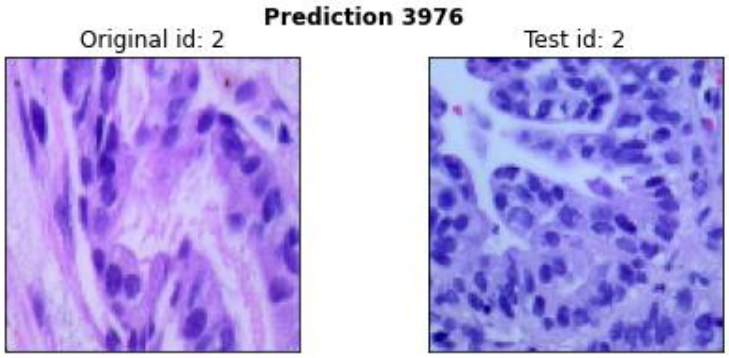


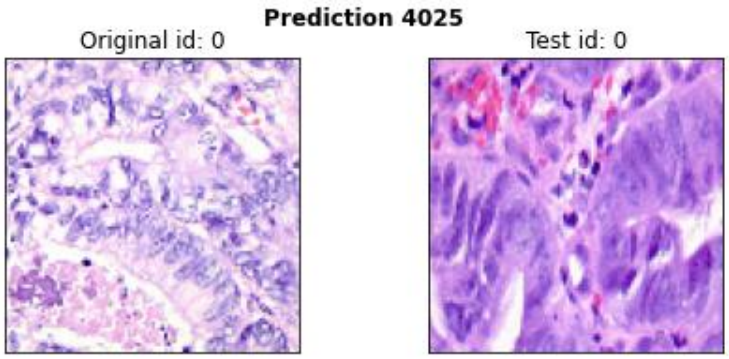


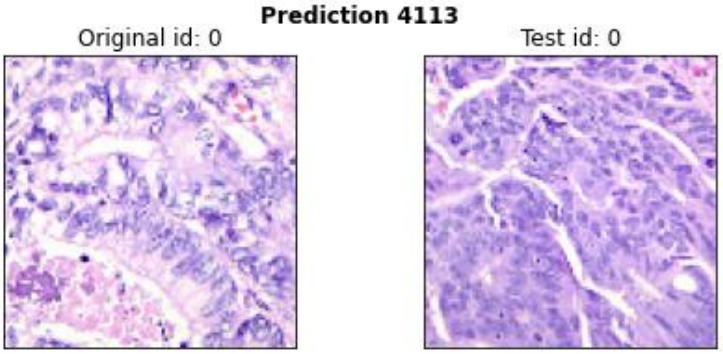


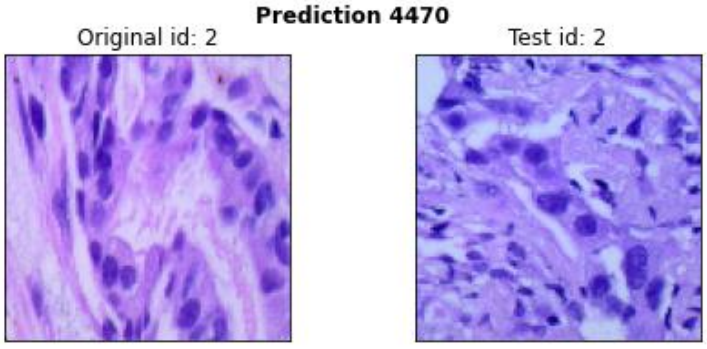

Prediction 3566
Original id: 4
Test id: 4

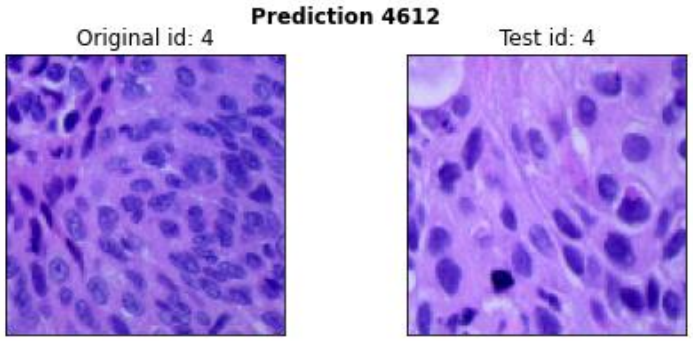

Prediction 3713
Original id: 1
Test id: 1

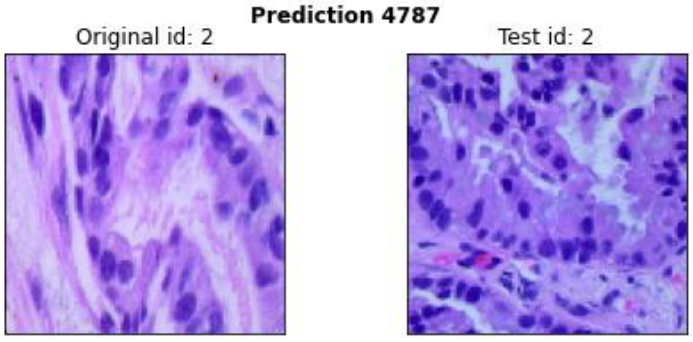

Prediction 3976
Original id: 2
Test id: 2


Prediction 4025
Original id: 0
Test id: 0


Prediction 4113
Original id: 0
Test id: 0


Prediction 4470
Original id: 2
Test id: 2


Prediction 4612
Original id: 4
Test id: 4


Prediction 4787
Original id: 2
Test id: 2

13. CONCLUSION

At this juncture, the results of the deep learning algorithms indicate a promising future for image classification models becoming successfully integrated into the medical industry. This does not mean that machines will replace humans. A balanced dataset has a positive effect on the performance of machine learning algorithms, if a business had a balanced implementation between machines and humans, this would have a positive effect on productivity, efficiency, and capacity. The CNN model's ability to process large amounts of data quickly will allow diagnosticians to focus on research and development into more advanced classification techniques. Also, there is a lot that can be learned from the deep learning algorithms in the way that they are able to differentiate one image from another. This will lead to other features and characteristics of cancerous cell structures that humans might have never considered.

