Image Classifier

目标

利用合适的网络对给定的图片数据集进行分类，找出训练网络的弱点。

分析

根据作业提供的下载地址，我们下载下来测试集合和训练集合。训练集合和测试集合中分别包括卡通和其他两个文件夹。训练集卡通文件夹包含 3000张卡通动漫图，其他文件夹包含3000比较杂的图片合集。测试集合分别包含400张验证的数据集。

首先这次作业和以往不一样的是，这次分类是对图片的分类而非文本，所以怎么理解处理图片选择合适的框架显得十分重要。

查阅了一下资料，一般图片分类分三种：

1. 使用KNN、SVM和BP神经网络进行图片分类，这个使用之前的SKLearn 就行
2. 使用CNN，深度神经网络，一般用Keras + Tensorflow
3. 再次训练深层神经网络的最后一层（称为Inception V3），同样也是由TensorFlow来实现。不过这个是在别人训练好的基础上做修改，准确率应该最高。

SKLearn 前几节课我们都了解过了，这里主要介绍下 Keras
Keras 是一个高层神经网络 API，它由纯 Python 编写而成并基于Tensorflow，并使用到深度学习框架 TensorFlow 和 Keras。TensorFlow 是Google AI 部门开源的机器学习和深度学习框架。

接下来我们将使用1，2种方式进行代码实现和验证结果。

设计

SKLearn + SVM

首先使用我们学过的 SVM

导入我们需要的 sklearn 库

```
import skimage.color
import skimage.feature
import skimage.io
import skimage.transform
import sklearn.svm

import os
```

我们使用的 skimage是Scikit-Image，基于python脚本语言开发的数字图片处理包。其中用的主要子模块如下，主要是对图片进行特征提取和处理。

子模块名称	主要实现功能
io	读取、保存和显示图片或视频
color	颜色空间变换
transform	几何变换或其它变换，如旋转、拉伸和拉东变换等
feature	特征检测与提取等

接下来我们定义训练数据集和测试数据集地址

```
# Data

TRAIN_PATH = '/Users/jackrex/Desktop/AILesson/L8/训练集/'
TEST_PATH = '/Users/jackrex/Desktop/AILesson/L8/验证集/'
```

读取训练和测试集中的图片，统一转成 256 * 256 的用于减少内存。 使用color模块的rgb2gray()函数，将彩色三通道图片转换为灰度图片，转换结果为float64类型的数组，范围在[0,1]之间。

```
def read_and_preprocess(im_path):
    im = skimage.io.imread(im_path)
    im = skimage.color.rgb2gray(im)
    im = skimage.transform.resize(im, (256, 256))
    return im


def get_data_tr():
    X = []
    Y = []
    for entry in os.scandir(TRAIN_PATH + '其他'):
        im = read_and_preprocess(entry.path)
        hf = skimage.feature.hog(im, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1))
        X.append(hf)
        Y.append(0)
    for entry in os.scandir(TRAIN_PATH + '卡通'):
        im = read_and_preprocess(entry.path)
        hf = skimage.feature.hog(im, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1))
        X.append(hf)
        Y.append(1)
    return X, Y


def get_data_te():
    X = []
    Y = []
    for entry in os.scandir(TEST_PATH + '其他'):
        im = read_and_preprocess(entry.path)
        hf = skimage.feature.hog(im, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1))
        X.append(hf)
        Y.append(0)
    for entry in os.scandir(TEST_PATH + '卡通'):
        im = read_and_preprocess(entry.path)
        hf = skimage.feature.hog(im, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1))
        X.append(hf)
        Y.append(1)
    return X, Y
```

接下来我们使用 SVM 进行训练，并且拿测试集合进行验证

```python
# Train
Xtr, Ytr = get_data_tr()
clf = sklearn.svm.SVC(probability=True)
clf.fit(Xtr, Ytr)

# Test
Xte, Yte = get_data_te()
r = clf.predict(Xte)
s = 0
for i in range(len(r)):
    if r[i] == Yte[i]:
        s += 1
print('acc:', s / len(r))
```

