Classification Analysis for Cancer Detection

In Machine Learning and Content Analytics

From the MSc in Business Analytics Full Time Students:

Chalioris Stefanos-anthimos (f2821916)

&

Melekos Panagiotis-Chrysovalantis (f2822008)

2021





Table of Contents

**1. Introduction3**

1.1 Our Project3

1.2 Our Goal3

**2. Methodology4**

2.1 Data Collection4

2.2 Dataset Overview4

2.3 Data Processing7

2.4 Algorithms7

**3.** **Setup - Configuration10**

3.1 Results & Quantitative Analysis12

3.2 Qualitative Analysis21

3.3 Future Work and Final Comments21

**4. Members and Roles22**

4.1 Time Plan22

**5. Bibliography23**

**Table of Tables**

[Table 1: Classification table of MLP 13](#_Toc81765409)

[Table 2: Classification table of MLP 14](#_Toc81765410)

[Table 3: Classification table of CNN 16](#_Toc81765411)

[Table 4: Classification table of CNN 17](#_Toc81765412)

[Table 5: Classification table of Inceptionv3 19](#_Toc81765413)

[Table 6: Classification table of Inceptionv3 20](#_Toc81765414)

**1. Introduction**

This assignment is conducted by the MSc in Business Analytics of Athens University of Economics and Business Chalioris Stefanos-Anthimos, Melekos Panagiotis-Chrysovalantis.

The last couple of years technology has improved in such level that we are now able to detect the disease of cancer in humans much easier, even in earlier stages. The most notable types of cancer are: carcinoma, sarcoma, leukemia, lymphoma, multiple myeloma, melanoma, brain and spinal cord tumors, while there are some others tumors that are called benign tumors which are mostly not life threatening and they do not spread in other parts of the human body.

**1.1 Our Project**

We found a dataset in Kaggle that had 25.000 histopathological pictures with 5 classes. In our assignment we will investigate two types of carcinomas in lungs and one type of carcinoma in colon. Both organs have at least one benign tumor category that needs to be investigated, based on the dataset’s pictures.

This analysis will be conducted via Jupiter notebook in Python programming language, with the help of machine learning algorithms such as: Inceptionv3, Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN). Along with these algorithms we used an augmentation technique that helped us to achieve better results.

**1.2 Our Goal**

The purpose of this assignment is to create a model that will be able to classify successfully batches of images in the afore mentioned 5 classes. This will be used by doctors worldwide in order to assist them in cancer detection on two different organs. The higher the precision of the model the better the model will be.

The creation of our model will be a huge leap in the oncologist departments, due to the fact that the human errors will be minimized, and our model will try to make the diagnosis as accurate as possible.

**2. Methodology**

When the assignment was released, we wanted to challenge ourselves by trying a dataset that had only images. We searched many sites and encountered a lot of interesting datasets that we could try to use for this assignment. The dataset that was the most intriguing was the “Lung and Colon Cancer Histopathological Images”. We selected this dataset because we believe that the health sector can be helped greatly by the use of machine learning utilities, and we wanted this assignment to be the example that if two master students can do it then a full-fledged Data Engineer can do wonders.

After the selection of the dataset, every image was processed and augmented in order to fit the selected models from the family of Neural Networks. It is important to mention that we will present the same models without an augmentation technique to be applied in our photos, just to display the difference in our methodologies. This topic will be examined with greater detail in the following parts.

Neural Networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. From the family of artificial neural networks, 2 multi-layer perceptrons (MLPs) models, 2 Convolutional neural networks (CNNs) models and 2 Inceptionv3 models will be used. Each pair of models contains similar structures of layers with augmented-non augmented data.

**2.1 Data Collection**

The data that we used for the creation of the models and the implementation of the assignment were taken from this dataset in [Kaggle](https://www.kaggle.com/andrewmvd/lung-and-colon-cancer-histopathological-images/code). No additional data from different datasets were used.

**2.2 Dataset Overview**

The dataset contains 25.000 images for 5 classes of cancer-benign tissue for 2 different organs. More specifically, 5000 images with lung benign tissue, 5000 images with lung adenocarcinoma, 5000 images with Lung squamous cell carcinoma, 5000 images with Colon adenocarcinoma and 5000 images with Colon benign tissue. All images are 768 x 768 pixels in size and are in jpeg file format.

Images:

