Q1

Before delving into the neural network model, we focus on preprocessing the data to achieve high accuracy—an unattainable score using the original dataset alone. We extract and sample small images from the dataset, utilizing these image snippets that potentially contain individual letters or at least a few letters. This approach allows the neural network model to capture distinct letter shapes across various languages. It's important to note that during the splicing of small images from the original dataset, some snippets may be entirely white and lacking any letters. Consequently, we exclude images primarily composed of whitespace.

1(a) a basic convnet

model	layers	filters	learning rate	best test acc	augmented
1	2	8,16	0.001	0.8976	
2	3	8,16,32	0.001	0.9659	
3	5	8,16,32,64,128	0.001	0.9790	
4	2	8,16	0.01	0.8976	
5	3	8,16,32	0.01	0.5774	
6	5	8,16,32,64,128	0.01	0.5774	
7	2	8*2,16*2	0.001	0.9239	
8	3	8*2,16*2,32*2	0.001	0.9764	
9	5	8*2,16*2,32*2,64*2,128*2	0.001	0.9816	
10	2	8*2,16*2	0.01	0.9344	
11	3	8*2,16*2,32*	0.01	0.5774	
12	5	8*2,16*2,32*2,64*2,128*2	0.01	0.5774	

Using identical layers and filters with different learning rates of 0.001 and 0.01, it becomes evident that a learning rate of 0.001 performs better. Model 9 is the best model above. To test whether data augmentation can improve performance, here we use model 9 and model 10 to see the results.

1(b) data augmentation

model	layers	filters	learning rate	best test acc	augmented
13	5	8*2,16*2,32*2,64*2,128*2	0.001	0.9764	F = T
14	5	8*2,16*2,32*2,64*2,128*2	0.001	0.9711	R = T
15	5	8*2,16*2,32*2,64*2,128*2	0.001	0.9633	both = T
16	2	8*2,16*2	0.01	0.8635	F = T
17	2	8*2,16*2	0.01	0.9055	R = T
18	2	8*2,16*2	0.01	0.8478	both = T

Here, where F represents flip and R represents rotation, it is evident that both flipping and rotating adversely impact performance compared to the original mode. Furthermore, combining both operations results in an even poorer outcome.

1(c) residual connections

model	layers	filters	learning rate	best test acc	augmented

1(c) batch/layer normalization

model	layers	filters	learning rate	best test acc	augmented
	2	8*2,16*2	0.001	0.9265	
	3	8*2,16*2,32*2	0.001	0.9580	
	5	8*2,16*2,32*2,64*2,128*2	0.001	0.9895	
	2	8*2,16*2	0.01	0.9528	
	3	8*2,16*2,32*2	0.01	0.9869	
	5	8*2,16*2,32*2,64*2,128*2	0.01	0.9843	
	2	8,16	0.01	0.9659	
	3	8,16,32	0.01	0.9869	
	5	8,16,32,6,128	0.01	0.9764	

When batch/layer normalization was applied, a noticeable improvement in performance was observed, particularly when the learning rate was set to 0.01. This effect was especially pronounced for lower filter numbers.

1(c) separable convolutions

model	layers	filters	learning rate	best test acc	augmented
	2	8,16	0.01	0.5774	
	3	8,16,32	0.01	0.5774	
	5	8,16,32,6,128	0.01	0.5774	
	2	8*2,16*2	0.01	0.9160	
	3	8*2,16*2,32*2	0.01	0.9633	
	5	8*2,16*2,32*2,64*2,128*2	0.01	0.5774	
	2	8*2,16*2	0.001	0.8294	
	3	8*2,16*2,32*2	0.001	0.9738	
	5	8*2,16*2,32*2,64*2,128*2	0.001	0.8294	
	2	8,16	0.001	0.5774	
	3	8,16,32	0.001	0.9528	
	5	8,16,32,64,128	0.001	0.9055	

When separable convolutions were employed, there was no discernible enhancement in performance, with the exception of the case where the number of layers was set to 3, where a slight improvement in performance was observed.

1.(d)
1.(e)

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 for the rotation parameter.

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

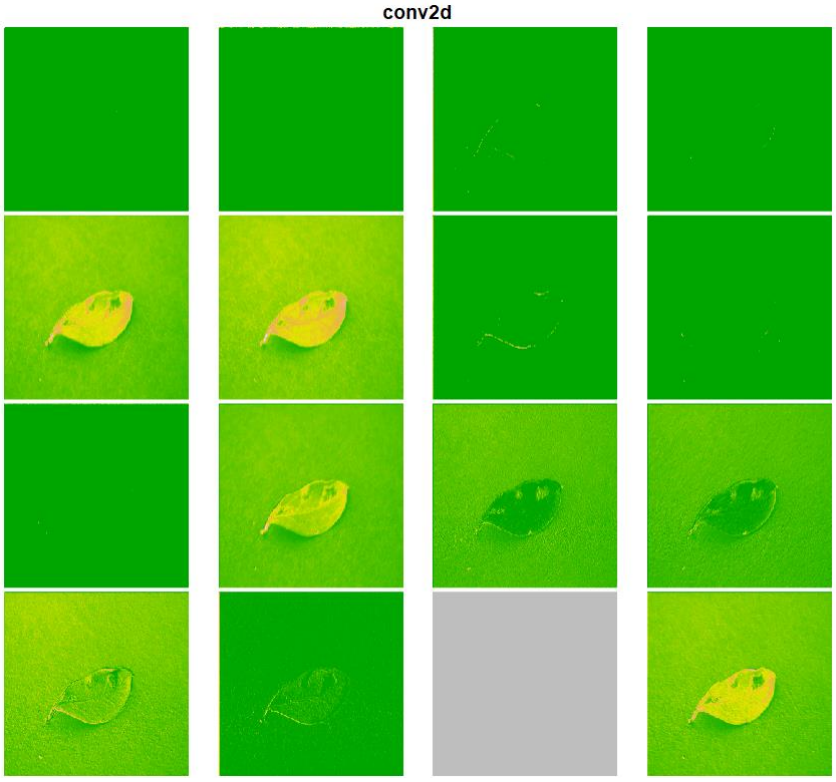
2(d)

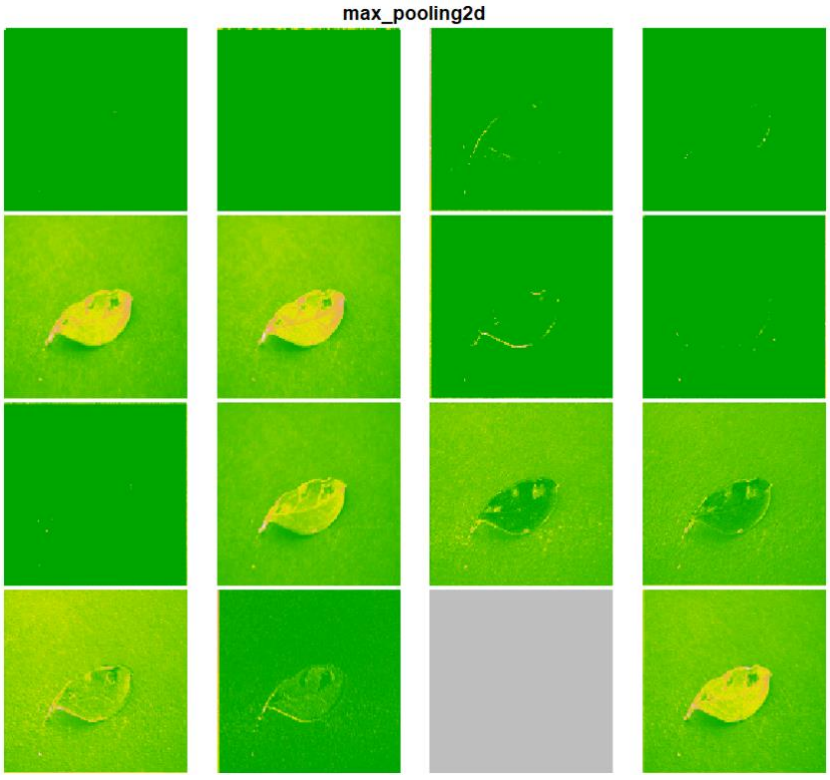
From the above result, choose a model with filter c(16,32), plus rotation to Provide a visualization of the interesting activation layers. The following figure is a summary of this model.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Trainable
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0         Y
 random_rotation (RandomRotation)  (None, 256, 256, 3)  0        Y
 conv2d (Conv2D)             (None, 256, 256, 16)      448       Y
 max_pooling2d (MaxPooling2D)  (None, 128, 128, 16)    0         Y
 conv2d_1 (Conv2D)           (None, 128, 128, 32)      4640      Y
 max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 32)    0         Y
 flatten (Flatten)           (None, 131072)            0         Y
 dense (Dense)               (None, 32)                4194336   Y
 dense_1 (Dense)             (None, 1)                 33        Y
=================================================================
Total params: 4199457 (16.02 MB)
Trainable params: 4199457 (16.02 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Displaying the test picture:

Visualizing every channel in every intermediate activation:





conv2d_1

max_pooling2d_1

2(e)

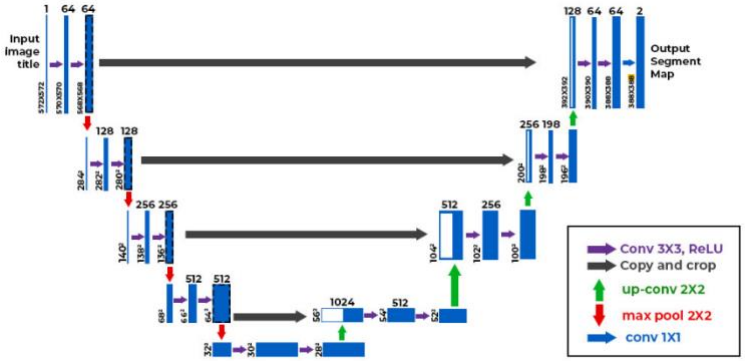
class_name	class_description	score
n01644373	tree_frog	0.30845919
n01644900	tailed_frog	0.03965421
n01739381	vine_snake	0.03001840

Q3

3(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

3(b)

Image 27 Loss/Accuracy

- Loss: 0.0495
- Accuracy: 0.9817

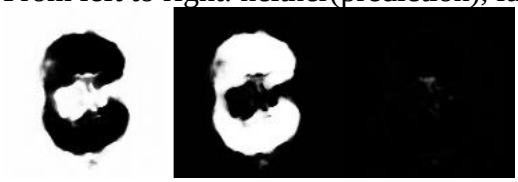
Confusion Matrix

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

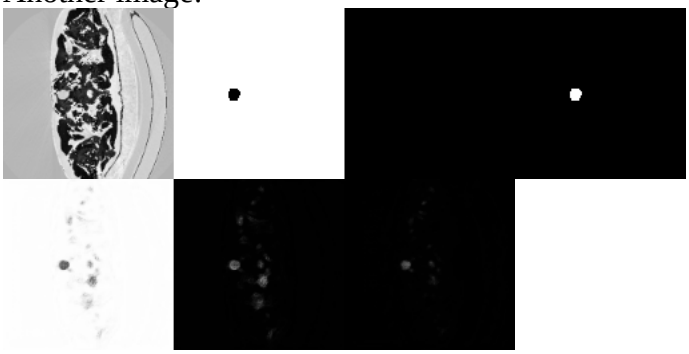
Slices of the images:

From left to right: x(data), neither(truth), lung(truth), airway(truth)

From left to right: neither(prediction), lung(prediction), airway(prediction)



Another image:



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.