**Q1**

**Q2**

**2(a)**

Here, as we do not have a specific learning rate requirement, we use learning\_rate\_schedule\_exponential\_decay() and set initial\_learning\_rate to 0.001. Then consider following different filter numbers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **lr\_initial** | **Filter** | **val\_loss** | **val\_acc** | **test\_loss** | **test\_acc** |
| 0.001 | c(8, 16) | 0.3078 | 0.8929 | 0.3195 | 0.8750 |
| 0.001 | c(16, 32) | 0.3207 | 0.9127 | 0.3075 | 0.9286 |
| 0.001 | c(8, 16, 32) | 0.3532 | 0.8770 | 0.3890 | 0.8214 |
| 0.001 | c(16, 32, 64) | 0.5048 | 0.8135 | 0.5017 | 0.8750 |
| 0.001 | c(8, 16, 32, 64) | 0.3330 | 0.8611 | 0.4089 | 0.8036 |
| 0.001 | c(16, 32, 64, 128) | 0.1675 | 0.9365 | 0.2483 | 0.9464 |
| 0.001 | c(32, 64, 128, 256) | 0.2123 | 0.9167 | 0.2645 | 0.9464 |
| 0.001 | c(8, 16, 32, 64, 128) | 0.2863 | 0.8929 | 0.3229 | 0.9107 |
| 0.001 | c(8, 16, 32, 64, 128, 256) | 0.2656 | 0.8929 | 0.3304 | 0.8929 |

From the above results, it is observed that filter sequences c(16, 32, 64, 128) and c(32, 64, 128, 256) exhibit superior performance with an accuracy of 0.9464. Moving on to 2(b), we also consider filter sequences c(16, 32) and c(8, 16, 32, 64, 128) as they demonstrate commendable accuracy.

**2(b)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| lr\_initial | filter | rotation | val\_loss | val\_acc | test\_loss | test\_acc |
| 0.001 | c(16, 32) | T | 0.2531 | 0.9286 | 0.2007 | 0.9643 |
| 0.001 | c(16, 32, 64, 128) | T | 0.3518 | 0.8810 | 0.2991 | 0.9107 |
| 0.001 | c(32, 64, 128, 256) | T | 0.4650 | 0.8571 | 0.4813 | 0.8214 |
| 0.001 | c(8, 16, 32, 64, 128) | T | 0.2729 | 0.9286 | 0.2159 | 0.9643 |

Here, factor = 0.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| lr\_initial | filter | flipping | val\_loss | val\_acc | test\_los | test\_ac |
| 0.001 | c(16, 32) | T | 0.3506 | 0.8452 | 0.5536 | 0.7143 |
| 0.001 | c(16, 32, 64, 128) | T | 0.5232 | 0.7381 | 0.5771 | 0.6786 |
| 0.001 | c(32, 64, 128, 256) | T | 0.2013 | 0.9444 | 0.3463 | 0.8214 |
| 0.001 | c(8, 16, 32, 64, 128) | T | 0.3652 | 0.8730 | 0.4586 | 0.8214 |

**2(c)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| lr\_initial | filter |  | val\_loss | val\_acc | test\_loss | test\_acc |
| 0.001 | c(16, 32) | batch\_norm | 1.0796 | 0.4960 | 0.6892 | 0.5357 |
| 0.001 | c(16, 32, 64, 128) | batch\_norm | 0.8139 | 0.5040 | 0.6911 | 0.4643 |
| 0.001 | c(32, 64, 128, 256) | batch\_norm | 0.6849 | 0.6508 | 0.6694 | 0.3571 |
| 0.001 | c(8, 16, 32, 64, 128) | batch\_norm | 1.1362 | 0.5040 | 0.7172 | 0.4643 |
| 0.001 | c(16, 32) | sep\_conv | 0.3154 | 0.9444 | 0.2650 | 0.9464 |
| 0.001 | c(16, 32, 64, 128) | sep\_conv | 0.6948 | 0.5040 | 0.6965 | 0.4643 |
| 0.001 | c(32, 64, 128, 256) | sep\_conv | 0.6939 | 0.5040 | 0.6937 | 0.4643 |
| 0.001 | c(8, 16, 32, 64, 128) | sep\_conv | 0.6931 | 0.5040 | 0.6938 | 0.4643 |
| 0.001 | c(16, 32) | resid\_conn | 0.6021 | 0.7302 | 0.5768 | 0.7143 |
| 0.001 | c(16, 32, 64, 128) | resid\_conn | 0.4415 | 0.8056 | 0.3955 | 0.8571 |
| 0.001 | c(32, 64, 128, 256) | resid\_conn | 0.1950 | 0.9524 | 0.1437 | 0.9643 |
| 0.001 | c(8, 16, 32, 64, 128) | resid\_conn | 0.4551 | 0.8294 | 0.4196 | 0.8571 |

**2(d)**

From the above result, choose a model with filter c(16,32), plus rotation to Provide a visualization of the interesting activation layers. The following figure is a summary of this model.

A screenshot of a computer program

Description automatically generated

**Displaying the test picture:**

A green leaf with black spots

Description automatically generated

**Visualizing every channel in every intermediate activation:**

**A green squares with a leaf

Description automatically generated**

A green squares with a leaf

Description automatically generated

**A green squares with a leaf pattern

Description automatically generated**

**A green squares with a leaf pattern

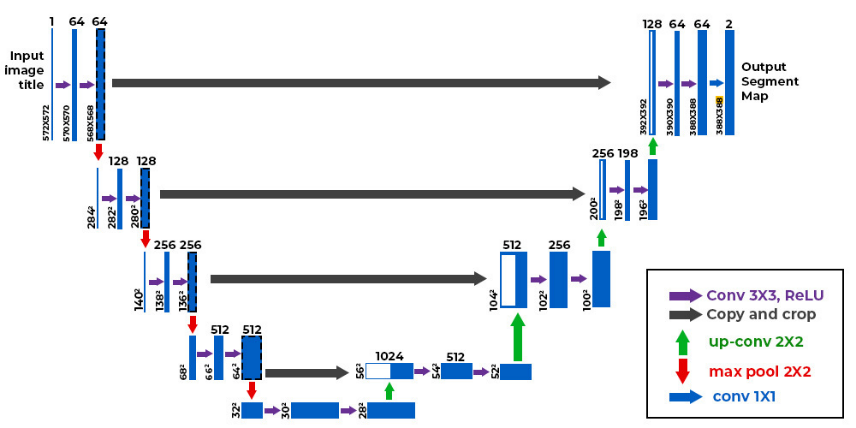
Description automatically generated**

**2(e)**

**Q3**

**(a)**

**Model Overview**

* We utilized U-Net for the segmentation of neither (background), lung, or airway.
* Input: 128 x 128 x 1 input images (rescaled and greyscale).
  + Trained on 26\*200 images (sampling 200 slices from each CT)
* Model: U-Net Architecture
* Settings:
  + Adam optimizer, learning rate = 0.001
  + Categorical cross-entropy with 3 categories (neither/lung/airway)
  + Accuracy metric
  + 10 epochs, 32 batch size

**Model Training Loss/Accuracy**

* Loss: 0.0345
* Accuracy: 0.9877

**Model Validation Loss/Accuracy**

* Loss: 0.0385
* Accuracy: 0.9862

**Model Test Loss/Accuracy (20% split)**

* Loss: 0.0369
* Accuracy: 0.9866

**(b)**

**Image 27 Loss/Accuracy**

* Loss: 0.0495
* Accuracy: 0.9817

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Ground Truth |  |
|  |  | Neither | Lung | Airway |
|  | Neither | 2830427 | 46265 | 562 |
| Prediction | Lung | 10528 | 383138 | 3 |
|  | Airway | 1217 | 1469 | 3191 |

The U-Net model provided good results: high accuracy across training, validation, and test datasets. However, because of the imbalanceness of our data, classifying them as neither or lung achieves a high accuracy score. From the confusion matrix, we noticed that the prediction accuracy for the airway is not consistent with the overall 98% accuracy.

To address this issue and for future investigations, it would be nice to use alternative metrics, such as the multiclass-F1 score, and also put weights to minor categories (airway in our case) to provide more accurate results.