Due to limited resources, the amount of epochs used for fitting the models was limited and the runtimes recorded might be different if the models were run on a machine with higher processing capabilities.

The cnnMk06 legitimate model is able to predict with 95% confidence the class of the histopathological image

Future development recommendations would include:

- Consider separate models for each cancer (colon/ lung)
- With better processing equipment, extend the number of epochs run
- Further finetuning of custom model
- store large numbers of digitized images enabling fast remote analysis and secure pathology information (96x96) instead of (768x768)
- preserve performance while decreasing the workload for pathologists so that they can focus on future research and development to enhance and evolve the medical industry

14. REFERENCES

- Burns, E., 2021. *In-Depth Guide to Machine Learning in the Enterprise*. [online] SearchEnterpriseAI. Available at: <https://www.techtarget.com/searchenterpriseai/In-depth-guide-to-machine-learning-in-the-enterprise> [Accessed 10 July 2022].
- Burns, E., 2022. *What Is Machine Learning and Why Is It Important?*. [online] SearchEnterpriseAI. Available at: <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.> [Accessed 10 July 2022].
- Firenze, A., 2020. *"14 Essential App Development Team Skills You Must Have – Ascendle"*. Ascendle, 2020 [online]. Available at: <https://ascendle.com/ideas/14-skills-your-app-development-team-must-have/> [Accessed 19 July 2022].
- Heath, N., 2020. *What is machine learning? Everything you need to know | ZDNet*. [online] ZDNet. Available at: <https://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/> [Accessed 10 July 2022].
- IBM Cloud Education, 2020. *What is Machine Learning?*. [online] IBM. Available at: <https://www.ibm.com/cloud/learn/machine-learning> [Accessed 10 July 2022].
- Mishra, N., 2020. *Deep Learning in Histopathology*. [online] Medium. Available at: <https://medium.com/swlh/deep-learning-in-histopathology-c104478c00cd> [Accessed 10 August 2022].
- Mishra, N., 2020. *Deep Learning in Histopathology*. [online] Medium. Available at: <https://towardsdatascience.com/deep-learning-in-histopathology-35c0294d38eb> [Accessed 10 August 2022].
- SAS, 2022. *Machine Learning: What it is and why it matters*. [online] SAS. Available at: <https://www.sas.com/en_au/insights/analytics/machine-learning.html#machine-learning-importance> [Accessed 10 July 2022].
- Selig, J., 2022. *What Is Machine Learning? A Definition.*. [online] Expert.ai. Available at: <https://www.expert.ai/blog/machine-learning-definition/> [Accessed 10 July 2022].
- Shehata, M., 2021. *Histopathological Images*. [online] Encyclopedia.pub. Available at: <https://encyclopedia.pub/entry/8696> [Accessed 10 August 2022].

15. APPENDIX

Jupyter Notebook:

- cancerHistopath05_preTrainedModels.ipynb
- cancerHistopath08_cnnMarkModels.ipynb
- cancerHistopath09_cnnEvalModels.ipynb

15.1. VGG19

- 24 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 7.86 minutes with 95% accuracy

