Fruit Recognition

Yang Zhang
Brown University
Providence RI
yang_zhang@brown.edu

Stella Xiao
Brown University
Providence RI
weizhi_xiao@brown.edu

Kaiqi Jiang
Brown University
Providence RI
kaiqi_jiang@brown.edu

Jing Qian
Brown University
Providence RI
jing_qian@brown.edu

Abstract

Recognizing fruits on a single RGB camera is challenging in real life scenarios due to complexities of the environment, lack of depth information and variety of perspectives, sizes and colors of the same objects. We developed a cloud-based convolutional neural network (CNN) based interface that allows a local mobile device to perform a three-step fruit recognition procedure using their onboard cameras. Our model supports a total of 29 categories of fruits with both testing accuracy and testing accuracy above 84%.

1. Introduction

Traditional fruit recognition supported by machine learning algorithms can recognize simple scenes with objects from RGB channel images [8]. However, these fruits can be placed in complex environments in real-life that are simply too complex for most traditional methods to perform well. Successful recognizing fruits require an algorithm to robustly extract, segment, and featurize nuance differences among different fruits. For instance, lime and avocado are similar in color and shape but different in texture. In this work, we propose a cloud-based CNN network with a web interface to perform real-time fruit detection on mobile devices.

We first experimented with images database from FIDS30, Fruit 360, ALOI, and ImageNet, totaling over 40,000 images. We then merged the dataset with our self-created data set, which adds 800 data points. Together we used this combined data set for a CNN model, including a preprocessing step that resizes and normalizes the data. We used a Leaky ReLU as the primary activation function for our four convolutional layers in the model. Further twitching of kernel size and hyperparameters allow us to reach 84% test accuracy and 85% validation accuracy.

A web-based interface was created to support front-end interactions. This interface is programmed in HTML and JavaScript for maximum scalability. During runtime, a three-step procedure was designed. First, the interface automatically turns on the onboard camera and ask the user to take a picture of a fruit; then, the user can choose to upload to our cloud CNN model for recognition; finally, they see a list of predictions ordered by likelihood with the most likely fruit's name displayed in a blue, bold font style.

Our system demonstrated the possibility of scaling cloud-based models and running real-time fruit recognition on a modern smartphone. Our contributions are two-fold: 1) a CNN model that can predict the type of a fruit (over 84% accuracy) through real-time image capturing via a smartphone; 2) our self-collected data set with 13 different types of fruit over 800 images.

2. Dataset

We picked 29 categories of common fruits to classify. The complete list of 29 different fruits is: golden apple, red apple, apricot, avocado, ripe avocado, banana, cantaloupe, black cherry, red cherry, blue grape, pink grape, white grape, pink grapefruit, white grapefruit, kiwi, lemon, lime, mandarine, green mango, yellow mango, orange, papaya, peach, pear, pineapple, plum, pomegranate, strawberry, and tomato.

5 datasets in a total of – images were used for training and testing. 4 of them are publicly available on the internet including Fruits 360 [4], Amsterdam Library of Object Images (ALOI) [3], Fruit Image Data Set (FIDS30) [6], and ImageNet [1]. We extracted the images of fruits in our list from those datasets. The fifth dataset is collected by ourselves.

Among them, 75% used for the training set, 5% used for the validation set, and 20% used for the testing set.

2.1. Fruits 360 and ALOI

Both of the datasets are shotted in a controlled environment.

Fruits 360 is a dataset that contains 101 categories of fruits. Each category contains around 600 images and each image is 100×100 pixels with a white background. The images were created by taking frames from a video filmed by rotating around the fruits.

ALOI is an online collection of color images of thousands of small objects including some fruits. Each category contains around 100 images and each image is 768x576 pixels with a black background. The images were created by taking photos of the rotating object.

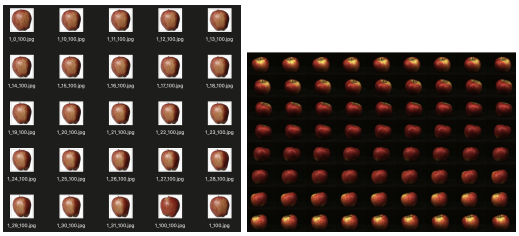

Figure 1. Sample images of red apples from Fruits360 (left) and ALOI (right)

These two datasets capture the signature feature of each fruit very well and are a good start to train the model. However, it is too perfect that the background and light are controlled and there is only one fruit in the middle of every image. Real-world photos are not perfect and usually have a more complex background. So, we need more complex data.

2.2. FIDS30 and ImageNet

Both of the datasets include images that are shotted in different environments.

FIDS30 contains 971 images of 30 different fruits and each type of fruit contains 32 different images. Although all images only contain one type of fruit, a lot of images have more than one fruit of the same type in it. Also, this dataset has various backgrounds.

ImageNet is an image database organized according to WordNet hierarchy. Images from ImageNet have different backgrounds and lighting settings and are more diverse comparing to other datasets.

These two datasets are closer to real-world photos and contributed to making our model more tolerable to different environments to predict better results on real-world photos.

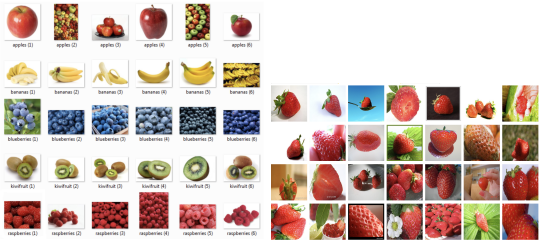

Figure 2. Sample images from FIDS30 (left) and ImageNet (right)

2.3. Self-created dataset

Each member of the group chose some categories of fruits and took 200 photos per chosen fruit. All of the images are non-digitally edited and are in the shape of a square. We photographed photos in different backgrounds, distances to the camera, and lighting settings.

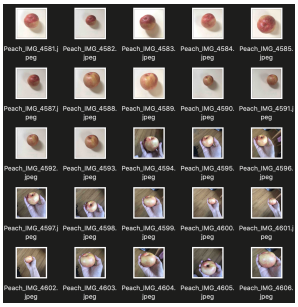

Figure 3. Sample images from our own dataset

3. Model

We apply Convolutional Neural Network(CNN) as our classifier since our dataset contains more than 40000 images, the good performance requires not only a full-fledged neural network model but also a feasible data preprocessing method.

3.1. Implementation details

The training program and model are implemented with PyTorch, we also define our dataset class, data loader (an iterator which yields input images by mini-batch) and data transformations (including operations like resizing, flipping, etc) using torchvision, a library from PyTorch team which contains some useful tools for computer vision tasks.

3.2. Data Augmentation and Normalization

The input size of images varies a lot in our dataset, so first, our training program will take in input images and resize them to a 200*200 image in RGB mode, i.e., the input will be a tensor whose shape is (200,200,3). For training images, we apply a random horizontal flip and a random verti-

cal flip on the image, with a flip probability of 30% and 10% respectively. Then, after converting image (stored in numpy array) to tensor, we apply the normalization step using values of mean and standard deviation in different channels of all images in the dataset, which are computed by a helper function. For validation and test images, these steps for preprocessing are identical to training images, but flipping will not be applied.

3.3. Convolutional Neural Network

The big picture of model architecture is shown in figure below, Leaky ReLU(Leaky Rectified Linear Unit) [7] is the activation function we use in this model. First here comes 4 convolutional blocks, within each block there lies an ordinary convolutional layer, a batch normalization [2] step, and a max pooling layer. The kernel size for convolutional layers is always 3*3, while for max poolings is always 2*2. The number of feature maps(also known as "output channels") is 64, 128, 256, 512 respectively, which means we will have 512 feature maps for each 3-channel (RGB) input image after all convolutional blocks.

Figure 4. The architecture of CNN

Then we apply a dropout on tensors from previous layers and feed them to linear layers(fully connected layers). There are 4 fully connected layers in total, and we apply softmax on outputs of the last one. Between 4 layers there are 3 dropout layers, the dropout mechanism [5] here is believed to be able to force the network to learn some critical information.

During training, at the end of each training epoch, we use validation data to test our trained model so far, once a higher validation accuracy is achieved, the corresponding model parameters will be saved in a global variable. Finally, the model state with the highest validation accuracy will be saved and used for the next period.

3.4. Hyperparameters and experiments

The hyperparameters we used for training the final mode l(which is deployed as the classifier online) are listed below:

We have also set the batch size to 16, but it turned out that the final accuracy is slightly different. The training was carried on a machine with CUDA gpu, and for 48 epochs of training it took around 3.5 hours to finish. The model state was then saved in the .pt file.

Batch Size	Learning Rate	No. of Epochs	Validation Accuracy	Test Accuracy
32	0.0008 (Adam Optimizer)	48	85.531%	84.146%

3.5. Model deployment

After we attained the trained model, we deployed the model to another machine which also supports PyTorch but is without CUDA toolkits, since inference will not take too much computation resources. In inference process, instead of retrieving only the prediction label for the most confident one (i.e., the one with highest softmax probability), we will sort the prediction list by probabilities and find the highest 4 predictions. The highest one (called "top1") is accompanied by three others fruit labels to test the "top4" accuracy on some real world images. We took some photos and download some from Google to form our real world test set, which contains about 40 pictures, and the "top4" accuracy on this test set is 75

4. Website

In order to visualize the output of the project, we created an easy-to-use webpage. Here is a screenshot of the user interface.

