1. Define a function to store the result of the training.

```
def store_result(cur_result_list,i_method,i_history,i_train_acc,i_train_loss,i_test_acc,i_test_loss):
    new_result = {
        'method':i_method,
        'history':i_history,
        'acc':i_train_acc,
        'loss':i_train_loss,
        'test_acc':i_test_acc,
        'test_loss':i_test_loss
    }
    cur_result_list.append(new_result)
    return cur_result_list
result_list = []
```

2. Import the following function, use pathlib to get the images.

```
import PIL
import pathlib
from PIL import Image
# import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
Init Plugin
Init Graph Optimizer
Init Kernel
```

```
import pathlib
dir_path = './train'
dir_path = pathlib.Path(dir_path)
```

```
In [4]: list(dir_path.glob('*.jpg'))
```

```
PosixPath('train/dog.8005.jpg'),
PosixPath('train/dog.7336.jpg'),
PosixPath('train/cat.5063.jpg'),
PosixPath('train/cat.10145.jpg'),
PosixPath('train/dog.1747.jpg'),
PosixPath('train/cat.3412.jpg'),
PosixPath('train/dog.6028.jpg'),
PosixPath('train/dog.991.jpg'),
PosixPath('train/dog.5509.jpg'),
PosixPath('train/dog.10220.jpg'),
PosixPath('train/dog.2266.jpg'),
PosixPath('train/dog.3178.jpg'),
PosixPath('train/dog.749.jpg'),
PosixPath('train/cat.7884.jpg'),
PosixPath('train/cat.9671.jpg'),
PosixPath('train/dog.4617.jpg'),
PosixPath('train/cat.6542.jpg'),
PosixPath('train/dog.12437.jpg'),
PosixPath('train/cat.2724.jpg'),
```

3. Use PIL function to preview the images and size

```
PIL.Image.open(str(cat[0]))
```

```
PIL.Image.open(str(cat[0])).size
```

```
(399, 300)
```

Use resize function to resize all images to all same size

```
PIL.Image.open(str(cat[0])).resize((300,300))
```

4. Use dictionary to give images labels

```
animal_images_dict = {
    'cat':cat,
    'dog':dog
}

animal_labels_dict = {
    'cat':0,
    'dog':1
}
```

```
X = []
y = []

for animal, images in animal_images_dict.items():
    for image in images:
        resized_img=PIL.Image.open(str(image)).resize((300,300))
        resized_img = np.asarray(resized_img)
        X.append(resized_img)
        y.append(animal_labels_dict[animal])
```

5. Due to the efficiency and saving time, I choose 4000 images for training and 1000 for testing. Normalized the number to [0,1] by divide 255.

```
X_n = np.array(X)
y_n = np.array(y)
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_n,y_n,shuffle=True)
```

```
X_train = X_train[:4000]
X_test = X_test[:1000]
y_train = y_train[:4000]
y_test = y_test[:1000]
```

```
X_train_normalized = X_train/255
X_test_normalized = X_test/255
y_train = y_train
y_test = y_test
```

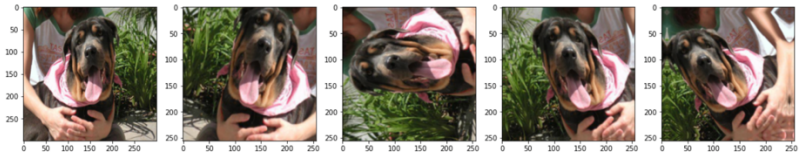
```
X_train_normalized.shape,X_test_normalized.shape
```

```
((4000, 300, 300, 3), (1000, 300, 300, 3))
```

```
X_train_tensor=tf.convert_to_tensor(X_train_normalized)
X_test_tensor=tf.convert_to_tensor(X_test_normalized)
y_train_tensor=tf.convert_to_tensor(y_train)
y_test_tensor=tf.convert_to_tensor(y_test)
```

6. Use tensorflow' s function keras.layers.experimental.preprocessing to add data augnmentation.

```
with tf.device("/cpu:0"):
    aug=keras.Sequential([
            layers.experimental.preprocessing.RandomRotation(0.3),
            layers.experimental.preprocessing.RandomCrop(256,256),
            layers.experimental.preprocessing.RandomZoom(0.1),
            layers.experimental.preprocessing.RandomFlip("horizontal",input_shape=(300,300,3))
            ])
fig,axs = plt.subplots(1,5,figsize=(20,20))
axs[0].imshow(X_train_tensor[0])
for i in range(4):
    aug_img = aug(X_train_tensor[0:1])
    axs[i+1].imshow(aug_img[0])
```



7. Define and train the model, save the model’ s result through step1’ s function.

```
model_method = 'CNN'
model = Sequential([
    layers.Conv2D(16,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(512,activation='relu'),
    layers.Dense(1,activation='sigmoid')
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),loss=tf.keras.losses.BinaryCrossentropy(),metr
model_history = model.fit(X_train_tensor,y_train_tensor,validation_data=(X_test_tensor,y_test_tensor),epochs=15)


train_loss = model_history.history['loss']
train_acc = model_history.history['accuracy']
test_loss = model_history.history['val_loss']
test_acc = model_history.history['val_accuracy']
store_result(result_list,model_method,model_history,train_acc,train_loss,test_acc,test_loss)
```