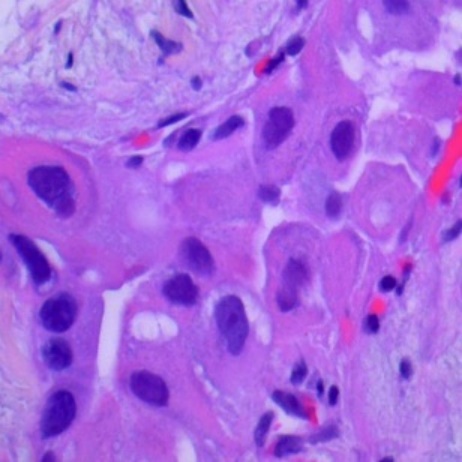


Figure 1:Lung adenocarcinoma

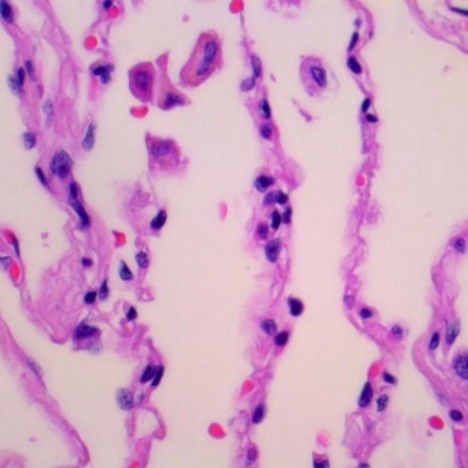


Figure 2:Lung benign tissue

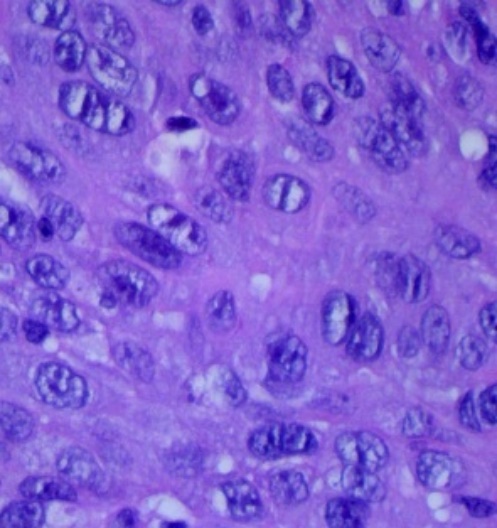


Figure 3:Lung squamous cell carcinoma

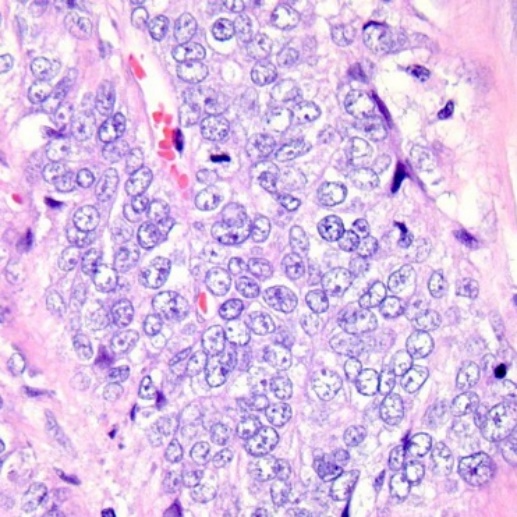


Figure 4:Colon adenocarcinoma

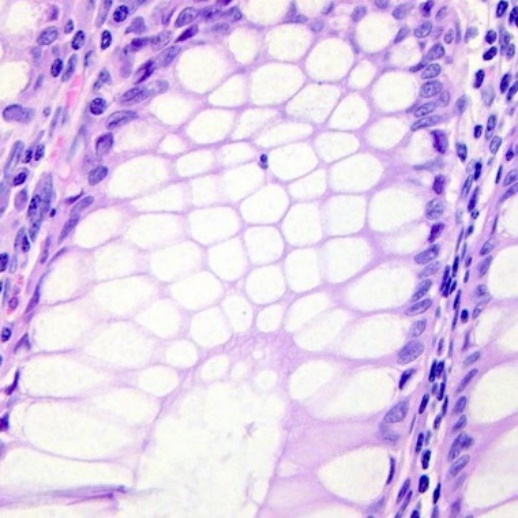


Figure 5:Colon benign tissue

**2.3 Data Processing**

For the preparation of the models, it was necessary to create the proper file directories for each photo category. During this process, we split our data into 3 different folders: train, validation and test. Training folder contained 70% of the total images, validation folder contained 20% of total images and test folder contained 10% of the total images. The relocation of photos was random, based on whether or not a variable p was between the intervals [0,0.7), [0.7,0.9), [0.9,1]. Train and validation folders were used for fitting the models and the test folder was used for the extraction of conclusions regarding the model’s accuracy, efficiency and predictivity.

Each image consists of RGB coefficients in the 0-255 interval. In order to be used as input for the models, each image was normalized from the division with 255 to an interval of [0,1]. Each image through a generator based on their folder, was given an input shape of 224 width and 224 height. Half of the models were trained with images without augmentation, apart from the normalization, and the other half were trained with augmented images with additional characteristics from an image generator. This augmentation technique is called “on the fly” which means that no original image was fed in the models. Only the augmented ones were used but only during the training process. None of the augmented images are saved or replaced any of the original images.

For the augmentation process, the pictures were rotated by 40 degrees, width shift and height shift were set to 0.2, shearing and zoom range was set to 0.2 with an additional horizontal flip and fill mode as nearest. Each one of the above parameters were applied randomly from the imageDataGenerator function.

**2.4 Algorithms**

Multi-Layer Perceptrons (MLPs)

A vast category of artificial neural networks are multi-layer perceptrons (MLPs). A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. It also uses backpropogation for training the network.

An MLP is very flexible and can be used generally to learn a mapping from inputs to outputs. This flexibility allows it to be applied to other types of data. For example, the pixels of an image can be reduced down to one long row of data and fed into a MLP. The words of a document can also be reduced to one long row of data and fed to a MLP. Even the lag observations for a time series prediction problem can be reduced to a long row of data and fed to a MLP.

In MLPs and in neural networks in general, an activation function is an important part of the machine learning procedure. It is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. Without an activation function, a neural network is a simple linear regression model Relu, softmax, sigmoid and tanh are the most common.

ReLU is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Softmax is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model. This is why softmax is sometimes referred to as a multinomial logistic regression.

A sigmoid function guarantees that in a neuron the output of this unit will always be between 0 and 1. Also, as the sigmoid is a non-linear function, the output of this unit would be a non-linear function of the weighted sum of inputs. Such a neuron that employs a sigmoid function as an activation function is termed as a sigmoid unit.

