Deep Learning

880663-M-6

Assignment

Using Deep Learning to Perform Multi-Class Classification on the Lung and Colon Cancer Histopathological Image Dataset (LC25000)

Report by:

Leonard Sugg (2039830)

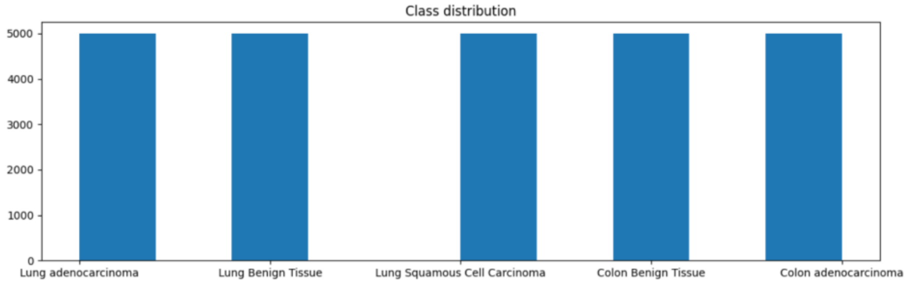
March 2024

1. Problem Definition

For this image classification assignment 25000 lung and colon images were used to train a CNN model to detect and classify cancer (Borkowski et al., 2019). There are 5 classes, two of them are benign tissue and three of them are cancerous tissue. The classes are 'Lung Benign Tissue', 'Colon Benign Tissue', 'Lung adenocarcinoma', 'Lung Squamous Cell Carcinoma', and 'Colon adenocarcinoma'. About 97% of the images are augmented from the original 750 images. The classes are balanced, before and after the augmentation.

2. Exploratory Data Analysis

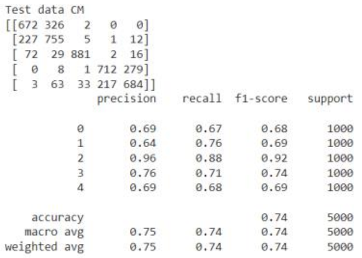
After running the provided code, the images were resized from 768 x 768 pixels to 120 x 120 pixels. For the EDA the class distribution and 15 randomly selected images were visualized.



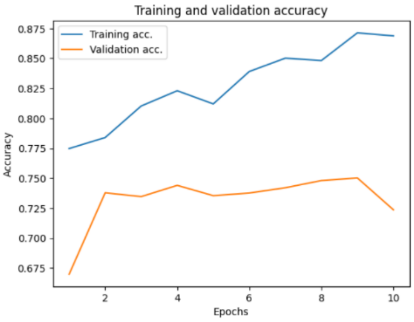
The preprocessing started by onehot encoding the image labels. Then the images were split up into stratified train (60%), validation (20%), and test (20%) sets with the random_state argument set to 42. Furthermore, the ROC function uses an one vs rest approach, meaning that the selected class will be evaluated as the positive class against the remaining classes. For this assignment the following common ROC AUC thresholds are used: 0.5 is considered bad, 0.7 to 0.8 acceptable, 0.8 to 0.9 is excellent, and above 0.9 is outstanding (Mandrekar, 2010).

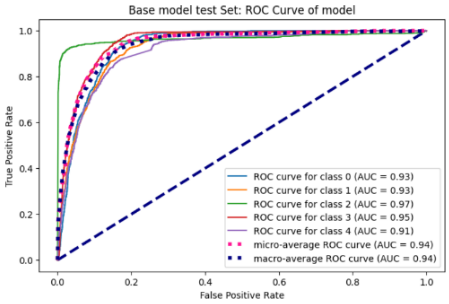
3. Results of the Baseline Model

The baseline model was strictly specified in the assignment and thus no architecture will be mentioned here. Also since the classes are balanced, the accuracy from the test set can reliably be used as evaluation metric, which was 74%. Below are the graphs for the accuracy. In the graph it is clearly visible that the accuracy of validation and training do diverge, to be specific, the base model seems to perform worse as the number of epochs increase. The F1 scores show that the base model is notably better at recognizing benign lung tissue (class 2) than the other tissue types. The Roc curves for the validation and test images are outstanding. In this report only the ROC and confusion matrix for the test data are shown, however, the visualizations for the validation data are in the notebook.