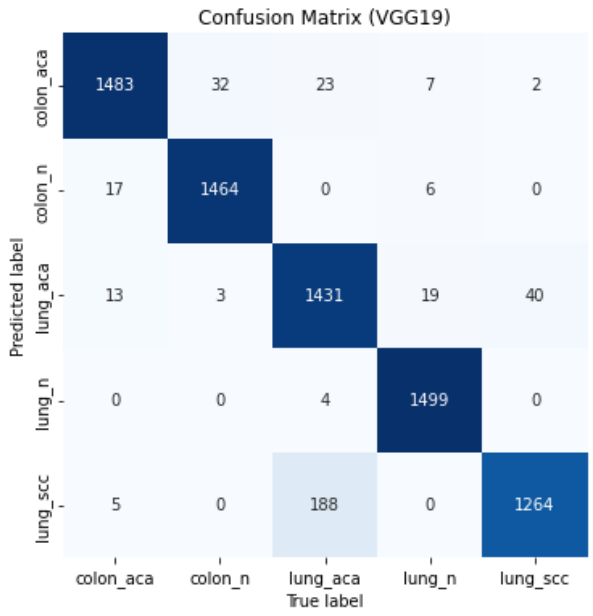
Model Architecture

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 96, 96, 3)]       0

 block1_conv1 (Conv2D)       (None, 96, 96, 64)        1792

 block1_conv2 (Conv2D)       (None, 96, 96, 64)        36928

 block1_pool (MaxPooling2D)  (None, 48, 48, 64)        0

 block2_conv1 (Conv2D)       (None, 48, 48, 128)       73856

 block2_conv2 (Conv2D)       (None, 48, 48, 128)       147584

 block2_pool (MaxPooling2D)  (None, 24, 24, 128)       0

 block3_conv1 (Conv2D)       (None, 24, 24, 256)       295168

 block3_conv2 (Conv2D)       (None, 24, 24, 256)       590080

 block3_conv3 (Conv2D)       (None, 24, 24, 256)       590080

 block3_conv4 (Conv2D)       (None, 24, 24, 256)       590080

 block3_pool (MaxPooling2D)  (None, 12, 12, 256)       0

 block4_conv1 (Conv2D)       (None, 12, 12, 512)       1180160

 block4_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_conv4 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 6, 6, 512)         0
```

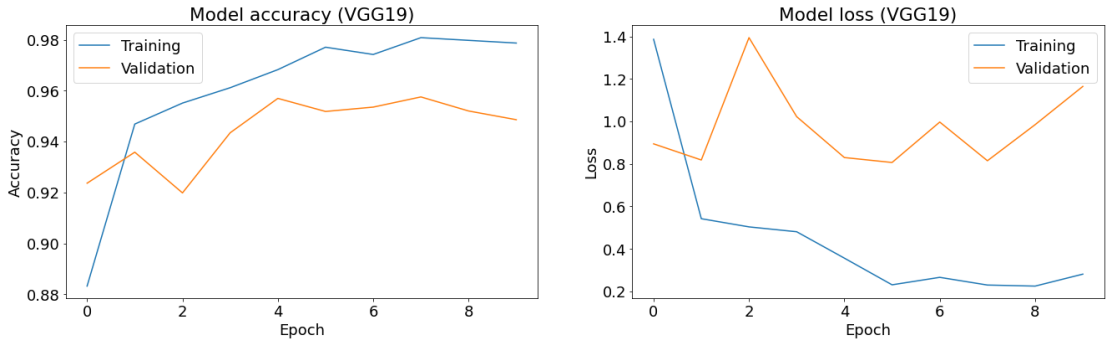
```
 block5_conv1 (Conv2D)        (None, 6, 6, 512)         2359808

 block5_conv2 (Conv2D)        (None, 6, 6, 512)         2359808

 block5_conv3 (Conv2D)        (None, 6, 6, 512)         2359808

 block5_conv4 (Conv2D)        (None, 6, 6, 512)         2359808

 block5_pool (MaxPooling2D)   (None, 3, 3, 512)         0

 flatten_19 (Flatten)         (None, 4608)              0

 dense_37 (Dense)             (None, 5)                 23045

=================================================================
Total params: 20,047,429
Trainable params: 23,045
Non-trainable params: 20,024,384
```

Classification Summary Report

	precision	recall	f1-score	support
0	0.98	0.96	0.97	1547
1	0.98	0.98	0.98	1487
2	0.87	0.95	0.91	1506
3	0.98	1.00	0.99	1503
4	0.97	0.87	0.91	1457
accuracy			0.95	7500
macro avg	0.95	0.95	0.95	7500
weighted avg	0.95	0.95	0.95	7500



Visualization of Cost



15.2. Xception

- 134 Hidden Layers (for full architecture refer to jupyter notebook)
- 10 epochs
- Classified 7500 unseen images in 4 minutes with 75% accuracy

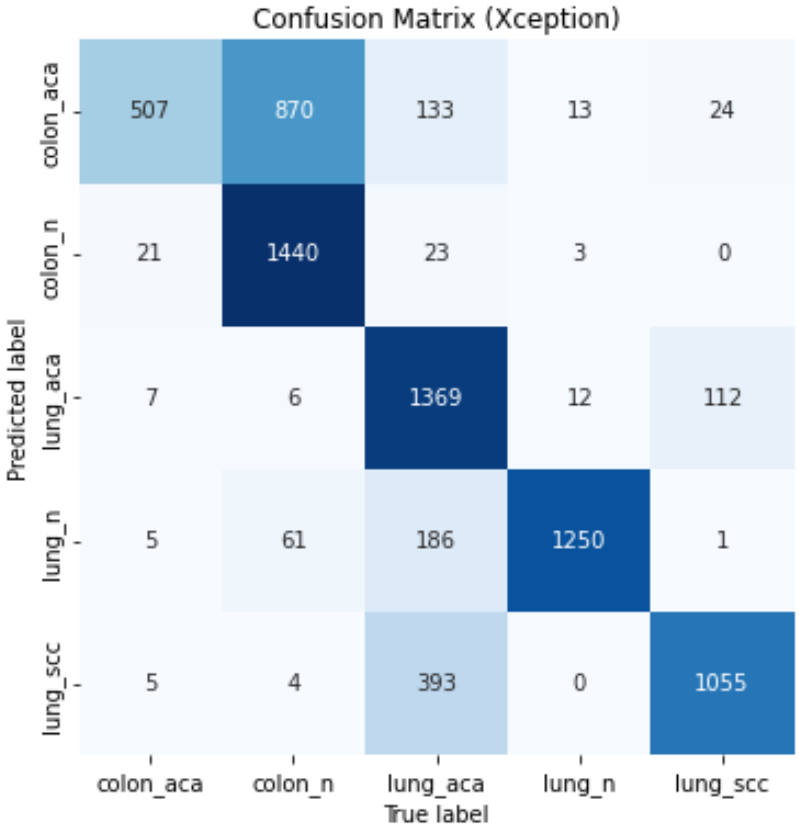
Model Architecture

```
Model: "model_3"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_4 (InputLayer)           [(None, 96, 96, 3)]  0           []

 block1_conv1 (Conv2D)          (None, 47, 47, 32)   864         ['input_4[0][0]']

 block1_conv1_bn (BatchNormaliz  (None, 47, 47, 32)  128         ['block1_conv1[0][0]']
 ation)

 block1_conv1_act (Activation)  (None, 47, 47, 32)   0           ['block1_conv1_bn[0][0]']

 block1_conv2 (Conv2D)          (None, 45, 45, 64)   18432       ['block1_conv1_act[0][0]']

 block1_conv2_bn (BatchNormaliz  (None, 45, 45, 64)  256         ['block1_conv2[0][0]']
 ation)

 block1_conv2_act (Activation)  (None, 45, 45, 64)   0           ['block1_conv2_bn[0][0]']

 block13_sepconv2 (SeparableCon  (None, 6, 6, 1024)  752024      ['block13_sepconv2_act[0][0]']

 conv2d_5 (Conv2D)              (None, 3, 3, 1024)   745472      ['add_10[0][0]']
```

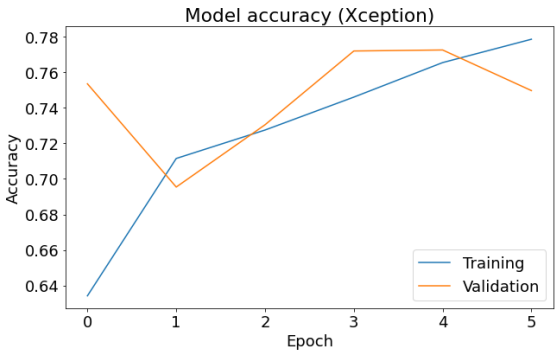
```
 block13_pool (MaxPooling2D)    (None, 3, 3, 1024)   0           ['block13_sepcon
v2_bn[0][0]']

 batch_normalization_3 (BatchNo  (None, 3, 3, 1024)  4096        ['conv2d_5[0][0]
']
 rmalization)

 add_11 (Add)                   (None, 3, 3, 1024)   0           ['block13_pool[0
][0]',
                                                                  'batch_normaliz
ation_3[0][0]']

 block14_sepconv1 (SeparableCon  (None, 3, 3, 1536)  1582080     ['add_11[0][0]']
 v2D)

 block14_sepconv1_bn (BatchNorm  (None, 3, 3, 1536)  6144        ['block14_sepcon
v1[0][0]']
 alization)

 block14_sepconv1_act (Activati  (None, 3, 3, 1536)  0           ['block14_sepcon
v1_bn[0][0]']
 on)

 block14_sepconv2 (SeparableCon  (None, 3, 3, 2048)  3159552     ['block14_sepcon
v1_act[0][0]']
 v2D)

 block14_sepconv2_bn (BatchNorm  (None, 3, 3, 2048)  8192        ['block14_sepcon
v2[0][0]']
 alization)

 block14_sepconv2_act (Activati  (None, 3, 3, 2048)  0           ['block14_sepcon
v2_bn[0][0]']
 on)

 flatten_21 (Flatten)           (None, 18432)        0           ['block14_sepcon
v2_act[0][0]']

 dense_39 (Dense)               (None, 5)            92165       ['flatten_21[0][
0]']

==================================================================================
=================
Total params: 20,953,645
Trainable params: 92,165
Non-trainable params: 20,861,480
```

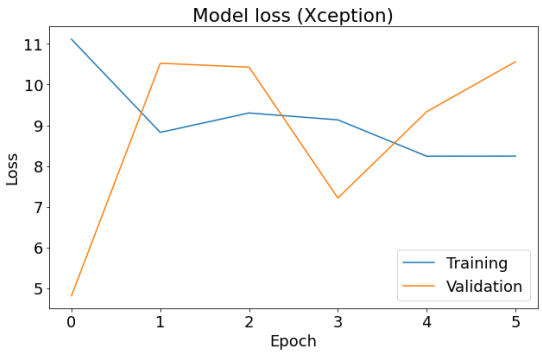
Classification Summary Report

	precision	recall	f1-score	support
0	0.93	0.33	0.48	1547
1	0.60	0.97	0.74	1487
2	0.65	0.91	0.76	1506
3	0.98	0.83	0.90	1503
4	0.89	0.72	0.80	1457
accuracy			0.75	7500
macro avg	0.81	0.75	0.74	7500
weighted avg	0.81	0.75	0.73	7500



Visualization of Cost





15.3. InceptionV3

- 313 Hidden Layers (for full architecture refer to jupyter notebook)
- 10 epochs
- Classified 7500 unseen images in 36.54 minutes with 61% accuracy

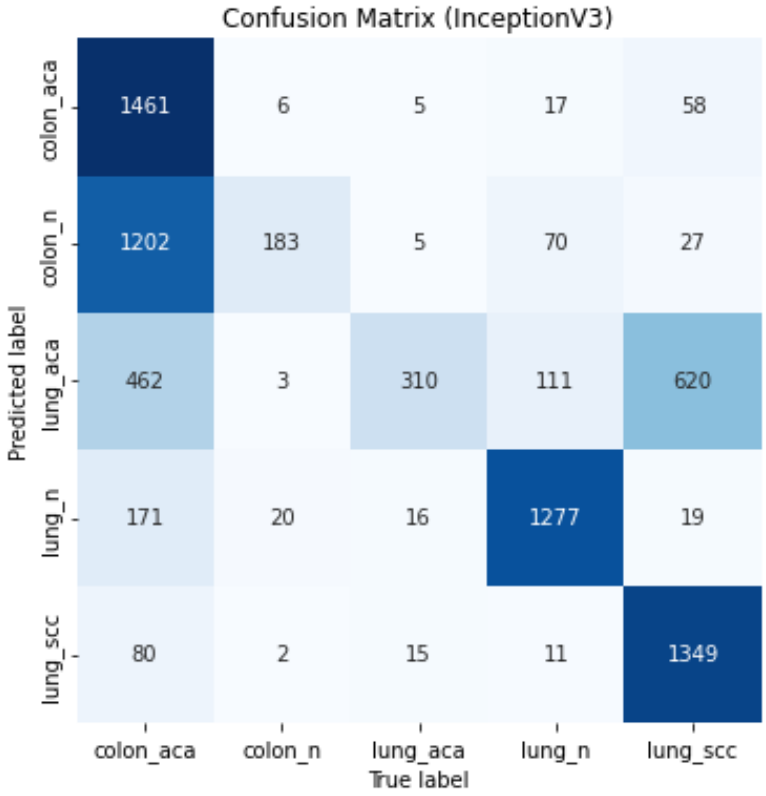
Model Architecture