Tanh is basically a scaled sigmoid function. Historically, the tanh function became preferred over the sigmoid function as it gave better performance for multi-layer neural networks. It shares the properties with softmax.

Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

CNN’s first layer is the Convolutional layer which is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. The output is termed as a feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

The last part of the architecture is a set of Fully Connected (FC) layers which consists of the weights and biases along with the neurons and are used to connect the neurons between two different layers. This is basically a MLP structure. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

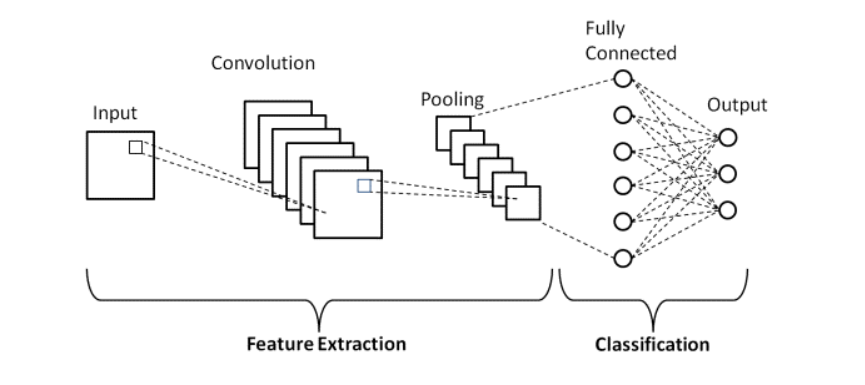


Figure 6: CNN Architecture

Inceptionv3

Inceptionv3 is the third installment of the inception networks by google in 2015. Inceptionv3 mainly focuses on burning less computational power by modifying the previous Inception architectures, both in terms of the number of parameters generated by the network and the economical cost incurred (memory and other resources).

The architecture of an Inceptionv3 network is progressively a built, step-by-step architecture. Its basic techniques involve factorized convolutions, regularization, dimension reduction, and parallelized computations.

Factorized Convolutions: this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.

Smaller convolutions: replacing bigger convolutions with smaller convolutions leads to faster training. For instance, a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3\*3 + 3\*3) parameters instead.

Asymmetric convolutions: A 3 × 3 convolution could be replaced by a 1 × 3 convolution followed by a 3 × 1 convolution. If a 3 × 3 convolution is replaced by a 2 × 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.

Auxiliary classifier: an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. It acts as a regularizer.

Grid size reduction: Grid size reduction is usually done by pooling operations.

**3. Setup - Configuration**

For the setup procedure we created a callback list that contains functions for the hyper parameter tuning that is similar for every model. EarlyStoping and ReduceLROnPlateau were used in our project. For both of them, the monitored parameter is validation loss, verbose parameter is value 1, and patience parameter is 3. EarlyStoping based on the whether or not the monitored parameter declines, stops the training of the model in the specific epoch. Value 1 for the verbose, displays a progress bar when the relative process is activated. Patience parameter sets the number of epochs to 3 where there is no improvement after which training will be stopped. Restore\_best\_weights parameter is set into TRUE. This restores model weights from the epoch with the best value of the monitored quantity. ReduceLROnPlateau Reduces learning rate when a metric has stopped improving, in this case validation loss. The factor parameter factor of value 0.2 reduces the learning rate based on the equation: new\_lr = lr \* factor. Callbacks are analyzed first because the same parameters are utilized in every model.

Multi-Layer Perceptrons (MLPs)

The first layer consists of the input parameters of the images and flatten function. The input parameters are: 224 x 224 x 3. The flatten function takes this input and creates an output from the multiplication of the sizes.

We added a second hidden layer with a dense function of 1024 neurons and an activation function called “relu”.

Adding a dropout layer, given a set of 25%, the layer randomly removes this fraction of units for reducing overfitting.

The output layer has 5 neurons, followed by a softmax activation function. The 5 neurons correspond to the 5 possible classes of the data. This output is produced in a dense layer.

After the completion of the layers, a model must be prepared for training using the .compile method. In this method we added Adam optimizer. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Since there are 5 classes, we used categorical crossentropy as a loss function and accuracy as a metric. Also, a learning rate of 0.0001 was applied.

The compile method with the same parameters was used also in the CNN models. The Inceptionv3 had also the same parameters apart from the optimizer. For the fit of the models, training and validation generators were used with 100 epochs and 20 steps per epoch, along with a callback list. This is same for every model that we created.

Convolutional Neural Networks (CNNs)

The first layer is Convolution layer with 32 filters (output space), kernel size 3x3 (height x width), an activation function and the input shape of the previous models.

Adding a second pooling layer of max pooling with a kernel size of 2x2.

A third convolutional layer is added with 64 filters, kernel size of 3x3 and activation function relu.

Adding a dropout layer with a set of 25%.

A flatten layer is placed after the dropout layer which serves as connectivity feature between convolution and dense layers. The rest of the structure is the same layers from the MLP models with the same parameters, except the input shape which was used at the start of the CNN.

Inceptionv3

Using the pretrained model from the InceptionV3 function we set as an input shape 224x224x3 for the images, include\_top parameter as False and weights as “Imagenet”. Include\_top if is set to False then the fully-connected output layers of the model used to make predictions is not loaded, allowing a new output layer to be added and trained. The weights are derived from the imagenet database with around 1 million images and 1000 classes.

The rest of the layers has the same characteristics from the previous MLP models. The only difference lies with the optimizer. In this case we used the RMSprop optimizer. RMSprop uses an adaptive learning rate instead of treating the learning rate as a hyperparameter. This means that the learning rate changes over time.