8. Add the augmentation

```
j=0
with tf.device("/cpu:0"):
    for i in [layers.experimental.preprocessing.RandomFlip("horizontal",input_shape=(300,300,3)),
              layers.experimental.preprocessing.RandomFlip("vertical",input_shape=(300,300,3)),
              layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical",input_shape=(300,300,3))]:

        method_name =['Flip_hor','Flip_ver','Flip_hor_n_ver']
        model_method = method_name[j]
        j+=1
        data_augmentation = keras.Sequential([
        i
        ])

        model = Sequential([
            data_augmentation,
            layers.Conv2D(16,3,padding='same',activation='relu'),
            layers.MaxPooling2D(),
            layers.Conv2D(32,3,padding='same',activation='relu'),
            layers.MaxPooling2D(),
            layers.Conv2D(64,3,padding='same',activation='relu'),
            layers.MaxPooling2D(),
            layers.Flatten(),
            layers.Dense(512,activation='relu'),
            layers.Dense(1,activation='sigmoid')
        ])
        with tf.device("/gpu:0"):
            model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),loss=tf.keras.losses.BinaryCrossent
            model_history = model.fit(X_train_tensor,y_train_tensor,validation_data=(X_test_tensor,y_test_tensor),ep

            train_loss = model_history.history['loss']
            train_acc = model_history.history['accuracy']
            test_loss = model_history.history['val_loss']
            test_acc = model_history.history['val_accuracy']

            store_result(result_list,model_method,model_history,train_acc,train_loss,test_acc,test_loss)
```

9. Draw the data from the list and check the result.

```
import pandas as pd
result_df = pd.DataFrame.from_dict(result_list)
```

```
result_df
```

	method	history	acc	loss	test_acc	test_loss
0	CNN	<tensorflow.python.keras.callbacks.History obj...	[0.54625004529953, 0.640250027179718, 0.724000...	[0.8300836682319641, 0.6385049819946289, 0.550...	[0.64000004529953, 0.6770000457763672, 0.69900...	[0.6526291370391846, 0.607878565788269, 0.5745...
1	Flip_hor	<tensorflow.python.keras.callbacks.History obj...	[0.5437500476837158, 0.6575000286102295, 0.729...	[0.8187201023101807, 0.6121012568473816, 0.535...	[0.5890000462532043, 0.7000000476837158, 0.666...	[0.6567996144294739, 0.582956075668335, 0.6133...
2	Flip_ver	<tensorflow.python.keras.callbacks.History obj...	[0.5275000333786011, 0.6057500243186951, 0.673...	[1.0572859048843384, 0.6474304795265198, 0.597...	[0.5879999995231628, 0.6600000262260437, 0.694...	[0.6677330732345581, 0.6398525834083557, 0.574...
3	Flip_hor_n_ver	<tensorflow.python.keras.callbacks.History obj...	[0.5490000247955322, 0.6187500357627869, 0.646...	[0.8725936412811279, 0.6415159106254578, 0.624...	[0.6080000400543213, 0.6550000309944153, 0.662...	[0.646767258644104, 0.6286848783493042, 0.6208...
4	Zoom_0.1	<tensorflow.python.keras.callbacks.History obj...	[0.5277500152587891, 0.5877500176429749, 0.636...	[0.8113829493522644, 0.6573504209518433, 0.625...	[0.5590000152587891, 0.5820000171661377, 0.645...	[0.6671802997589111, 0.6663954257965088, 0.640...
5	Zoom_0.2	<tensorflow.python.keras.callbacks.History obj...	[0.5557500123977661, 0.6175000071525574, 0.677...	[0.8266465663909912, 0.6380121111869812, 0.587...	[0.5580000281333923, 0.6160000562667847, 0.725...	[0.654741108417511, 0.6301575303077698, 0.5619...
6	Zoom_0.3	<tensorflow.python.keras.callbacks.History obj...	[0.530500054359436, 0.5587500333786011, 0.5980...	[0.8801295757293701, 0.6721411347389221, 0.654...	[0.5800000429153442, 0.6000000238418579, 0.625...	[0.672224760055542, 0.6557781100273132, 0.6362...

10. Plot the data through matplotlib' s function

```
fig,axs = plt.subplots(1,2,figsize=(15,5))
axs[0].plot(result_df['acc'][0],'orangered',label='train_acc')
axs[0].plot(result_df['test_acc'][0],'orangered',linestyle='--',label='test_acc')
axs[0].set_title('CNN')
axs[0].legend(loc='upper left')

axs[1].plot(result_df['loss'][0],'royalblue',label='train_loss')
axs[1].plot(result_df['test_loss'][0],'royalblue',linestyle='--',label='test_loss')
axs[1].set_title('CNN')
axs[1].legend(loc='lower left')
axs[0].set_xlabel('epoch')
axs[1].set_xlabel('epoch')
plt.show()
```