最终结果如下：

```
→  L8 git:(master) ✗ python3 ImageClassifier-SVM.py
/usr/local/lib/python3.6/site-packages/skimage/transform/_warps.py:105: UserWarning: The default mode, 'constant',
  warn("The default mode, 'constant', will be changed to 'reflect' in "
/usr/local/lib/python3.6/site-packages/skimage/transform/_warps.py:110: UserWarning: Anti-aliasing will be enabled
g images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
/usr/local/lib/python3.6/site-packages/skimage/feature/_hog.py:150: skimage_deprecation: Default value of `block_no
ress this message specify explicitly the normalization method.
  skimage_deprecation)
/usr/local/lib/python3.6/site-packages/sklearn/svm/base.py:196: FutureWarning: The default value of gamma will cha
 features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/usr/local/lib/python3.6/site-packages/skimage/transform/_warps.py:105: UserWarning: The default mode, 'constant',
  warn("The default mode, 'constant', will be changed to 'reflect' in "
/usr/local/lib/python3.6/site-packages/skimage/transform/_warps.py:110: UserWarning: Anti-aliasing will be enabled
g images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
/usr/local/lib/python3.6/site-packages/skimage/feature/_hog.py:150: skimage_deprecation: Default value of `block_no
ress this message specify explicitly the normalization method.
  skimage_deprecation)
acc: 0.56875
```

准确率大约 56% 左右。

Keras + Tensorflow

编写代码：
导入我们需要的库

```python
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import SGD
from keras_tqdm import TQDMCallback  # add progress
from keras.callbacks import Callback
import matplotlib.pyplot as plt
```

接着我们使用 ImageDataGenerator 类进行图像增强和加载本地数据。

```
# Data

TRAIN_PATH = '/Users/jackrex/Desktop/AILesson/L8/训练集/'
TEST_PATH = '/Users/jackrex/Desktop/AILesson/L8/验证集/'

# Build Model

batch_size = 32
train_data_gen = ImageDataGenerator(rescale=1 / 255., shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_data_gen = ImageDataGenerator(rescale=1 / 255.)

train_generator = train_data_gen.flow_from_directory(TRAIN_PATH, target_size=(150, 150), batch_size=batch_size,
                                                     class_mode="categorical")
test_generator = test_data_gen.flow_from_directory(TEST_PATH, target_size=(150, 150), batch_size=batch_size,
                                                   class_mode="categorical")
```

接下来进行模型构建

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation="relu", padding="same"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

epochs = 20
l_rate = 0.01
decay = l_rate / epochs
sgd = SGD(lr=l_rate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.summary()
```

我们使用了拥有 3 个卷积/池化层和 2 个全连接层的 CNN。3 个卷积层将分别使用 32，32，64 的 3 * 3的滤波器（fiter）。在两个全连接层，使用了 dropout 来避免过拟合。

最后打出 model 模型 其实可以看到整个神经网络模型如下：
可以清楚的看到整个网络结构和每一层。

```
→  L8 git:(master) python3 ImageClassifier-Keras.py
Using TensorFlow backend.
Found 6000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 150, 150, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 18, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                1327168
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 1,355,938
Trainable params: 1,355,938
Non-trainable params: 0
```

最后开始训练，并且把 training loss 和 validation loss 用图表展示出来。

```
n = 30
ratio = 0.1
fit_model = model.fit_generator(train_generator, steps_per_epoch=int(n * (1 - ratio)), epochs=epochs,
                                validation_data=test_generator, validation_steps=int(n * ratio),
                                callbacks=[TQDMCallback(), early_stopping, history])


losses, val_losses = history.losses, history.val_losses
fig = plt.figure(figsize=(15, 5))
plt.plot(fit_model.history['loss'], 'g', label='train losses')
plt.plot(fit_model.history['val_loss'], 'r', label='val losses')
plt.grid(True)
plt.title('Training Loss vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