**3.1 Results & Quantitative Analysis**

Multi-Layer Perceptrons (MLPs)without Augmentation

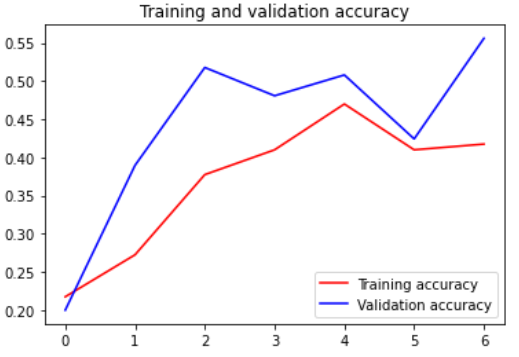
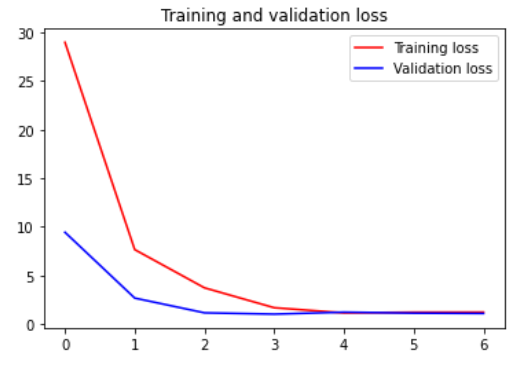


Figure 7: Training and Validation Accuracy of MLP Figure 8: Training and Validation Loss of MLP

The highest accuracy that the validation set reaches is around 55% at the 6th epoch, while the accuracy of the training set reaches 45% max value at 4th epoch. The loss for the training and validation is flattened at the fourth epoch.

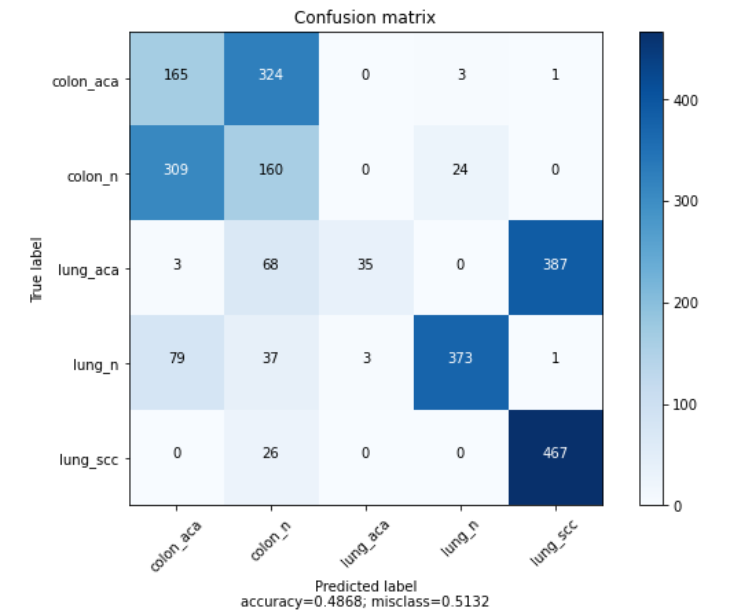


Figure 9: Confusion Matrix of MLP

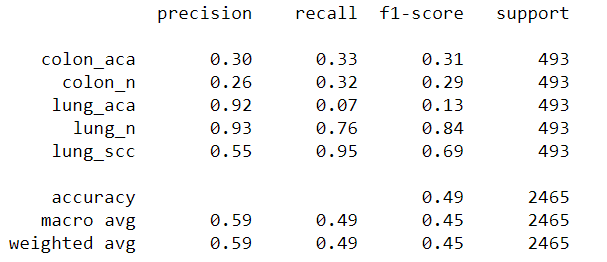


Table 1: Classification table of MLP

Τhe model’s accuracy performs at 49% for the test set with mixed results about the classification of the classes. A poor result overall.

Multi-Layer Perceptrons (MLPs)with Augmentation

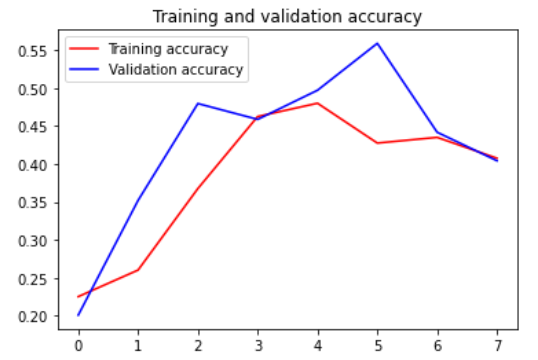
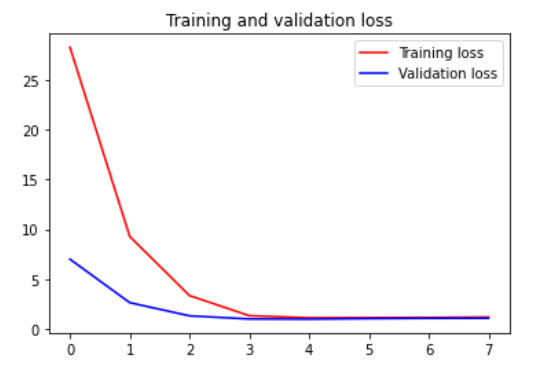


Figure 10: Training and Validation Accuracy of MLP Figure 11: Training and Validation Loss of MLP

Validation set accuracy peaks at the 5th epoch with 55%, while the training set accuracy reaches 46% at the 4th epoch. Training and validation loss flatten at the 3rd epoch. Nothing has changed with the augmented features in the photos.

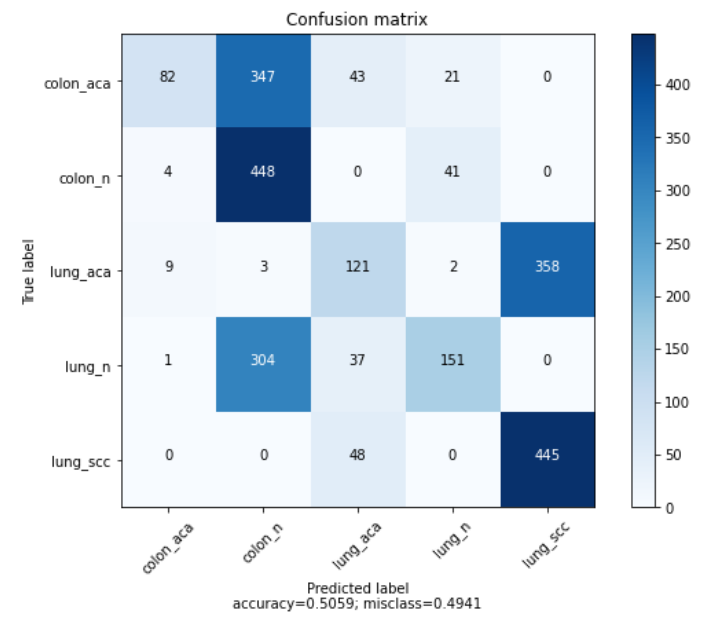


Figure 12: Confusion Matrix of MLP

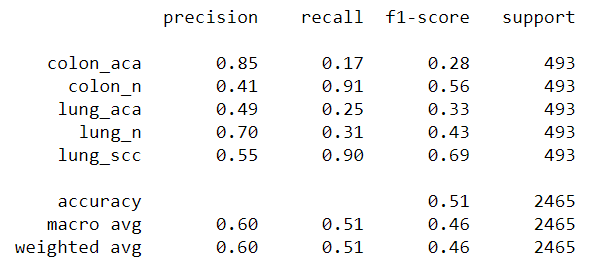


Table 2: Classification table of MLP

A slightly improved accuracy from the previous model occurred with the test set at 51% with a more balanced but still extreme predictions for the classification of the categories.

Convolutional Neural Networks (CNNs) without Augmentation

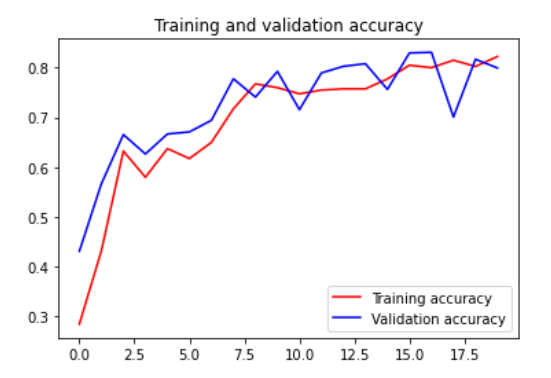
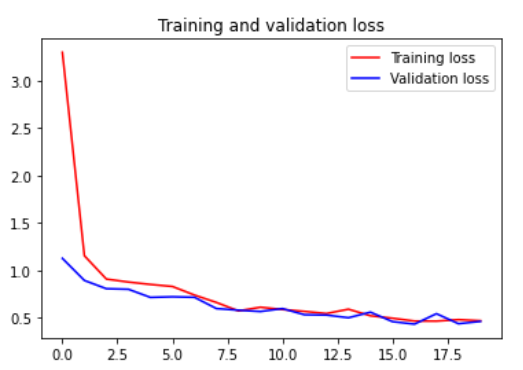


Figure 13: Training and Validation Accuracy of CNN Figure 14: Training and Validation Loss of CNN

Highest accuracy for both training and validation sets occur at 80% with the 15th epoch. Also, training and validation loss flatten at the 5th epoch.

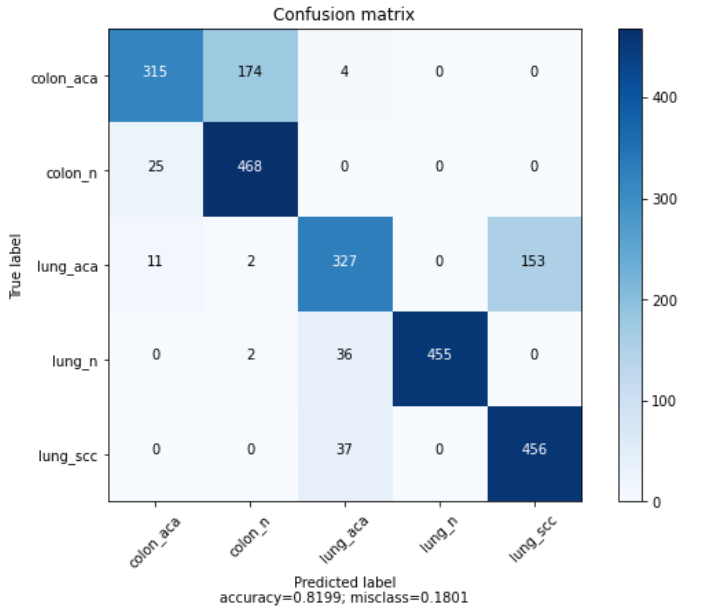


Figure 15: Confusion Matrix of CNN

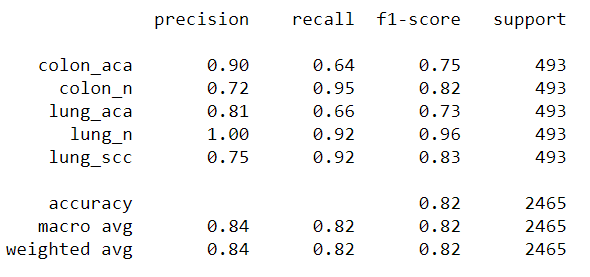


Table 3: Classification table of CNN

CNN achieves 82% at the test set accuracy with a perfect classification score at the lung\_n category.

Convolutional Neural Networks (CNNs) with augmentation

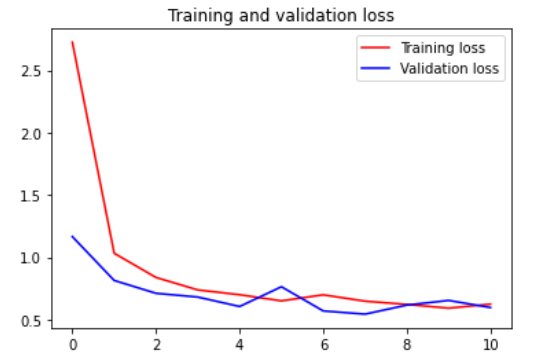
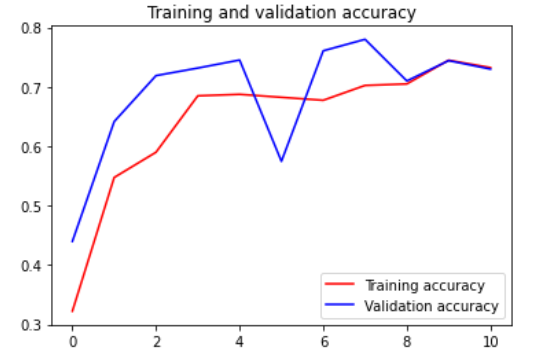


Figure 16: Training and Validation Accuracy of CNN Figure 17: Training and Validation Loss of CNN

Validation set has 78% accuracy at its peak with the 7th epoch, while the training set achieves around 73% accuracy with the 8th epoch. Validation and training loss does not appear to be reduced in a steady line near 0 up until the 10th epoch.

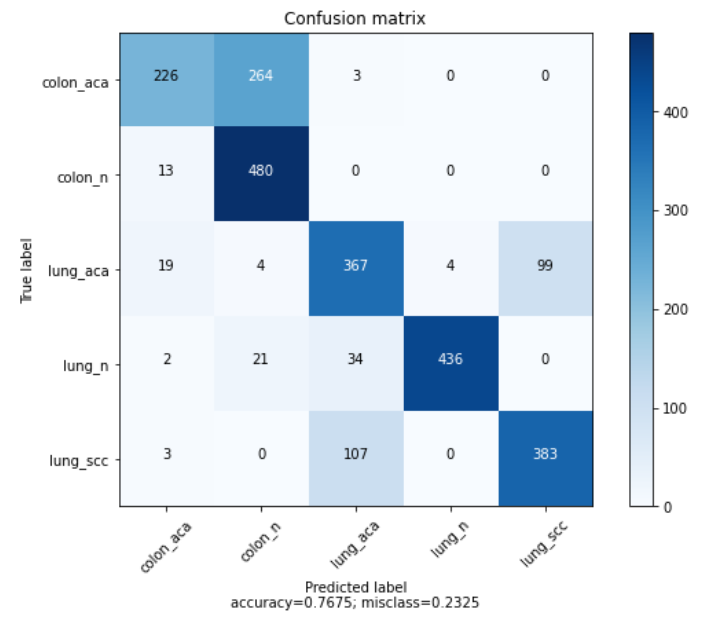


Figure 18: Confusion Matrix of CNN

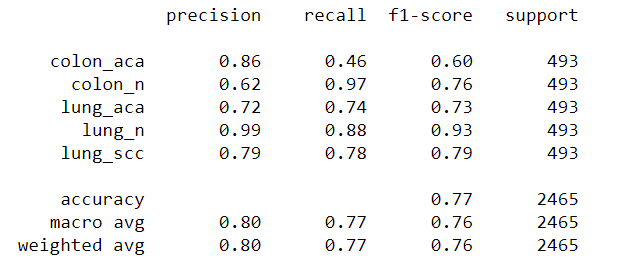


Table 4: Classification table of CNN

The accuracy at the test set reaches 77% with near perfect score again for the lung\_n category. It appears that the augmentation process deteriorated the results in this case, in contradiction with the MLP model.

Inceptionv3 without Augmentation

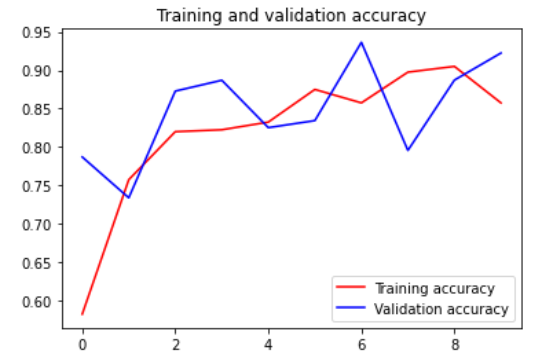
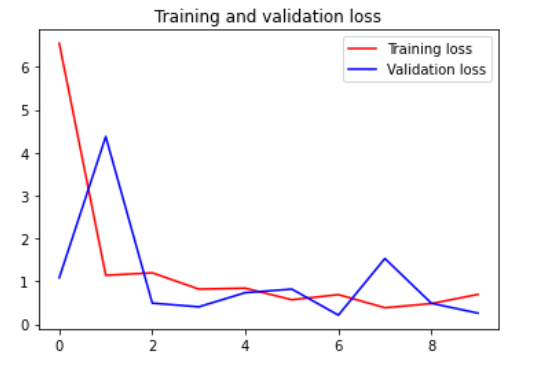


Figure 19: Training and Validation Accuracy of Inceptionv3 Figure 20: Training and Validation Loss of Inceptionv3

Training set reaches 90% accuracy at the 7th epoch, while validation set reaches 94% accuracy at the 6th epoch. Validation and training loss reach below 1 at the 8th epoch.

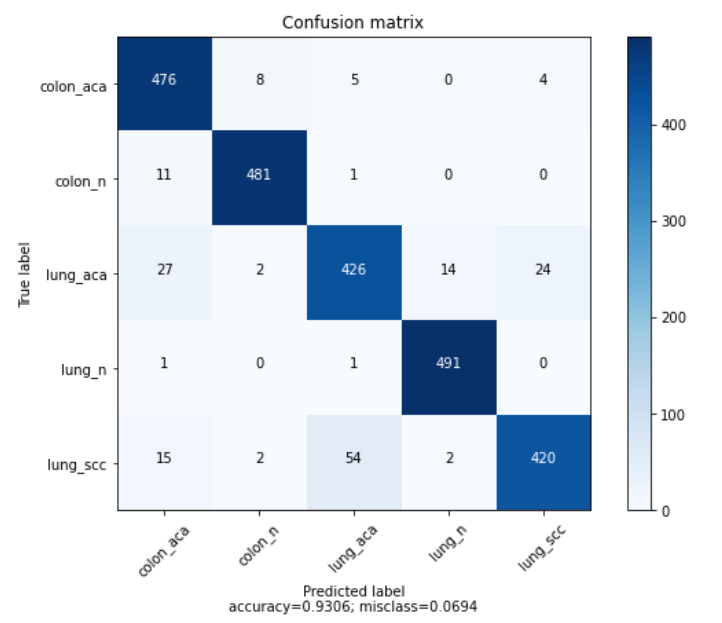


Figure 21: Confusion Matrix of Inceptionv3

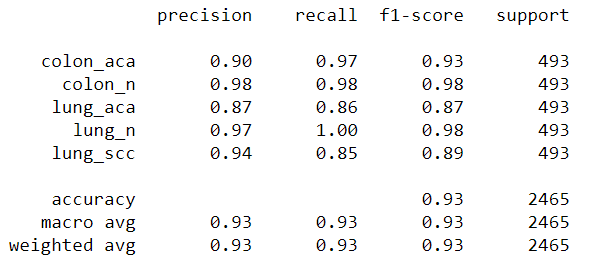


Table 5: Classification table of Inceptionv3

Inceptionv3 achieves 93% at the test set with a balanced and high prediction values for all the classes.

Inceptionv3 with Augmentation

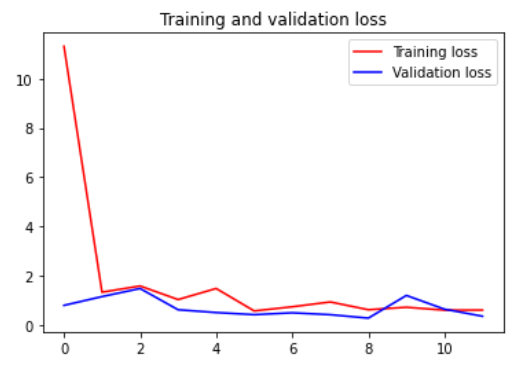
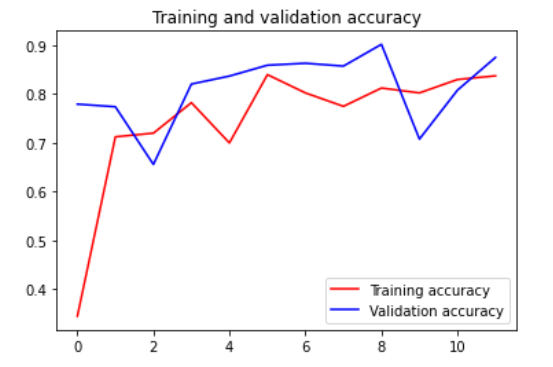


Figure 22: Training and Validation Accuracy of Inceptionv3 Figure 23: Training and Validation Loss of Inceptionv3

Validation set has 90% accuracy at the 8th, while training set achieves 82% accuracy at the 5th epoch. Training and validation loss flatten at the 5th epoch.

