Fruit Recognition Deep Learning Project

INFO 7390: Advances in Data Science & Architecture

Under, Prof. Srikanth Krishnamurthy, and TA Pramod Nagare

Team Members in Group 7:

Chiwen Shi

Shuhao Xia

Wensong Liu

Ying Lu

Contents

1. Overview

In most cases, when we try to log in or register a new account in a system, we have to enter verification code or select some specific kind of pictures to certify that we are not robots. Trying to train the model on how to distinguish a certain kind of pictures can help it verify whether the user's verification information is correct in a more intelligent and accurate way.

2. Objectives

To implement an application which can identify the certain kind of fruit with the image we input, there are mainly two parts of work we have to do:

- Build a Convolutional Neural Network (CNN) using Keras for classification and deploy the CNN model in Alibaba Cloud for the further use of the web application
- Build a web application using Flask through which users can enter a URL of the target image ang get the classification result from server

3. Data

3.1 Data Source and Dataset Description

The CNN model we built can identify seven kinds of fruits including apple, banana, kiwi, lemon, orange, pitaya and pineapple. The original data for training the CNN model comes from a Kaggle dataset: https://www.kaggle.com/moltean/fruits From this Kaggle dataset, we selected all the images of red apple, banana, kiwi, lemon, orange, pitaya and pineapple.

3.2 Data Augmentation

As is known to us, data is important in machine learning, as well as deep learning. Basically, the more images we have for the dataset, the more accurate our model will be. (There is a threshold, the accuracy of the classification model will not be improved when the threshold is achieved.) Therefore, in order to improve the performance of our CNN model, data augmentation is needed.

At first, we want to implement data augmentation on the original dataset. Data augmentation is used to create multiple alteration of each image, where the signal or the object in the picture is kept invariant, while the noise or the background is distorted. These distortions include cropping, scaling and rotating the image.

When we tried to test our model after building it, we found that it cannot identify pineapple correctly. We found that data augmentation is not the best way to expand the dataset. For example, all the pineapple images in our dataset do not have stems, so the CNN model trained by these images cannot identify pineapple with stem on it. What we need to do is to

increase the diversity of the dataset (for example, we need to have pineapple with stem in our dataset).

Solution: we downloaded images in bulk using Bing Image Search API v7.0 to expand our dataset.

4. Process Outline

4.1 Image Preprocessing

```
# loop over the input images
for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))
    image = img_to_array(image)
    data.append(image)
```

Read the image and resize it to specific width and height, then convert the image to array.

```
# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
print("[INFO] data matrix: {:.2f}MB".format(
    data.nbytes / (1024 * 1000.0)))
```

Normalize the array data so that they can be fit into the convolutional neural network.

4.2 Modeling

```
76  # partition the data into training and testing splits using 80% of
77  # the data for training and the remaining 20% for testing
78  (trainX, testX, trainY, testY) = train_test_split(data,
79      labels, test_size=0.2, random_state=42)
80
```

Split the whole dataset to train dataset (80% of the entire dataset) and test dataset (20% of the entire dataset).

```
 94  # train the network
 95  print("[INFO] training network...")
 96  H = model.fit_generator(
 97      aug.flow(trainX, trainY, batch_size=BS),
 98      validation_data=(testX, testY),
 99      steps_per_epoch=len(trainX) // BS,
100      epochs=EPOCHS, verbose=1)
```

Fit CNN model with our processed dataset.

4.3 Deployment Implementation

Below steps briefly introduces how we deploy our flask application on Alibaba ECS virtual machine:

1) **Start the ECS on Alibaba cloud, and login to the ECS virtual machine, which would be a ubuntu 14.04 system.** However, this does not meet our requirement, because Tensorflow does not support ubuntu 14.04. To do the upgrade, just simply do:
   - `sudo apt-get update`
   - `sudo apt-get upgrade`
   - `sudo apt-get install update-manager-core`
   - `do-release-upgrade`

   This might take 1 hour or more.

2) **Install Python v3.6.5** (because version higher than 3.6 is not supported by Tensorflow):
   - install all the dependencies: `sudo apt-get install openssl-devel bzip2-devel expat-devel gdbm-devel readline-devel sqlite-devel gcc gcc-c++ openssl-devel`
   - download python from official site: `wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz`
   - decompress the file: `tar -xf Python-3.6.5.tgz`
   - cd into folder and modify the setup file(to enable ssl): `vim ~/Python-3.6.5/Modules/Setup.dist`

   In this file, several lines about ssl and socketmodule are commented out, need to delete the comment, otherwise python would raise 'No Module named _ssl'

   - configure the installation:`./configure --prefix=/whatever u want to install python/`, `make`, `make install`

3) **Install all other packages using pip (may need to reset the environment PATH to use** pip3.6 rather than pip3.7): tensorflow, keras, sklearn, gunicorn, supervisor, and most importantly, virtualenv.

