In [1]:

```
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import glob
```

In [2]:

```
img_path=glob.glob("d:/datasets/2_class/*/*.jpg")
```

In [3]:

```
img_path[:5],img_path[-5:]
```

Out[3]:

```
(['d:/datasets/2_class\\airplane\\airplane_001.jpg',
  'd:/datasets/2_class\\airplane\\airplane_002.jpg',
  'd:/datasets/2_class\\airplane\\airplane_003.jpg',
  'd:/datasets/2_class\\airplane\\airplane_004.jpg',
  'd:/datasets/2_class\\airplane\\airplane_005.jpg'],
 ['d:/datasets/2_class\\lake\\lake_696.jpg',
  'd:/datasets/2_class\\lake\\lake_697.jpg',
  'd:/datasets/2_class\\lake\\lake_698.jpg',
  'd:/datasets/2_class\\lake\\lake_699.jpg',
  'd:/datasets/2_class\\lake\\lake_700.jpg'])
```

In [4]:

```
import random
random.shuffle (img_path )
```

In [5]:

```
img_path[:5],img_path[-5:]
```

Out[5]:

```
(['d:/datasets/2_class\\lake\\lake_357.jpg',
  'd:/datasets/2_class\\lake\\lake_674.jpg',
  'd:/datasets/2_class\\airplane\\airplane_212.jpg',
  'd:/datasets/2_class\\airplane\\airplane_099.jpg',
  'd:/datasets/2_class\\lake\\lake_558.jpg'],
 ['d:/datasets/2_class\\airplane\\airplane_641.jpg',
  'd:/datasets/2_class\\lake\\lake_517.jpg',
  'd:/datasets/2_class\\airplane\\airplane_296.jpg',
  'd:/datasets/2_class\\airplane\\airplane_654.jpg',
  'd:/datasets/2_class\\lake\\lake_180.jpg'])
```

In [6]:

```
all_image_labels = [ int(p.split("\\")[1]=="lake") for p in img_path]
```

In [7]:

```
all_image_labels[-5:]
```

Out[7]:

```
[0, 1, 0, 0, 1]
```

In [8]:

```
def load_and_preprocess_image(path):
    image = tf.io.read_file(path)
    image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize(image, [256, 256])
    image = tf.cast(image, tf.float32)
    image = image/255.0  # normalize to [0,1] range
    return image
```

In [9]:

```
import matplotlib.pyplot as plt
rr=random.randint(0,len(img_path))
img_path_ = img_path[rr]
plt.imshow(load_and_preprocess_image(img_path_))
plt.axis("off")
print(rr,np.where(all_image_labels[rr]==1 ,"lake","airplane") )
```

```
223 airplane
```

In [10]:

```
path_ds = tf.data.Dataset.from_tensor_slices(img_path[:int(len(img_path)*.85)])
test_ds=tf.data.Dataset.from_tensor_slices(img_path[int(len(img_path)*.85):])
```

In [11]:

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
image_ds = path_ds.map(load_and_preprocess_image, num_parallel_calls=AUTOTUNE)
```

In [12]:

```
test_image_ds = test_ds.map(load_and_preprocess_image, num_parallel_calls=AUTOTUNE)
```

In [13]:

```
label_ds = tf.data.Dataset.from_tensor_slices(tf.cast(all_image_labels[:int(len(img_path)*.85)], tf.
test_label_ds=all_image_labels[int(len(img_path)*.85):]
```

In [14]:

```
image_label_ds = tf.data.Dataset.zip((image_ds, label_ds))
```

In [15]:

```
#image_label_ds.shuffle(len(img_path))
```

In [16]:

```
val_count = int(len(img_path)*0.85*0.2)
train_count = (int(len(img_path)*0.85)) - val_count
```

In [17]:

```
val_count,train_count
```

Out[17]:

```
(238, 952)
```

In [18]:

```
val_data = image_label_ds.take(val_count)
train_data = image_label_ds.skip(val_count)
```

In [19]:

```
train_data.take(5),test_ds,test_image_ds
```

Out[19]:

```
(<TakeDataset element_spec=(TensorSpec(shape=(256, 256, 3), dtype=tf.float32, name=N
one), TensorSpec(shape=(), dtype=tf.int64, name=None))>,
 <TensorSliceDataset element_spec=TensorSpec(shape=(), dtype=tf.string, name=None)>,
 <ParallelMapDataset element_spec=TensorSpec(shape=(256, 256, 3), dtype=tf.float32,
name=None)>)
```

In [20]:

```
BATCH_SIZE = 32
```

In [21]:

```
train_data = train_data.repeat()
train_data = train_data.batch(BATCH_SIZE)
train_data = train_data.prefetch(AUTOTUNE)
```

In [22]:

```
val_data = val_data.batch(BATCH_SIZE)
test_image_ds = test_image_ds.batch(BATCH_SIZE)
```

In [23]:

```
train_data,val_data,test_image_ds
```

Out[23]:

```
(<PrefetchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float
32, name=None), TensorSpec(shape=(None,), dtype=tf.int64, name=None))>,
 <BatchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32,
name=None), TensorSpec(shape=(None,), dtype=tf.int64, name=None))>,
 <BatchDataset element_spec=TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32,
name=None)>)
```

建立模型

In [36]:

```
model = tf.keras.Sequential()   #顺序模型
model.add(tf.keras.layers.Conv2D(128, (3, 3), input_shape=(256, 256, 3), activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D())
model.add(tf.keras.layers.Conv2D(128, (3, 3), input_shape=(256, 256, 3), activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

In [37]:

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 254, 254, 128)     3584

 batch_normalization_4 (Batc  (None, 254, 254, 128)    512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 127, 127, 128)    0
 2D)

 conv2d_5 (Conv2D)           (None, 125, 125, 128)     147584

 batch_normalization_5 (Batc  (None, 125, 125, 128)    512
 hNormalization)

 global_average_pooling2d_2   (None, 128)              0
 (GlobalAveragePooling2D)

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 168,833
Trainable params: 168,321
Non-trainable params: 512
_________________________________________________________________
```

In [38]:

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc']
)
```

In [39]:

```
steps_per_epoch = train_count//BATCH_SIZE
validation_steps = val_count//BATCH_SIZE
```

In [40]:

```
steps_per_epoch,validation_steps
```

Out[40]:

```
(29, 7)
```

In [*]:

```
history = model.fit(train_data, epochs=70, steps_per_epoch=steps_per_epoch, validation_data=val_data
```

```
Epoch 1/70
29/29 [==============================] - 9s 231ms/step - loss: 0.2472 - acc: 0.9084
- val_loss: 0.7798 - val_acc: 0.4496
Epoch 2/70
29/29 [==============================] - 5s 177ms/step - loss: 0.1230 - acc: 0.9537
- val_loss: 0.9876 - val_acc: 0.4496
Epoch 3/70
29/29 [==============================] - 5s 175ms/step - loss: 0.0938 - acc: 0.9698
- val_loss: 1.0417 - val_acc: 0.4496
Epoch 4/70
29/29 [==============================] - 5s 177ms/step - loss: 0.0869 - acc: 0.9698
- val_loss: 0.9223 - val_acc: 0.4496
Epoch 5/70
29/29 [==============================] - 5s 178ms/step - loss: 0.0733 - acc: 0.9784
- val_loss: 0.9540 - val_acc: 0.4496
Epoch 6/70
20/29 [===================>.....] - ETA: 1s - loss: 0.0707 - acc: 0.9734
```

In [60]:

```
history.history.keys()
```

Out[60]:

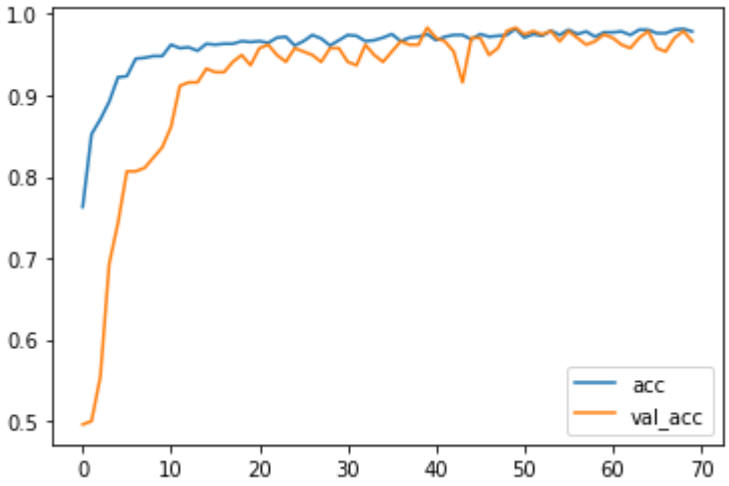
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

In [62]:

```
plt.plot(history.epoch, history.history.get('acc'), label='acc')
plt.plot(history.epoch, history.history.get('val_acc'), label='val_acc')
plt.legend()
```

Out[62]:

<matplotlib.legend.Legend at 0x236701ab430>

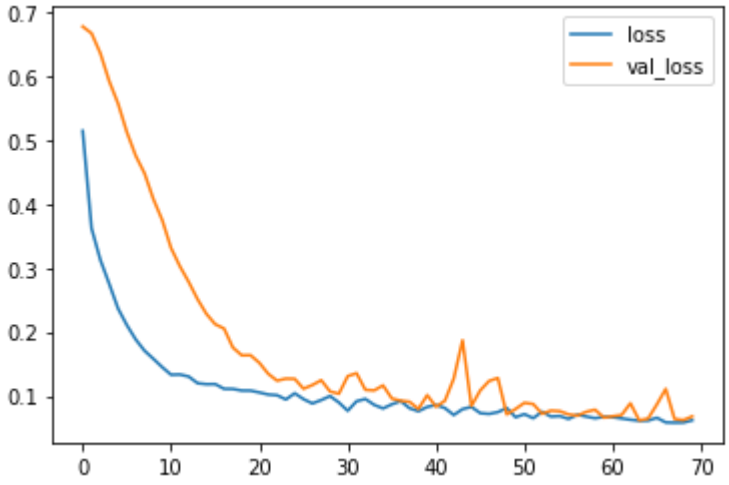


In [61]:

```
plt.plot(history.epoch, history.history.get('loss'), label='loss')
plt.plot(history.epoch, history.history.get('val_loss'), label='val_loss')
plt.legend()
```

Out[61]:

<matplotlib.legend.Legend at 0x2366f459340>



In [63]:

```
pred=model.predict(test_image_ds)
```

In [64]:

```
pred_=[int(i>0.5) for i in pred]
```

In [65]:

```
from  sklearn.metrics import classification_report
```

```
tt=test_data.unbatch()
k=[int(l) for _,l in tt]
```

In [66]:

```
print(classification_report(test_label_ds,pred_))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       103
           1       1.00      0.96      0.98       107

    accuracy                           0.98       210
   macro avg       0.98      0.98      0.98       210
weighted avg       0.98      0.98      0.98       210
```

In [ ]:

```

```