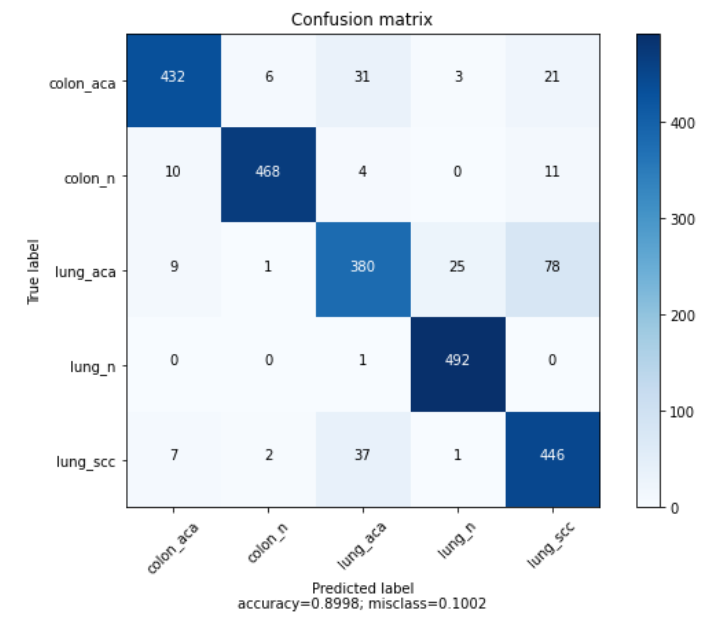


Figure 24: Confusion Matrix of Inceptionv3

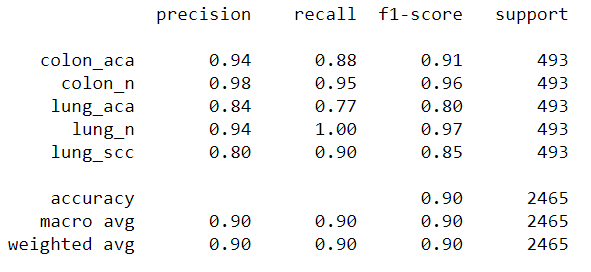


Table 6: Classification table of Inceptionv3

Accuracy for the test set is 90% with a classification precision of over 94% for the 3 classes. Inceptionv3 with augmentation has worse results from the same model without augmentation.

**3.2 Qualitative Analysis**

MLP: Due to the simplicity of the structure of the model, each step from every epoch took 3 seconds. Resulting in 480 seconds for the augmented model and 420 second for the non-augmented model

CNN: From the additional layers, along with the mlp structure, each step from every epoch took 5 seconds to complete. The first non-augmented model took 2000 seconds and the second augmented model took 1100 seconds to complete.

Inceptionv3: The inception models took the most time out of all the 3 algorithms per epoch, due to the different types of layers and due to the overall structural size. The non-augmented model took 2000 seconds with 10 seconds on average per epoch. The augmented model took 1200 seconds with the same average as the previous model.

The MLP models are highly simplistic structurally and parametrically. Due to this fact, the accuracy outputs for both validation and training set are very bad as can be seen from figure 7 and figure 10. This indicates high bias and underfitting. CNN and Inceptionv3 models have significant improvements in their structures, thus better results overall. The paradox of CNN and Inceptionv3 models is that the augmentation technique that we added seemed to worsen the results, instead of improving them. Validation sets performed better than the training sets in these models, so despite the high accuracy an assumption of bias can still be in effect. More data and a wider range of features could be helpful in the solution of this problem.

**3.3 Future Work and Final Comments**

Similar work has been conducted from various analysts on some of the classes of the same images with pretrained models such as VGG16, Inception-Resnetv2, Resnet-50 and various CNN structures. The results varied between 85% and 98% accuracy.

Concluding in our assignment, by far the best model is the inceptionv3 without any augmentation parameters with an accuracy of nearly 96%. In comparison with the second model of the inception with augmentation on the photos, this is a bizarre output, since most of the time augmentation is being used to reduce bias and variance of the model, as well as underfitting. In this case, the model has increased bias and variance without changing any of the initial parameters.

Pre-trained model EfficientNet and different MLP, CNN structures can be used to display different samples of results either with or without any implementation of augmentation techniques. Experimentation with different parameters and optimizers can also improve the performance of the models.

**4. Members and Roles**

The members of this assignment are: Chalioris Stefanos-Anthimos, Melekos Panagiotis-Chrysovalantis.

The roles of the assignment were: Communication with the professor, Preprocessing the data, Augmentation of the dataset, Creation of the Inceptionv3 model, Creation of the Multilayer Perceptron (MLP) model, Creation of the Convolutional Neural Network (CNN) model, Parameters fine tuning, running the models, Creation of the report and Creation of the presentation.

Chalioris Stefanos-Anthimos was responsible for Communication with the professor, Creation of the Inceptionv3 model, Creation of the Convolutional Neural Network (CNN) model, Parameters fine tuning, Creation of the report and Creation of the presentation.

Melekos Panagiotis-Chrysovalantis was responsible for Preprocessing the data, Augmentation of the dataset, Creation of the Multilayer Perceptron (MLP) model, running the models, Creation of the report and Creation of the presentation.

Εικόνα που περιέχει κείμενο, σταυρόλεξο, ντουλάπι

Περιγραφή που δημιουργήθηκε αυτόματα**4.1 Time Plan**

Figure 25: Time Plan

**5. Bibliography**

1. [https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image- classification-with-python-code/](https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-%20%20%20%20%20%20classification-with-python-code/)

2. <https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/?utm_source=blog&utm_medium=top4_pre-trained_image_classification_models>

3. <https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8>

4. <https://medium.com/@sidathasiri/building-a-convolutional-neural-network-for-image-classification-with-tensorflow-f1f2f56bd83b>

5. <https://wiki.pathmind.com/multilayer-perceptron>

6. <https://www.tensorflow.org/tutorials/images/classification>

7. <https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/>

8. <https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c>

9. <https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be>

10. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

11. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

12. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

13. <https://arxiv.org/pdf/1512.00567v3.pdf>

14. <https://keras.io/api/applications/inceptionv3/>