开始运行，装好 Keras 之后，直接运行报错。

```
/usr/local/Cellar/python3/3.7.1/bin/python3.7 /Users/jackrex/Desktop/AILesson/L8/ImageClassifier.py
Using TensorFlow backend.
Traceback (most recent call last):
  File "/Users/jackrex/Desktop/AILesson/L8/ImageClassifier.py", line 4, in <module>
    import keras
  File "/usr/local/lib/python3.7/site-packages/keras/__init__.py", line 3, in <module>
    from . import utils
  File "/usr/local/lib/python3.7/site-packages/keras/utils/__init__.py", line 6, in <module>
    from . import conv_utils
  File "/usr/local/lib/python3.7/site-packages/keras/utils/conv_utils.py", line 9, in <module>
    from .. import backend as K
  File "/usr/local/lib/python3.7/site-packages/keras/backend/__init__.py", line 89, in <module>
    from .tensorflow_backend import *
  File "/usr/local/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py", line 5, in <module>
    import tensorflow as tf
ModuleNotFoundError: No module named 'tensorflow'

Process finished with exit code 1
```

因为缺少了tensorflow 作为后端支持，我们安装 tensorflow

```
➜  AILesson git:(master) pip3 install tensorflow
Collecting tensorflow
  Could not find a version that satisfies the requirement tensorflow (from versions: )
No matching distribution found for tensorflow
➜  AILesson git:(master) pip3 install tensorflow
```

安装失败，原因是因为我们的Python 版本是 3.7.1 tensorflow 还有支持该版本的包，所以我们降级Python 版本。

降级其实并不复杂
先 brew unlink python

其次执行一个别人写好的脚本，省事多了
brew install
https://raw.githubusercontent.com/Homebrew/homebrew-core/f2a764ef944b1080be64bd88dca9a1d80130c558/Formula/python.rb

接着继续安装，安装成功

```
➜  AILesson git:(master) pip3 install tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/81/07/be624c7e0a63b080a76b7f6faf417eecdc5f6480f6a740a8bcf8991bce0b/tensorflow-1.12.0-cp36-cp36m-m
    100% |████████████████████████████████| 62.0MB 471kB/s
Collecting numpy>=1.13.3 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/74/68/2b00ba3c7390354db2a1706291750b6b7e911f6f79c0bd2184ae04f3c6fd/numpy-1.15.4-cp36-cp36m-macosx
x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (24.5MB)
    100% |████████████████████████████████| 24.5MB 917kB/s
Collecting gast>=0.2.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/33/35/f7d7f3d1a63a2b32f1f54bf342ec9af2bb411f2a82d94cf5b88381d9dbd5/gast-0.2.1.post0.tar.gz
Collecting keras-preprocessing>=1.0.5 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/fc/94/74e0fa783d3fc07e41715973435dd051ca89c550881b3454233c39c73e69/Keras_Preprocessing-1.0.5-py2
Collecting six>=1.10.0 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.w
Collecting absl-py>=0.1.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/0c/63/f505d2d4c21db849cf80bad517f0065a30be6b006b0a5637f1b95584a305/absl-py-0.6.1.tar.gz (94kB)
    100% |████████████████████████████████| 102kB 10.8MB/s
Collecting tensorboard<1.13.0,>=1.12.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/07/53/8d32ce9471c18f8d99028b7cef2e5b39ea8765bd7ef250ca05b490880971/tensorboard-1.12.2-py3-none-an
    100% |████████████████████████████████| 3.1MB 4.6MB/s
Collecting grpcio>=1.8.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/14/26/e26ed357d2b7ddb30043b6f17458c1fb8d3a2e7a243fb03a022beef9084a/grpcio-1.17.1-cp36-cp36m-macos
    100% |████████████████████████████████| 2.1MB 5.5MB/s
```

执行脚本，报错

```
  File "/usr/local/lib/python3.6/site-packages/six.py", line 693, in reraise
    raise value
  File "/usr/local/lib/python3.6/site-packages/keras/utils/data_utils.py", line 685, in get
    inputs = self.queue.get(block=True).get()
  File "/usr/local/Cellar/python/3.6.5_1/Frameworks/Python.framework/Versions/3.6/lib/python3.6/multiprocessing/pool.py", line 644, in get
    raise self._value
  File "/usr/local/Cellar/python/3.6.5_1/Frameworks/Python.framework/Versions/3.6/lib/python3.6/multiprocessing/pool.py", line 119, in worker
    result = (True, func(*args, **kwds))
  File "/usr/local/lib/python3.6/site-packages/keras/utils/data_utils.py", line 626, in next_sample
    return six.next(_SHARED_SEQUENCES[uid])
  File "/usr/local/lib/python3.6/site-packages/keras_preprocessing/image.py", line 1526, in __next__
    return self.next(*args, **kwargs)
  File "/usr/local/lib/python3.6/site-packages/keras_preprocessing/image.py", line 1970, in next
    return self._get_batches_of_transformed_samples(index_array)
  File "/usr/local/lib/python3.6/site-packages/keras_preprocessing/image.py", line 1923, in _get_batches_of_transformed_samples
    interpolation=self.interpolation)
  File "/usr/local/lib/python3.6/site-packages/keras_preprocessing/image.py", line 496, in load_img
    raise ImportError('Could not import PIL.Image. '
ImportError: Could not import PIL.Image. The use of `array_to_img` requires PIL.
→  AILesson git:(master) ✗ pip3 install pillow
```

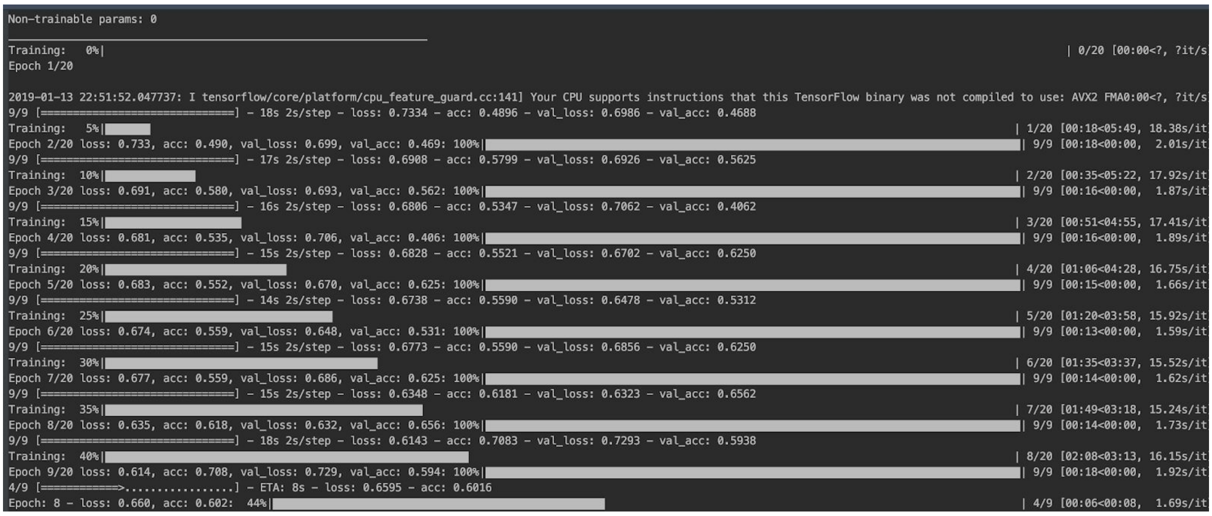
使用 Keras + Tensorflow 需要安装 PIL pillow 处理图片相关，先安装Pillow，装好之后再次运行，继续报错。

```
AILesson
    File "/Users/jackrex/Desktop/AILesson/L8/ImageClassifier.py", line 61, in <module>
      fit_model = model.fit_generator(train_generator, steps_per_epoch=int(n * (1 - ratio)), epochs=50, validation_data=test_generator, validation_steps=int(n*ratio))
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/legacy/interfaces.py", line 91, in wrapper
      return func(*args, **kwargs)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/engine/training.py", line 1418, in fit_generator
      initial_epoch=initial_epoch)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/engine/training_generator.py", line 234, in fit_generator
      workers=0)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/legacy/interfaces.py", line 91, in wrapper
      return func(*args, **kwargs)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/engine/training.py", line 1472, in evaluate_generator
      verbose=verbose)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/engine/training_generator.py", line 330, in evaluate_generator
      generator_output = next(output_generator)
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/utils/data_utils.py", line 709, in get
      six.reraise(*sys.exc_info())
    File "/Users/jackrex/Desktop/AILesson/venv/lib/python2.7/site-packages/keras/utils/data_utils.py", line 685, in get
      inputs = self.queue.get(block=True).get()
    File "/usr/local/Cellar/python@2/2.7.15_1/Frameworks/Python.framework/Versions/2.7/lib/python2.7/multiprocessing/pool.py", line 572, in get
      raise self._value
  IOError: image file is truncated (4 bytes not processed)
```

解决方案
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

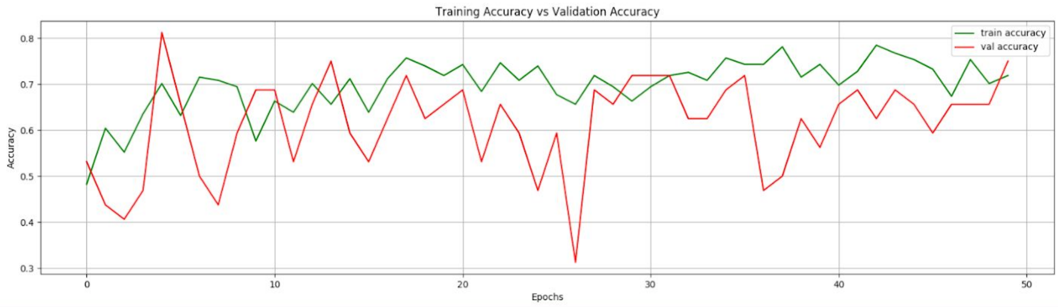
可能因为部分图片比较大，程序处理的时候做了截断处理。但是后面验证的时候抛出截断异常，PIL 默认的 LOAD_TRUNCATED_IMAGES 是 false

训练中



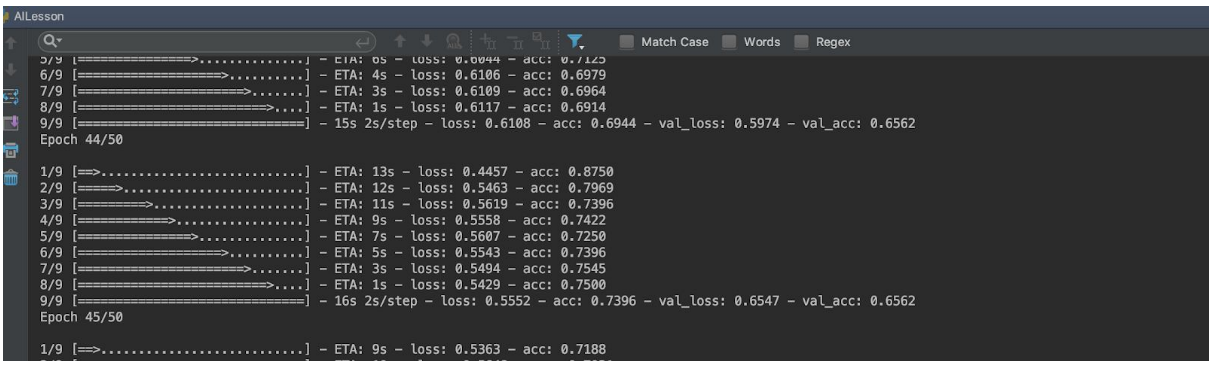
```
Non-trainable params: 0
________________________________________________________________
Training:   0%|                                                  | 0/20 [00:00<?, ?it/s
Epoch 1/20

2019-01-13 22:51:52.047737: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA0:00<?, ?it/s
9/9 [==============================] - 18s 2s/step - loss: 0.7334 - acc: 0.4896 - val_loss: 0.6986 - val_acc: 0.4688
Training:   5%|                                                  | 1/20 [00:18<05:49, 18.38s/it
Epoch 2/20 loss: 0.733, acc: 0.490, val_loss: 0.699, val_acc: 0.469: 100%|| 9/9 [00:18<00:00,  2.01s/it
9/9 [==============================] - 17s 2s/step - loss: 0.6908 - acc: 0.5799 - val_loss: 0.6926 - val_acc: 0.5625
Training:  10%|                                                  | 2/20 [00:35<05:22, 17.92s/it
Epoch 3/20 loss: 0.691, acc: 0.580, val_loss: 0.693, val_acc: 0.562: 100%|| 9/9 [00:16<00:00,  1.87s/it
9/9 [==============================] - 16s 2s/step - loss: 0.6806 - acc: 0.5347 - val_loss: 0.7062 - val_acc: 0.4062
Training:  15%|                                                  | 3/20 [00:51<04:55, 17.41s/it
Epoch 4/20 loss: 0.681, acc: 0.535, val_loss: 0.706, val_acc: 0.406: 100%|| 9/9 [00:16<00:00,  1.89s/it
9/9 [==============================] - 15s 2s/step - loss: 0.6828 - acc: 0.5521 - val_loss: 0.6702 - val_acc: 0.6250
Training:  20%|                                                  | 4/20 [01:06<04:28, 16.75s/it
Epoch 5/20 loss: 0.683, acc: 0.552, val_loss: 0.670, val_acc: 0.625: 100%|| 9/9 [00:15<00:00,  1.66s/it
9/9 [==============================] - 14s 2s/step - loss: 0.6738 - acc: 0.5590 - val_loss: 0.6478 - val_acc: 0.5312
Training:  25%|                                                  | 5/20 [01:20<03:58, 15.92s/it
Epoch 6/20 loss: 0.674, acc: 0.559, val_loss: 0.648, val_acc: 0.531: 100%|| 9/9 [00:13<00:00,  1.59s/it
9/9 [==============================] - 15s 2s/step - loss: 0.6773 - acc: 0.5590 - val_loss: 0.6856 - val_acc: 0.6250
Training:  30%|                                                  | 6/20 [01:35<03:37, 15.52s/it
Epoch 7/20 loss: 0.677, acc: 0.559, val_loss: 0.686, val_acc: 0.625: 100%|| 9/9 [00:14<00:00,  1.62s/it
9/9 [==============================] - 15s 2s/step - loss: 0.6348 - acc: 0.6181 - val_loss: 0.6323 - val_acc: 0.6562
Training:  35%|                                                  | 7/20 [01:49<03:18, 15.24s/it
Epoch 8/20 loss: 0.635, acc: 0.618, val_loss: 0.632, val_acc: 0.656: 100%|| 9/9 [00:14<00:00,  1.73s/it
9/9 [==============================] - 18s 2s/step - loss: 0.6143 - acc: 0.7083 - val_loss: 0.7293 - val_acc: 0.5938
Training:  40%|                                                  | 8/20 [02:08<03:13, 16.15s/it
Epoch 9/20 loss: 0.614, acc: 0.708, val_loss: 0.729, val_acc: 0.594: 100%|| 9/9 [00:18<00:00,  1.92s/it
4/9 [============>.................] - ETA: 8s - loss: 0.6595 - acc: 0.6016
Epoch: 8 - loss: 0.660, acc: 0.602:  44%|                            | 4/9 [00:06<00:08,  1.69s/it
```

图表生成结果



可以看到随着 epochs 的增加绿色代表的 train accuracy 准确率逐步上升。

最终结果



```
AILesson
Match Case  Words  Regex
5/9 [===============>..............] - ETA: 6s - loss: 0.6044 - acc: 0.7125
6/9 [===================>..........] - ETA: 4s - loss: 0.6106 - acc: 0.6979
7/9 [======================>.......] - ETA: 3s - loss: 0.6109 - acc: 0.6964
8/9 [==========================>...] - ETA: 1s - loss: 0.6117 - acc: 0.6914
9/9 [==============================] - 15s 2s/step - loss: 0.6108 - acc: 0.6944 - val_loss: 0.5974 - val_acc: 0.6562
Epoch 44/50

1/9 [==>...........................] - ETA: 13s - loss: 0.4457 - acc: 0.8750
2/9 [=====>........................] - ETA: 12s - loss: 0.5463 - acc: 0.7969
3/9 [=========>....................] - ETA: 11s - loss: 0.5619 - acc: 0.7396
4/9 [============>.................] - ETA: 9s - loss: 0.5558 - acc: 0.7422
5/9 [===============>..............] - ETA: 7s - loss: 0.5607 - acc: 0.7250
6/9 [===================>..........] - ETA: 5s - loss: 0.5543 - acc: 0.7396
7/9 [======================>.......] - ETA: 3s - loss: 0.5494 - acc: 0.7545
8/9 [==========================>...] - ETA: 1s - loss: 0.5429 - acc: 0.7500
9/9 [==============================] - 16s 2s/step - loss: 0.5552 - acc: 0.7396 - val_loss: 0.6547 - val_acc: 0.6562
Epoch 45/50

1/9 [==>...........................] - ETA: 9s - loss: 0.5363 - acc: 0.7188
```

可以看到随着 epoch 的增加，loss 逐渐减少，Accuracy 逐步增加，经过个第45个 epoch 最终准确率结果达到了87.5% 。

攻击

那么我们要找出训练网络的弱点，该怎么去着手呢，比较直接的做法就是我们在测试集合中加入混淆的图片来去判断。

除了上述所说我们可以拿训练集合去训练，测试集合去验证，最后得到最终的准确率之外，我们也可以使用单个图片去判断。代码入下：

```
file_path = TEST_PATH + '其他/' + "006510_0.jpg"
img = image.load_img(file_path, target_size=(150, 150))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)

y = model.predict(x)
print(y)
```

我们使用训练完的 fit_model， 把任意一张测试数据加载出来，数据化之后作为 X 输入，使用 predict 进行预测，得到预测结果。

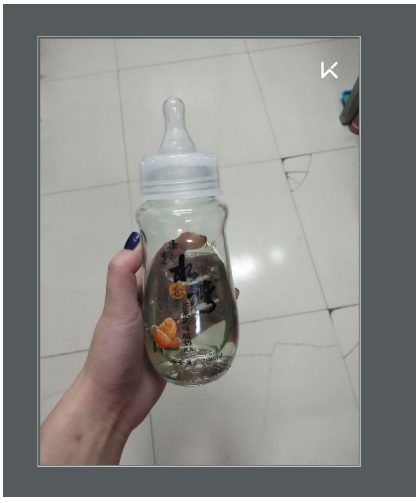
```
→  AILesson git:(master) ✗ python3 L8/ImageClassifier-Keras.py
Using TensorFlow backend.
Found 6000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
{'其他': 0, '卡通': 1}
```

```
27/27 [==============================] - 1s 55ms/step - loss: 0.7002 - acc: 0.5185 - val_loss: 0.6938 - val_acc: 0.0000e+
Epoch 30/30
27/27 [==============================] - 1s 39ms/step - loss: 0.6900 - acc: 0.5926 - val_loss: 0.7238 - val_acc: 0.3333
[[0.88974017 0.11025986]]
→  AILesson git:(master) ✗
```

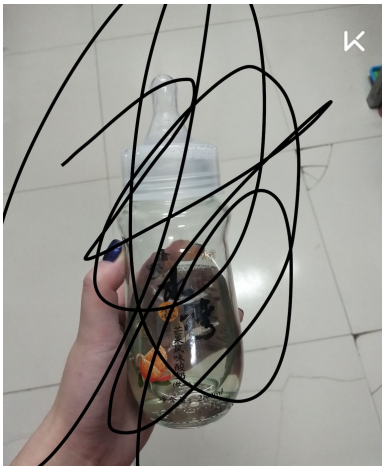
可以看到这个预测结果 88% 是其他，11% 是卡通，应为我们预测的是其他里面的 006510_0.jpg 所以，这个结果是对的。