Figure 5. Index page on PC browser

The webpage can be viewed consistently on PC, Mac or mobile devices. Though it is an app compatible with multiple platforms, it is designed mainly for mobile phone users. The reason behind this is that people nowadays will always have their mobile phones with them wherever they go. That means when they go traveling they may find something that is new to them possibly including different types of fruits, then they can definitely have a quick idea of what the unknown fruit is by simply taking a photo through the app. There are rare circumstances when they have to use the app via their laptops.

4.1. User interface

In general, we choose blue as the theme color since blue suggests confidence. We are confident with our product which will accurately tell all you need to know.

The box with dashed lines in the middle is the input of the fruit image. There are mainly four ways users can upload their image.

At the bottom of the page, there are references of our group members with respective links of personal info or Github page.

4.1.1 Drag and drop

Users can easily drag and drop an image to the box, then the image will show directly inside the box. Note when a non-image file is dragged including a folder, nothing will happen.

4.1.2 Browse from local directory

Users can also browse an image from a local directory, which is the traditional way of uploading an image from PC or Mac. Similarly, a non-image file is not acceptable.

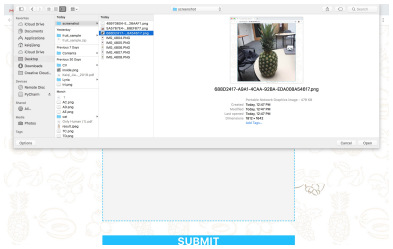

Figure 6. Browsing

4.1.3 Choose from the album

The upload function is also made for mobile phone users. When the box is touched, users can choose a photo from their albums.

4.1.4 Take a photo

This is the most common way for users to upload their images on mobile browser. They can take a picture of the unknown fruit at any time. Specifically, when the box is touched, the built-in camera will pop up, and then the captured image will be shown inside the box.

After the image is uploaded, the fruit image will be processed through the model via the backend. The result will be displayed on the line marked with Best and Also can be after the Submit button is clicked where Best refers to the best prediction of the fruit type, and Also can be refers to the 2nd, 3rd, and 4th highest accuracy of predictions. Since the most valuable result is concerned with Best, the text within that line is highlighted.

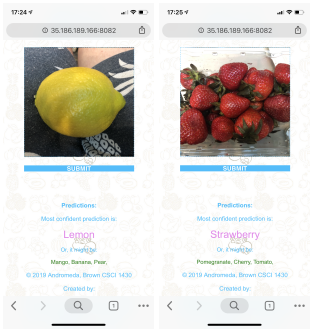

Figure 7. Taking photos

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[3] G. J. Burghouts J. M. Geusebroek and A. W. M. Smeulders. The amsterdam library of object images, 2005. Int. J. Comput. Vision, 61(1), 103-112.

[4] Horea Muresan and Mihai Oltean. Fruit recognition from images using deep learning, 2018. Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[6] Škrjanec Marko. Automatic fruit recognition using computer vision. (Mentor: Matej Kristan), Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

[7] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[8] Yudong Zhang and Lenan Wu. Classification of fruits using computer vision and a multiclass support vector machine. *sensors*, 12(9):12489–12505, 2012.

5. Contribution

Yang Zhang: I implemented the CNN as our classifier model, since I had some computing resources, I was also responsible for model fine-tuning. After three weeks, our final model is good enough to predict most of usual daily fruits in our dataset, and after this achievement, I deployed the saved trained model to web server, so that in the back-end it can support the UI interaction designed by Kaiqi.

Stella(Weizhi) Xiao: The real-world testing result of our initial model is really weak, so we wanted to include more data to improve our performance. I found several different datasets and we decided to use 4 of them as our datasets. Since we decided to only classify in a total of 29 categories of fruits, I extracted those fruits from the datasets from the internet.

Kaiqi Jiang: Visualizing the result is important for our project, so we have come up with a simple user interface instead of running commands via terminal. I was responsible for designing the dashed box structure for users to upload images and fine-tuning the layout of the website that can be viewed consistently both on laptops and mobile phones.

Jing Qian: Helped intellectually on discussion of UI design / implementation and cloud-based model pipeline. Fine-tuned CSS for optimal smartphone display. Wrote supporting script to facilitate data preprocessing. Took images for the collective self-created dataset.