```
Model: "model_4"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_5 (InputLayer)           [(None, 96, 96, 3)]  0           []

 conv2d_6 (Conv2D)              (None, 47, 47, 32)   864         ['input_5[0][0]'
]

 batch_normalization_4 (BatchNo  (None, 47, 47, 32)  96          ['conv2d_6[0][0]
']
 rmalization)

 activation (Activation)        (None, 47, 47, 32)   0           ['batch_normaliz
ation_4[0][0]']

 conv2d_7 (Conv2D)              (None, 45, 45, 32)   9216        ['activation[0][
0]']

 batch_normalization_5 (BatchNo  (None, 45, 45, 32)  96          ['conv2d_7[0][0]
']
 rmalization)

 activation_1 (Activation)      (None, 45, 45, 32)   0           ['batch_normaliz
ation_5[0][0]']

 conv2d_8 (Conv2D)              (None, 45, 45, 64)   18432       ['activation_1[0
][0]']

 batch_normalization_6 (BatchNo  (None, 45, 45, 64)  192         ['conv2d_8[0][0]
']
 rmalization)

 activation_2 (Activation)      (None, 45, 45, 64)   0           ['batch_normaliz
ation_6[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 22, 22, 64)  0           ['activation_2[0
][0]']

 conv2d_9 (Conv2D)              (None, 22, 22, 80)   5120        ['max_pooling2d_
1[0][0]']
```

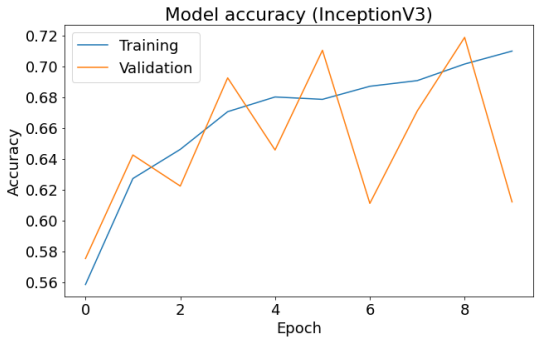
```
 activation_87 (Activation)     (None, 1, 1, 384)    0           ['batch_normaliz
ation_91[0][0]']

 activation_88 (Activation)     (None, 1, 1, 384)    0           ['batch_normaliz
ation_92[0][0]']

 activation_91 (Activation)     (None, 1, 1, 384)    0           ['batch_normaliz
ation_95[0][0]']

 activation_92 (Activation)     (None, 1, 1, 384)    0           ['batch_normaliz
ation_96[0][0]']

 batch_normalization_97 (BatchN  (None, 1, 1, 192)   576         ['conv2d_99[0][0
]']
 ormalization)

 activation_85 (Activation)     (None, 1, 1, 320)    0           ['batch_normaliz
ation_89[0][0]']

 mixed9_1 (Concatenate)         (None, 1, 1, 768)    0           ['activation_87[
0][0]',
                                                                  'activation_88[
0][0]']

 concatenate_1 (Concatenate)    (None, 1, 1, 768)    0           ['activation_91[
0][0]',
                                                                  'activation_92[
0][0]']

 activation_93 (Activation)     (None, 1, 1, 192)    0           ['batch_normaliz
ation_97[0][0]']

 mixed10 (Concatenate)          (None, 1, 1, 2048)   0           ['activation_85[
0][0]',
                                                                  'mixed9_1[0][0]
',
                                                                  'concatenate_1[
0][0]',
                                                                  'activation_93[
0][0]']

 flatten_22 (Flatten)           (None, 2048)         0           ['mixed10[0][0]'
]

 dense_40 (Dense)               (None, 5)            10245       ['flatten_22[0][
0]']