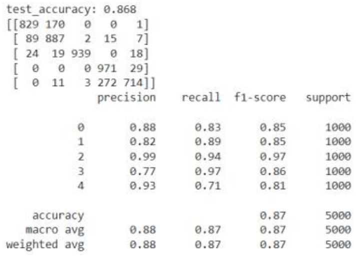
```
Test data CM
[[672 326   2   0   0]
 [227 755   5   1  12]
 [ 72  29 881   2  16]
 [  0   8   1 712 279]
 [  3  63  33 217 684]]
              precision    recall  f1-score   support

           0       0.69      0.67      0.68      1000
           1       0.64      0.76      0.69      1000
           2       0.96      0.88      0.92      1000
           3       0.76      0.71      0.74      1000
           4       0.69      0.68      0.69      1000

    accuracy                           0.74      5000
   macro avg       0.75      0.74      0.74      5000
weighted avg       0.75      0.74      0.74      5000
```



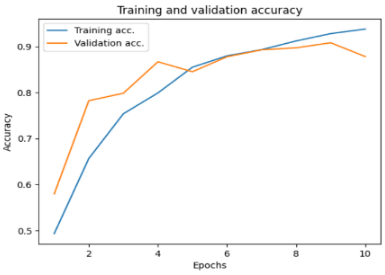


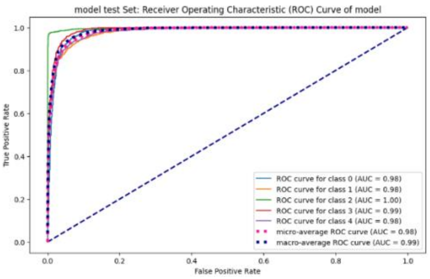
4. Improved (Fine-tuned) Model and Its Results

The first change to the base model was implementing an additional 3x3 filter with a 32 output, which is followed by an additional 2x2 max pooling layer. Additionally, the kernel size of the first filter was increased to 5x5, since this allows for considering bigger patterns in the images. Altering the kernel size and filter amount resulted in an accuracy of 87%. While the f1 score remarkably increased for every class, the second class had a smaller but steady increase to 0.97. The ROC curves were all at least 0.98, thus outstanding. The accuracy between training and test set did mostly converge, however, the validation accuracy did decrease in the last epoch.

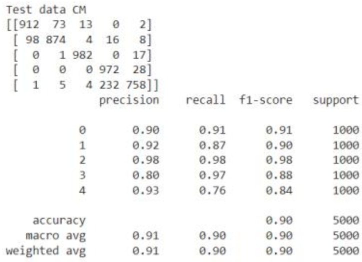


```
test_accuracy: 0.868
[[829 170   0   0   1]
 [ 89 887   2  15   7]
 [ 24  19 939   0  18]
 [  0   0   0 971  29]
 [  0  11   3 272 714]]
              precision    recall  f1-score   support

           0       0.88      0.83      0.85      1000
           1       0.82      0.89      0.85      1000
           2       0.99      0.94      0.97      1000
           3       0.77      0.97      0.86      1000
           4       0.93      0.71      0.81      1000

    accuracy                           0.87      5000
   macro avg       0.88      0.87      0.87      5000
weighted avg       0.88      0.87      0.87      5000
```

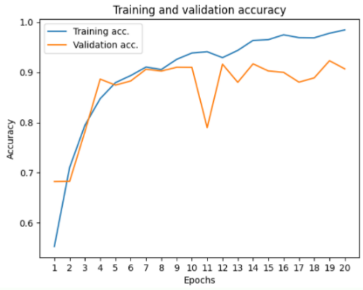


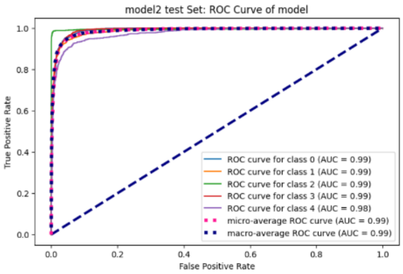


The second change of the model was doubling the epochs from 10 to 20 to investigate whether the accuracy drop on the last two epochs was a sign of overfitting or a temporary issue. As shown in the graphs below, doubling the epochs further increased the accuracy to 90%. The f1 scores did increase for every class. As visible in the graph, the accuracy of the training data and validation data were converging until 9 epochs and then slowly diverging for the remaining epochs. Based on this observation, the model seems to slightly overfit (Brownlee, 2020).

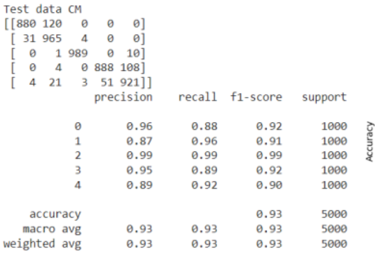


```
Test data CM
[[912  73  13   0   2]
 [ 98 874   4  16   8]
 [  0   1 982   0  17]
 [  0   0   0 972  28]
 [  1   5   4 232 758]]
              precision    recall  f1-score   support

           0       0.90      0.91      0.91      1000
           1       0.92      0.87      0.90      1000
           2       0.98      0.98      0.98      1000
           3       0.80      0.97      0.88      1000
           4       0.93      0.76      0.84      1000

    accuracy                           0.90      5000
   macro avg       0.91      0.90      0.90      5000
weighted avg       0.91      0.90      0.90      5000
```

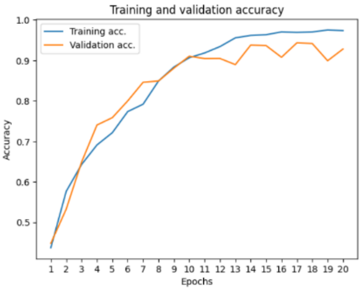


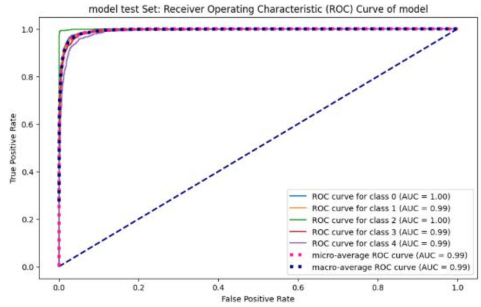


The third change of the model, thus was to introduce a measure against the hypothesized overfitting: A dropout layer. The dropout layer was set to a 25% probability, thus blocking 25% of the previous 128 neurons from contributing with their weights. The addition of the dropout layer increased the accuracy slightly to 93%. The f1 scores of the classes now range from 0.9 (lung squamous cell carcinoma) to 0.99 (lung benign tissue). The ROC values range from excellent 0.99 to 1 for this model. The training and validation accuracy did diverge, however, the validation accuracy seemed to periodically increase and decrease to the same maxima and minima from the 14th epoch onwards.

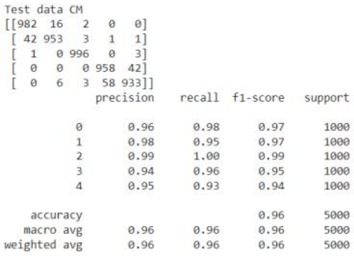


```
Test data CM
[[880 120   0   0   0]
 [ 31 965   4   0   0]
 [  0   1 989   0  10]
 [  0   4   0 888 108]
 [  4  21   3  51 921]]
              precision    recall  f1-score   support

           0       0.96      0.88      0.92      1000
           1       0.87      0.96      0.91      1000
           2       0.99      0.99      0.99      1000
           3       0.95      0.89      0.92      1000
           4       0.89      0.92      0.90      1000

    accuracy                           0.93      5000
   macro avg       0.93      0.93      0.93      5000
weighted avg       0.93      0.93      0.93      5000
```

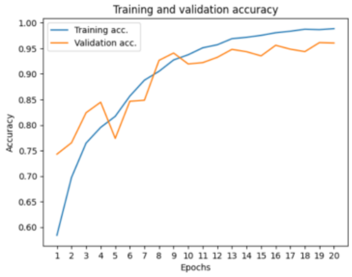


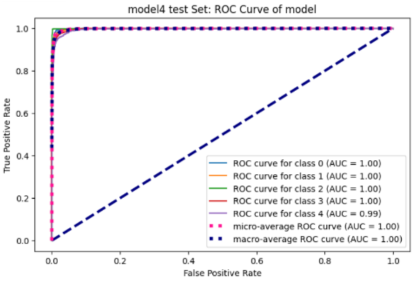


The fourth change of the model was to use adamax as optimizer instead of adam, since adamax is the extended version of adam and more robust against noisy gradients (Yi et al., 2020). Implementing adamax as optimizer increased the model accuracy to 96% and all f1 scores surpassed 0.94. Furthermore, all ROC values were 1 except 0.99 for class 4 (lung adenocarcinoma). As seen in the graph, the validation accuracy plateaued around 95% for the last eight epochs while the training accuracy increased.