4) **Start a virtualenv using python3.6**
   - `virtualenv -p /path/to/python3.6 env`

5) **Upload Flask app using sftp**

6) **Configure gunicorn in Flask:**
   - `touch gunicorn.conf`
   - `vim gunicorn.conf`

only two lines in the configuration file: 1. workers = any number of workers you want, 2.bind='127.0.0.1:8888'

7) **Configure supervisor:**

   cd to the supervisor folder: `cd /etc/supervisor/conf.d`

   create configuration file: `touch info7390.conf`

   need to specify the web app folder path and also the command to start the server

   the reason we use supervisor is because it can help us supervise the server, autostart and autorestart when the server is down

   for more details about how to configure the supversor, please refer to `supervisor.org`

   after configuration:

   - `sudo supervisorctl reread`
   - `sudo supervisorctl update`
   - `sudo supervisorctl start info7390`

8) **Setup nginx:**
   - install nginx: `sudo apt-get install nginx`, do `sudo apt-get update` first if failed
   - cd to the nginx configure folder: `cd /etc/nginx/sites-available`
   - create configuration file: `touch info7390.conf`, and `vim info7390.conf`
   - need to specify the port number we want to listen to, here we are listening to 8001, and redirect to port 8888 as we alreadu set it in the gunicorn configuration
   - start the server: `cd ../sites-enabled`, link to the config file`ln -s ../sites-available/info7390 ./info7390`
   - and then restart nginx: `sudo service nginx restart`

9) **Setup the security rules on Alibaba cloud, for both IN and OUT, allow for port number 8000-9000.**

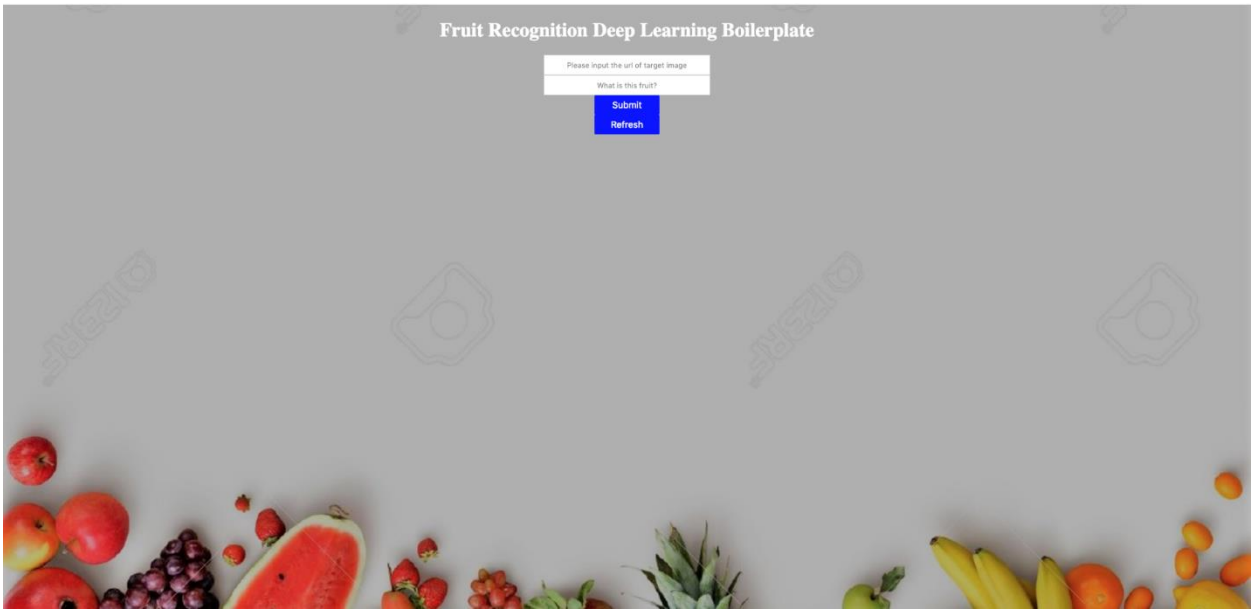
10) **Done!**

4.4 Deployment Details

- Language: Python, HTML, CSS
- Container: Docker
- Cloud Platform: Alibaba Cloud ECS
- Data Analysis Tools: Jupyter Notebook

5. Application Details

5.1 User Interface



- **The first text box:** “Please input the url of target image”, users can enter the URL of the image they want to test here
- **The second text box:** “What is this fruit”, users can enter what the fruit they input is in this text box, our model will compare its prediction result with this value and then decide whether the prediction result is correct or not
- **Submit button:** click submit when users input the URL to see the prediction result
- **Refresh button:** click this button to reload the newest model in Alibaba Cloud Disk

5.2Request and Response

```
28  # pre-process the image for classification
29  image = cv2.resize(image, (100, 100))
30  image = image.astype("float") / 255.0
31  image = img_to_array(image)
32  image = np.expand_dims(image, axis=0)
33
34  # load the trained convolutional neural network and the label
35  # binarizer
36  print("[INFO] loading network...")
37  model = load_model(args["model"])
38  lb = pickle.loads(open(args["labelbin"], "rb").read())
39
```

After loading the input image, it will be pre-processed first.

```
50  correct = "correct" if filename.rfind(label) != -1 else "incorrect"
```

The model will compare the prediction result with the actual value and then return whether the prediction is correct or not.

```
53   label = "{}: {:.2f}% ({})".format(label, proba[idx] * 100, correct)
```

The prediction and comparison result will be returned as following: