What to submit:

- **Submit python code (jupyter notebook or .py file).**
- **Word document containing the accuracy results for different training mentioned in the Question below.**

In this assignment, you will use a technique called "data augmentation" which can be used in deep learning when the training data is small. "Data augmentation" simply means generation of artificial data to increase the training size so that deep learning model doesn't over fit and improves its prediction accuracy. **NOTE: data augmentation is only applied to the TRAINING SET and not to the testing set.**

You are given a folder "**cat_dog**" which contains only dog and cat images from cifar10 data set illustrated in class. The following are main subfolders contained in "**cat_dog**" folder.

- **train** subfolder: contains 10,000 32 x 32 x 3 dog and cat training images.
- **test** subfolder: contains 2,000 32 x 32 x 3 dog and cat testing images.
- **train_class_labels.txt** and **test_class_labels.txt** contain class labels of training and testing images, respectively. Cat is labeled '0' and dog as '1'.

There is also a python code called "**Augmentation.ipynb**", which will create **two additional images** (You can create more additional images by changing the variable '**NUM_ROTATE'** in **the third cell** of the notebook) for each original image by rotating the original image randomly with a degree between -45 degrees and 45 degrees. This script will save original images and the additional images in the folder **augmented/train/**. The rotated images' name with start with 'rotated'. This script will also create a file **augmented/train_class_labels.txt** that contain the class labels for images in **augmented/train/**. The script might take a few minutes to complete.

Your goal is to train and test the 2D **CNN model** with the following layers below.

Layer	**Notes**
2-dimensional CNN layer	32 (3,3) kernels, ReLU activation
2-dimensional max pooling	Pooling size of (2, 2). Add batch normalization and dropout rate of 0.2.
2-dimensional CNN layer	64 (3,3) kernels, ReLU activation
2-dimensional max pooling	Pooling size of (2, 2). Add batch normalization and dropout rate of 0.3.
Dense layer	100 nodes, ReLU activation. Add batch normalization and dropout rate of 0.5.
Output layer	1 node, sigmoid activation

But the learning rate hyperparameter in the optimizer should be varied.

- Try three different learning rate (**lr**) of optimizer: 0.001, 0.005, 0.05. You can use optimizer 'Adam'.

```
from tensorflow.keras.optimizers import Adam
```

- For example, to set learning rate to 0.01, simply use **OPTIMIZER = Adam(lr = 0.01)** when compiling the model.

```
model.compile(…, optimizer = Adam(lr=0.01), …)
```

Report a table for the accuracies on the testing set (Note the testing set is same):

	lr =0.05	lr =0.001	lr =0.0005
Model trained on original training data set			
Model trained on augmented training data set			

Plot the optimization history for the best learning rate for

- the Model trained on **original training data**
- the Model trained on the **augmented training data**

NOTE:

Since this is binary classification:

- The loss function should be 'binary_crossentropy'
- The output layer should have one node with 'Sigmoid' activation.
- No need to change the y_train and y_test into categorical format.

Grading notes:

- This part2 of midterm1 is worth 65 points.
- Augmenting the images: 10 points
- Model architecture: 20 points
- Model training for different learning rates: 15 points
- Reporting accuracy results: 10 points
- Reporting plots: 10 points