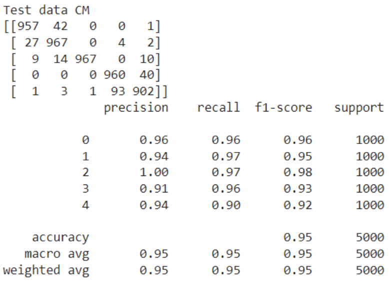


```
Test data CM
[[982  16   2   0   0]
 [ 42 953   3   1   1]
 [  1   0 996   0   3]
 [  0   0   0 958  42]
 [  0   6   3  58 933]]
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1000
           1       0.98      0.95      0.97      1000
           2       0.99      1.00      0.99      1000
           3       0.94      0.96      0.95      1000
           4       0.95      0.93      0.94      1000

    accuracy                           0.96      5000
   macro avg       0.96      0.96      0.96      5000
weighted avg       0.96      0.96      0.96      5000
```

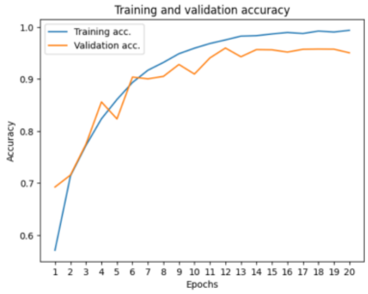


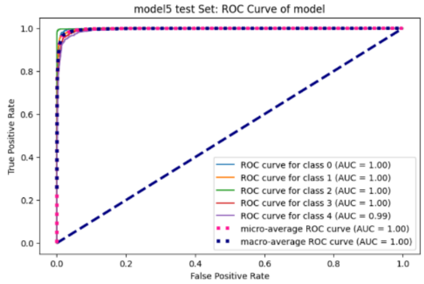


The fifth change of the model was to half the batch size from 32 to 16 to get a slightly better generalizability since more noise is introduced (Devansh, 2022). However, there is a higher risk of getting stuck at a local minimun. The accuracy stayed at 95%, however, some f1 scores slightly dropped. The model did stop improving around the 12th epoch, while the training accuracy improved to 99% accuracy. The ROC values did not change from the previous model. Since decreasing the batch size did not improve the accuracy of the model, 32 will be used as batch size for the next improvement.

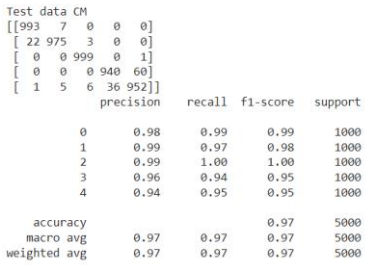


```
Test data CM
[[957  42   0   0   1]
 [ 27 967   0   4   2]
 [  9  14 967   0  10]
 [  0   0   0 960  40]
 [  1   3   1  93 902]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1000
           1       0.94      0.97      0.95      1000
           2       1.00      0.97      0.98      1000
           3       0.91      0.96      0.93      1000
           4       0.94      0.90      0.92      1000

    accuracy                           0.95      5000
   macro avg       0.95      0.95      0.95      5000
weighted avg       0.95      0.95      0.95      5000
```

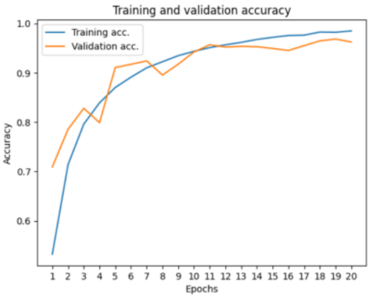


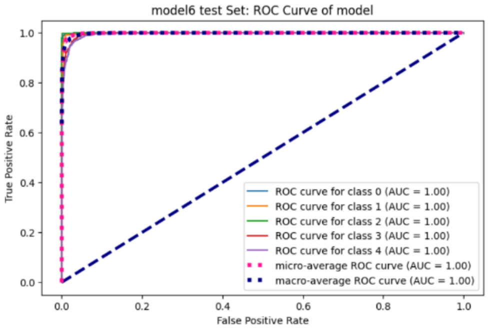


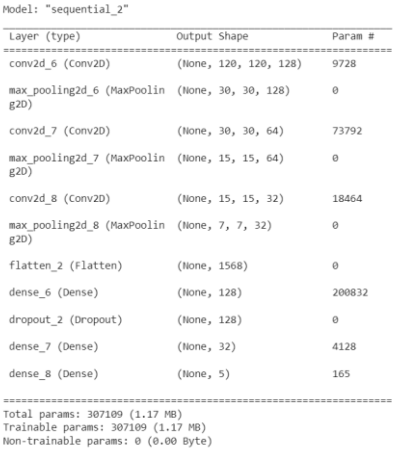
The sixth and final change of the model was to increase the pooling size of the first max pooling layer because it increases the downsampling, thus less details are kept. With 97% accuracy this was the best model out of all enhanced models, thus the architecture is shown. Furthermore, all f1 scores did increase with the lowest being 0.95 and the highest being 1.



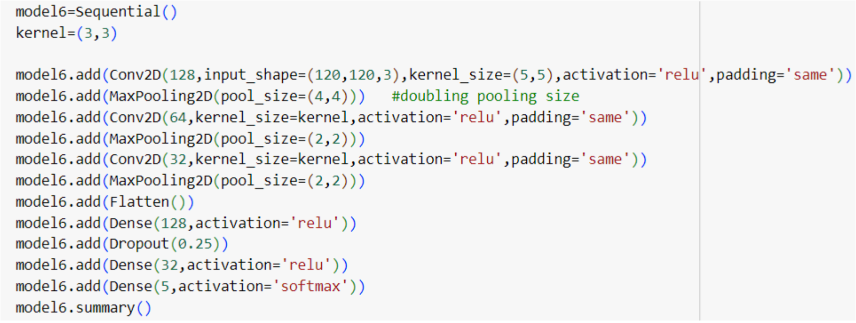
```
Test data CM
[[993   7   0   0   0]
 [ 22 975   3   0   0]
 [  0   0 999   0   1]
 [  0   0   0 940  60]
 [  1   5   6  36 952]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1000
           1       0.99      0.97      0.98      1000
           2       0.99      1.00      1.00      1000
           3       0.96      0.94      0.95      1000
           4       0.94      0.95      0.95      1000

    accuracy                           0.97      5000
   macro avg       0.97      0.97      0.97      5000
weighted avg       0.97      0.97      0.97      5000
```







```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 120, 120, 128)     9728

 max_pooling2d_6 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 conv2d_7 (Conv2D)           (None, 30, 30, 64)        73792

 max_pooling2d_7 (MaxPoolin  (None, 15, 15, 64)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 15, 15, 32)        18464

 max_pooling2d_8 (MaxPoolin  (None, 7, 7, 32)          0
 g2D)

 flatten_2 (Flatten)         (None, 1568)              0

 dense_6 (Dense)             (None, 128)               200832

 dropout_2 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 32)                4128

 dense_8 (Dense)             (None, 5)                 165

=================================================================
Total params: 307109 (1.17 MB)
Trainable params: 307109 (1.17 MB)
Non-trainable params: 0 (0.00 Byte)
```



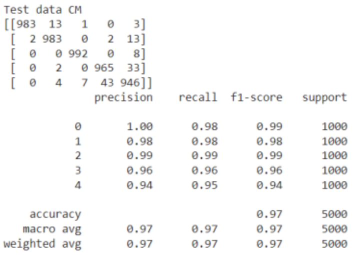
```python
model6=Sequential()
kernel=(3,3)

model6.add(Conv2D(128,input_shape=(120,120,3),kernel_size=(5,5),activation='relu',padding='same'))
model6.add(MaxPooling2D(pool_size=(4,4)))   #doubling pooling size
model6.add(Conv2D(64,kernel_size=kernel,activation='relu',padding='same'))
model6.add(MaxPooling2D(pool_size=(2,2)))
model6.add(Conv2D(32,kernel_size=kernel,activation='relu',padding='same'))
model6.add(MaxPooling2D(pool_size=(2,2)))
model6.add(Flatten())
model6.add(Dense(128,activation='relu'))
model6.add(Dropout(0.25))
model6.add(Dense(32,activation='relu'))
model6.add(Dense(5,activation='softmax'))
model6.summary()
```

5. Transfer Learning Model and Its Results

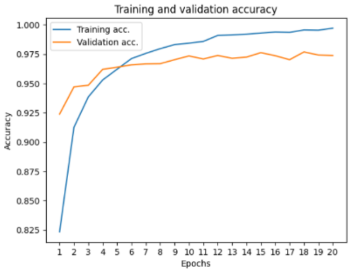
For the transfer learning model VGG16 was selected. VGG16 was made accessible to the public after its successful participation in the 2014 ImageNet competition, where it won the classification and localization categories (Simonyan & Zisserman, 2014). It is a very deep neural network with 16 weighted layers and 3x3 convolutional layers (Simonyan & Zisserman, 2014).