==================================================================================
=================
Total params: 21,813,029
Trainable params: 10,245
Non-trainable params: 21,802,784
```

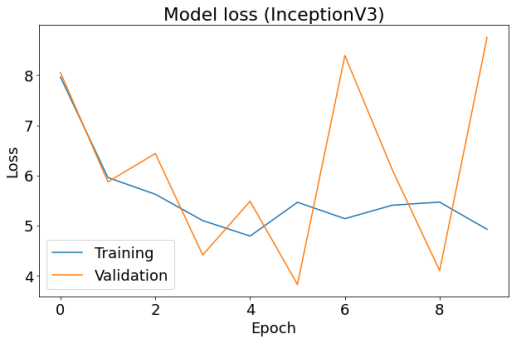
Classification Summary Report

	precision	recall	f1-score	support
0	0.43	0.94	0.59	1547
1	0.86	0.12	0.22	1487
2	0.88	0.21	0.33	1506
3	0.86	0.85	0.85	1503
4	0.65	0.93	0.76	1457
accuracy			0.61	7500
macro avg	0.74	0.61	0.55	7500
weighted avg	0.73	0.61	0.55	7500



Visualization of Cost





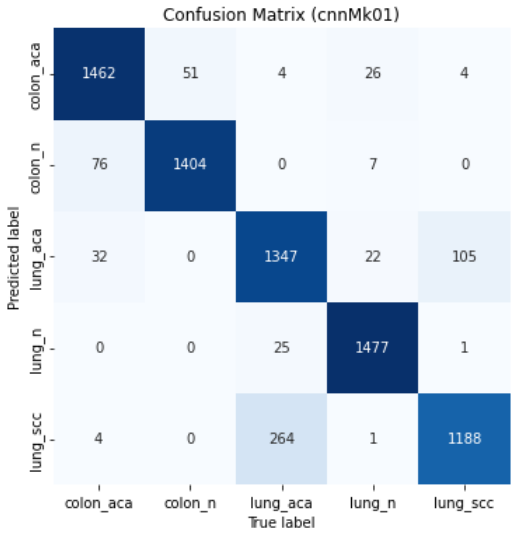
15.4. cnnMk01

- 8 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 0.87 minute with 92% accuracy

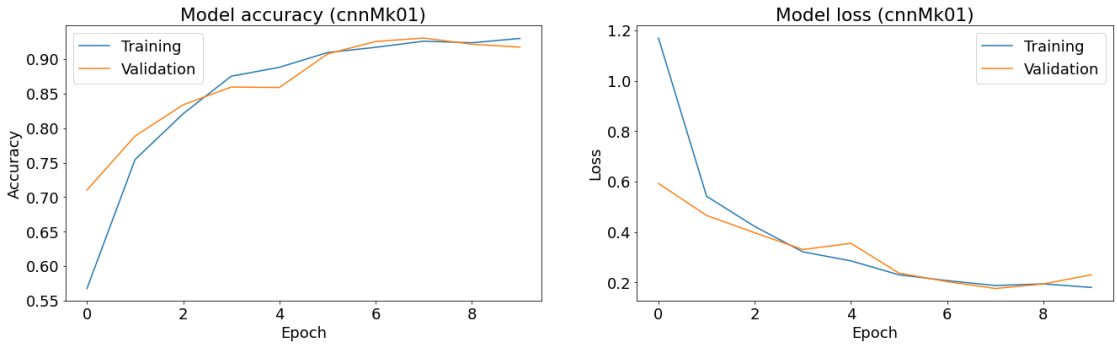
Model Architecture

```
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_163 (Conv2D)          (None, 94, 94, 16)        448

 max_pooling2d_42 (MaxPoolin  (None, 18, 18, 16)        0
 g2D)

 conv2d_164 (Conv2D)          (None, 16, 16, 32)        4640

 max_pooling2d_43 (MaxPoolin  (None, 8, 8, 32)          0
 g2D)

 conv2d_165 (Conv2D)          (None, 6, 6, 32)          9248

 flatten_32 (Flatten)         (None, 1152)              0

 dense_54 (Dense)             (None, 32)                36896

 dense_55 (Dense)             (None, 5)                 165

=================================================================
Total params: 51,397
Trainable params: 51,397
Non-trainable params: 0
```

Classification Summary Report



Visualization of Cost



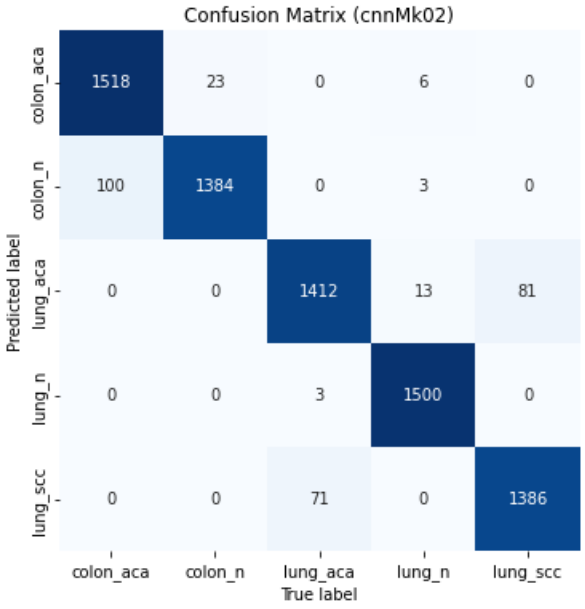
15.5. cnnMk02

- 10 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 6.72 minutes with 96% accuracy

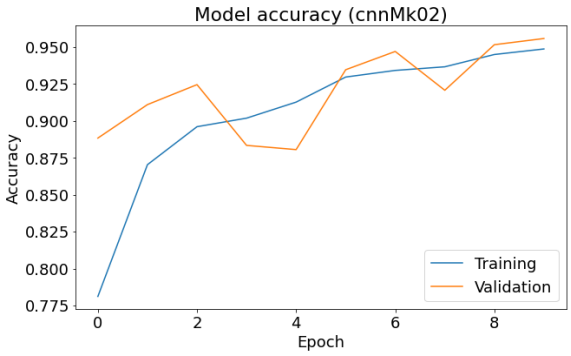
Model Architecture

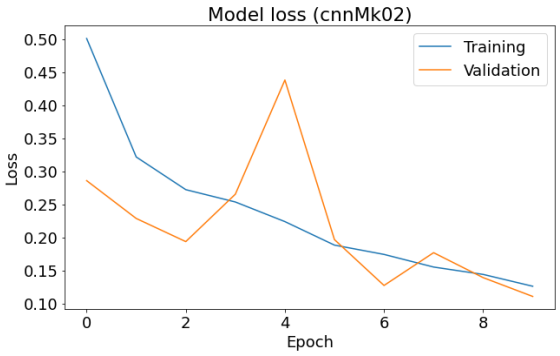
```
Model: "sequential_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_151 (Conv2D)         (None, 94, 94, 32)        896

 max_pooling2d_34 (MaxPoolin  (None, 18, 18, 32)       0
 g2D)

 conv2d_152 (Conv2D)         (None, 16, 16, 64)        18496

 batch_normalization_114 (Ba  (None, 16, 16, 64)       256
 tchNormalization)

 max_pooling2d_35 (MaxPoolin  (None, 8, 8, 64)         0
 g2D)

 conv2d_153 (Conv2D)         (None, 6, 6, 64)          36928

 flatten_28 (Flatten)        (None, 2304)              0

 dense_48 (Dense)            (None, 32)                73760

 dropout_31 (Dropout)        (None, 32)                0

 dense_49 (Dense)            (None, 5)                 165

=================================================================
Total params: 130,501
Trainable params: 130,373
Non-trainable params: 128
```

Classification Summary Report



Visualization of Cost





15.6. cnnMk03

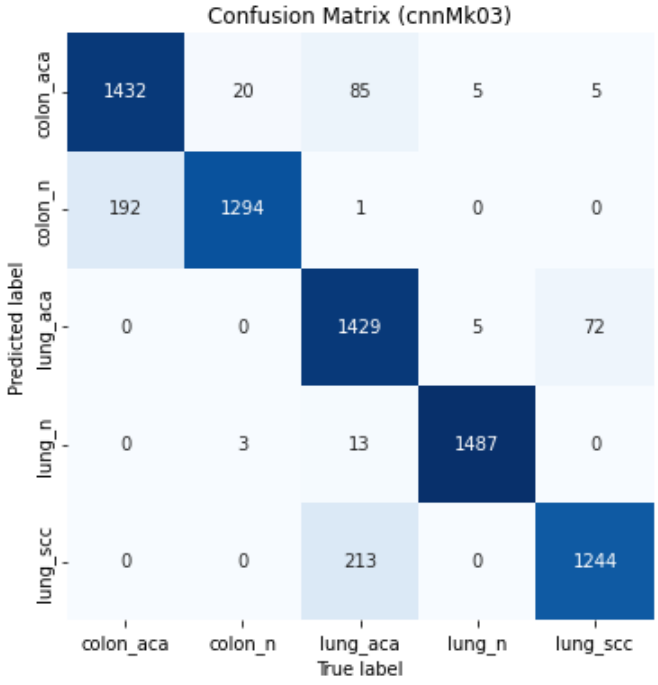
- 7 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 1.06 minutes with 92% accuracy

Model Architecture

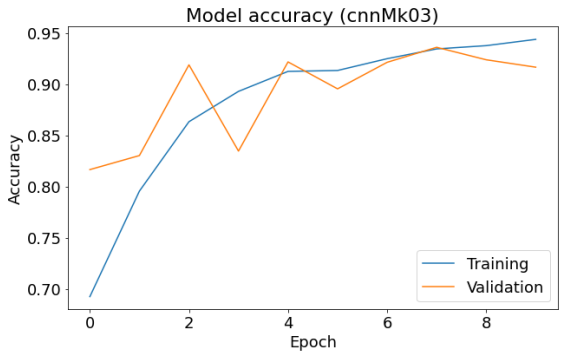
```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_45 (Conv2D)          (None, 94, 94, 64)        1792

 max_pooling2d_30 (MaxPoolin  (None, 18, 18, 64)       0
 g2D)

 conv2d_46 (Conv2D)          (None, 16, 16, 32)        18464
```

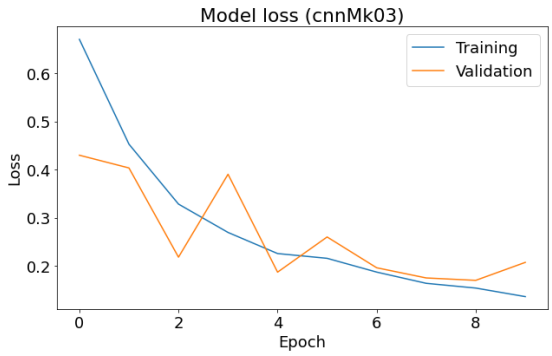
```
 max_pooling2d_31 (MaxPoolin   (None, 8, 8, 32)          0
 g2D)