现在我们对这个图片进行修改

这是原来的图：



修改后：



进行测试，结果如下

```
27/27 [==============================] - 1s 55ms/step - loss: 0.7005 - acc: 0.2963 - val_loss: 0.6952 - val_acc: 0.3333
Epoch 30/30
27/27 [==============================] - 1s 53ms/step - loss: 0.6830 - acc: 0.5185 - val_loss: 0.7493 - val_acc: 0.3333
[[0.1403865 0.8596135]]
➜  AILesson git:(master) ✗
```

可以看到已经预测错了。之前的预测结果 88% 是其他，11% 是卡通，现在变成了 14是其他，85%是卡通，其实我就是乱画了几下。

既然这么脆弱，我是不是可以轻微的修改下：
底下这张图，我在中间点击了一个红点。

最后的结果：

```
24/27 [=========================>....] - ETA: 0s - loss: 0.7070 - acc: 0.5417
25/27 [=========================>...] - ETA: 0s - loss: 0.7032 - acc: 0.5600
27/27 [==============================] - 1s 42ms/step - loss: 0.6956 - acc: 0.5926 - val_loss: 0.6624 - val_acc: 0.6667
[[0.70201975 0.29798025]]

Process finished with exit code 0
```

其实可以看到准确率和最初的 88% 相比，已经下降到了 70%，变化是非常大的，其实我只是做了一个很小的改动。

接下来对这个图片在做一个处理 对比度亮度调整了一下：

结果：

```
25/27 [============================>...] - E
26/27 [============================>..] - E
27/27 [==============================] - 1
[[6.283207e-04 9.993717e-01]]
```

可以看到概率极速下降，变成了一个非常小的数字。而且判断直接错误，因为概率对比卡通更大一些，但其实我们验证的是其他的图片。

同理我们对一组数据进行预测，然后可以进行攻击后看结构是否发生变化

```
27/27 [==============================] - 1s 54ms/step - loss: 0.7044 - acc: 0.4815 - val_loss: 0.8767 - val_acc: 0.0000e+00
20/20 [==============================] - 1s 29ms/step
[[0.5838573  0.4161427 ]
 [0.5838612  0.41613877]
 [0.5838573  0.4161427 ]
 [0.5838573  0.4161427 ]
 [0.5838573  0.4161427 ]
 [0.5838573  0.4161427 ]
 [0.58386457 0.41613543]
 [0.5838571  0.41614285]
 [0.5838572  0.41614282]
 [0.58385795 0.41614208]
 [0.5838517  0.4161484 ]
 [0.58385414 0.41614586]
 [0.583857   0.416143  ]
 [0.5838793  0.4161207 ]
 [0.5838573  0.4161427 ]
 [0.5838573  0.4161427 ]
 [0.5838572  0.4161428 ]
 [0.58385736 0.41614264]
 [0.5838573  0.4161427 ]
 [0.58385605 0.41614395]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
predict  %s 其他
title    其他/006510_0.jpg
```

这只是个示例，就不一个个改图片了，和单个图片攻击方法类似，然后看结果。

最终结论：

可以通过对图像添加小量的人类察觉不到的扰动误导神经网络做出误分类。这种攻击可以降低识别概率。还有另外的一种方式是调整色差，色值，或者使用滤镜。那么整个图片的 RGB 都发生了巨大改变导致判错概率极速上升。这也从另外一个维度反映了图片识别网络的脆弱性。

如果要输出策略的话，我们可以通过修改整个图片中找到权重比较大的几个像素分布点做修改，那么可以使得图片从人的角度看完全一样，但是对于机器来说就是天壤之别了。 整个攻击策略可以总结为求解让神经网络做出误分类的最大扰动的方程，然后判断在哪个位置、修改像素去做攻击。

攻击的方法很多，查了一下网上的资料，有一个总结了12种攻击方法和15种防御方法。

https://cloud.tencent.com/developer/news/130245

总结一下这个文章说描述的防御方法：
1）在学习过程中修改训练过程或者修改的输入样本。
2）增强网络，比如：添加更多层/子网络、改变损失/激活函数等。
3）当分类未见过的样本时，用外部模型作为附加网络

由于时间关系，暂时就没有去实验了，但是基本的策略和方向是这些应该没有问题。