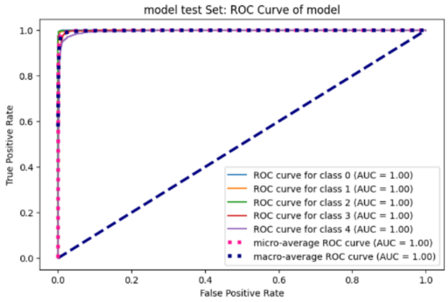
The model was loaded and the layers were deactivated for training. New trainable, layers were added and the model was trained. The added architecture consisted of a flatten layer, followed by the neurons of the last and best performing enhanced model. The compiler arguments were the same as for the enhanced model.

As seen in the graph the validation accuracy mostly stagnated around 97% from the ninth epoch onwards, while the training accuracy slowly approached the 100%. The f1 scores are all above 0.94 and the ROC values are outstanding.



```
Test data CM
[[983  13   1   0   3]
 [  2 983   0   2  13]
 [  0   0 992   0   8]
 [  0   2   0 965  33]
 [  0   4   7  43 946]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1000
           1       0.98      0.98      0.98      1000
           2       0.99      0.99      0.99      1000
           3       0.96      0.96      0.96      1000
           4       0.94      0.95      0.94      1000

    accuracy                           0.97      5000
   macro avg       0.97      0.97      0.97      5000
weighted avg       0.97      0.97      0.97      5000
```





6. Discussion

While the baseline model was doing well with 74% accuracy, increasing filter size and amount improved the performance significantly to 87% accuracy. This spike in accuracy is hypothesized to be due to the detection of bigger patterns but it also increased the computational cost. Doubling the epochs in the next step allowed the weights to be better adjusted leading to 90% accuracy. Since the model seemed overfit, a dropout layer was added as a computationally cheap form of regularization, leading to a 93% accuracy. After trying the adamax optimizer instead of adam, the accuracy increased to 96%. This could be due to adamax being less sensitive to gradient noise. With adamax the training accuracy did further increase while the validation accuracy stagnated around 95% for several epochs, indicating room for further improvement regarding generalizability. Thus, the next change was to decrease the batch sizes, which can lead to more noise in the training data and thus better generalizability. Unfortunately, this did not work as intended as the accuracy slightly declined to 95%. The next step in improving the model therefore was done with the batch size of 32 again.

Last but not least, the pooling size of the first max pooling layer was doubled to increase the down sampling and allowing for detection of bigger patterns. This improved the accuracy to 97%, which is the same accuracy as the transfer model. However, the transfer model had better training accuracy and required less epochs to train, thus, there seem to be more options to improve it. To further improve the model it should first be assessed on new datasets. Furthermore, it should be tested with normalized images as well as non-normalized images. In case it still performs as well, adding more epochs and slightly more regularization could help to further improve it. Another way could be to introduce momentum to the optimizer.

7. References

Borkowski, A. A., Bui, M. M., Thomas, L. B., Wilson, C. P., DeLand, L. A., & Mastorides, S. M. (2019). *LC25000 lung and colon histopathological image dataset*. Academic Torrents. https://academictorrents.com/details/7a638ed187a6180fd6e464b3666a6ea0499af4af

Brownlee, J. (2020, November 26). *How to identify overfitting machine learning models in Scikit-Learn*. MachineLearningMastery.com. https://machinelearningmastery.com/overfitting-machine-learning-models/

Devansh. (2022, January 8). *Why small batch sizes lead to greater generalization in deep learning*. Medium. https://medium.com/geekculture/why-small-batch-sizes-lead-to-greater-generalization-in-deep-learning-a00a32251a4f

Mandrekar, J. N. (2010). Receiver operating characteristic curve in diagnostic test assessment. *Journal of Thoracic Oncology*, *5*(9), 1315–1316. https://doi.org/10.1097/jto.0b013e3181ec173d

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Vision and Pattern Recognition*. https://doi.org/ https://doi.org/10.48550/arXiv.1409.1556

Yi, D., Ahn, J., & Ji, S. (2020). An effective optimization method for machine learning based on adam. *Applied Sciences*, *10*(3), 1073. https://doi.org/10.3390/app10031073