 conv2d_47 (Conv2D)           (None, 6, 6, 32)          9248

 flatten_15 (Flatten)         (None, 1152)              0

 dense_18 (Dense)             (None, 5)                 5765

=================================================================
Total params: 35,269
Trainable params: 35,269
Non-trainable params: 0
```

Classification Summary Report



Visualization of Cost





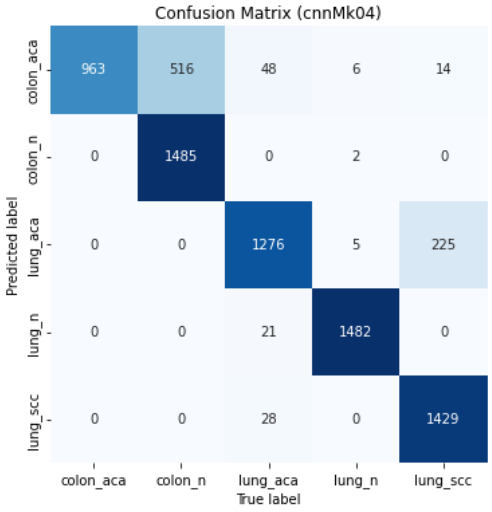
15.7. cnnMk04

- 8 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 1.24 minutes with 88% accuracy

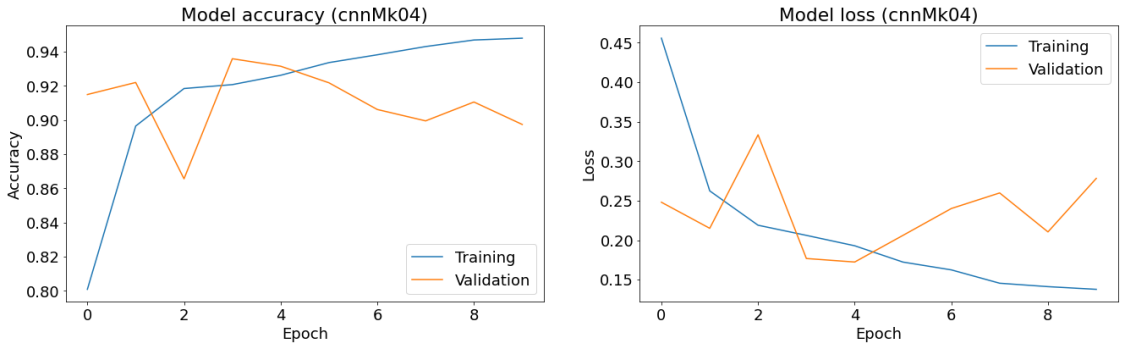
Model Architecture

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_42 (Conv2D)          (None, 94, 94, 32)        896

 max_pooling2d_28 (MaxPoolin  (None, 18, 18, 32)       0
 g2D)

 conv2d_43 (Conv2D)          (None, 16, 16, 16)        4624

 batch_normalization_13 (Bat  (None, 16, 16, 16)       64
 chNormalization)

 max_pooling2d_29 (MaxPoolin  (None, 8, 8, 16)         0
 g2D)

 conv2d_44 (Conv2D)          (None, 6, 6, 16)          2320

 flatten_14 (Flatten)        (None, 576)               0

 dense_17 (Dense)            (None, 5)                 2885

=================================================================
Total params: 10,789
Trainable params: 10,757
Non-trainable params: 32
```

Classification Summary Report



Visualization of Cost



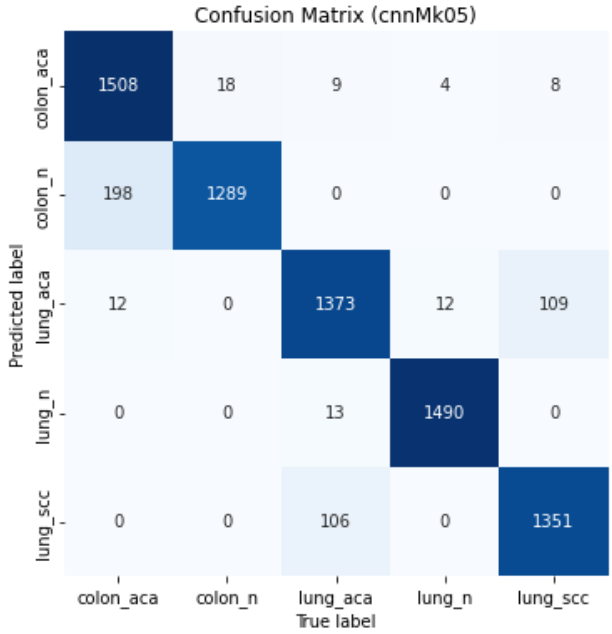
15.8. cnnMk05

- 8 Hidden Layers
- 10 epochs
- Classified 7500 unseen images in 0.7 minute with 93% accuracy

Model Architecture

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_54 (Conv2D)          (None, 94, 94, 64)        1792

 max_pooling2d_36 (MaxPoolin  (None, 18, 18, 64)       0
 g2D)

 conv2d_55 (Conv2D)          (None, 16, 16, 32)        18464

 max_pooling2d_37 (MaxPoolin  (None, 8, 8, 32)         0
 g2D)

 conv2d_56 (Conv2D)          (None, 6, 6, 32)          9248

 flatten_18 (Flatten)        (None, 1152)              0

 dense_22 (Dense)            (None, 32)                36896

 dense_23 (Dense)            (None, 5)                 165

=================================================================
Total params: 66,565
Trainable params: 66,565
Non-trainable params: 0
```

Classification Summary Report



Visualization of Cost