University of Bordeaux

Master 2 Computational Biology

Fruit recognition using Deep Learning

Marie Economides

Professor:
LE Van Linh

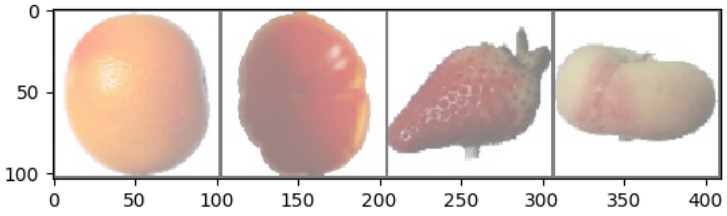

Figure 1:

Contents

Introduction

This report presents an algorithm allowing fruits recognition using deep learning. The dataset used to train and test the model is fruits-360. Some papers based on a deep learning for fruits recognition already exists, [1], [2].

A Convolutional Neural Network has been designed to classify fruits contained in the fruits-360 dataset. The network has been implemented by using Pytorch framework. The network has been tested with different hyper-parameters, i.e update rule, number of epochs and results are presented in the result part.

Method

Deep learning

Deep learning is a class of machine learning algorithms. It is an efficient way to obtain successful results in image recognition and classification. The concept of deep learning is to use artificial neural networks. Each level learns to transform its input data into a slightly more abstract and composite representation. Deep neural network, especially Convolutional Neural Networks have been proved to obtain great results in image recognition. A network is composed by convolutional layers, pooling layers, ReLu layers, fully connected layers and loss layers.

Convolutional neural network

Convolutional neural networks (CNN) are a part of deep learning models. In this type of network, each convolutional layer is followed by a Rectified Linear Unit layer (ReLu) and Pooling layer. Then, the network is composed by one or more convolutional layer and finally one or more fully connected layer.

Convolutional layer

The convolutional layer allows the extraction of features from the input image. It preserves the spatial relationship between pixels by learning image features using small squares of input data. The convolutional layer execute a mathematical operation where two functions produce a third function that is the convoluted version of one of the original function. THe ReLu operation is a function added in order to rectify the output feature map after the operation of convolution.

Pooling layer

The function of pooling continue to reduce the number of parameter and control the problem of overfitting. It has been proved that Max pooling work better. So, in the case of max pooling, a spatial neighborhood can be defined, for example a 2*2 size window. Then,the largest element from the rectified feature map within that window is chosen.

Fully connected layer

The output from the convolutional and pooling layers represent high-level features of the input image. The Fully Connected layer use these features for classifying the input image into various classes based on the training dataset.

Loss Layer

Loss layer is the last layer of the network. This layer is used to penalize the network for deviating from the expected output during the training. It exists different loss function as softmax. Softmax is used for predicting a class from multiple disjunct classes. Another function is the sigmoid cross-entropy and is used to predict multiple independent probabilities, [2]

Name of Fruits	Varieties
Apples	Golden, Golden-Red, Granny Smith, Red, Red Delicious
Apricot	
Avocado	
Avocado ripe	
Banana	Yellow, Red, Lady Finger
Cactus fruit	
Cantaloupe	2 varieties
Carambula	
Cherry	Rainier, etc
Cherry Wax	Yellow, Red, Black
Chestnut	
Clementine	
Cocos	
Dates	
Granadilla	
Grape	Blue, Pink, White
Grapefruit	Pink, White
Guava	
Hazelnut	
Huckleberry	
Kiwi	
Kaki	
Kumsquats	
Lemon	normal, Meyer
Lime	
Lychee	
Mandarine	
Mango	
Mangostan	
Maracuja	
Melon Piel de Sapo	
Mulberry	
Nectarine	
Orange	
Papaya	
Passion fruit	
Peach	different varieties
Pepino	
Pear	Abate, Kaiser, Monster, Williams
Physalis	normal, with Husk
Pineapple	normal, Mini
Pitahaya Red	
Plum	different varieties
Pomegranate	
Pomelo Sweetie	
Quince	
Rambutan	
Raspberry	
Redcurrant	
Salak	
Strawberry	normal, Wedge
Tamarillo	
Tangelo	
Tomato	Maroon, Cherry Red
Walnut	

Table 1: fruits included in the dataset

Data set

Fruits 360 dataset, is available on kaggle [3] and github [4]. The dataset contains a total numbers of 65 429 images with 48 905 images in the training dataset and 16 421 images in the test dataset. Each image contains only one fruit per image and sizes is 100 * 100 pixels. The fruits included in the dataset are presented in the table 2.

Material

Pytorch

In order to classify fruits, Pytorch framework has been used for this project. PyTorch is an open source machine learning library for Python, based on Torch. PyTorch provides Tensor computation, like NumPy, with strong GPU acceleration.

Steps of classification

Five steps were executed to classify fruits.

1. Load data for the training and tests with torchvision
2. Define the structure of the Convolutional Neural Network
3. Define a loss function
4. Train the network on the training data
5. Test the network on the test data

Structure of neural network

The implemented neural network is composed by 2 convolutional layers with a kernel size of 5*5.

Layer Type	Dimension	Output
Convolutional	5*5*3	6
Max pooling	2*2	-
Convolutional 5*5*6	16	
Max pooling	2*2	-
Fully connected	16*22*22	120
Fully connected	120	84
Softmax	84	83

Table 2: Structure of neural network

Results

epochs=2, lr=0.01

```
cpu
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=7744, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=83, bias=True)
)
[1, 2000] loss: 4.401
[1, 4000] loss: 4.411
[1, 6000] loss: 4.413
[1, 8000] loss: 4.414
[1,10000] loss: 4.410
[2, 2000] loss: 4.412
[2, 4000] loss: 4.413
[2, 6000] loss: 4.413
[2, 8000] loss: 4.414
[2,10000] loss: 4.412
Finished Training
('Predicted:', 'tensor(19) tensor(9) tensor(67) tensor(29)')
Accuracy of the network on the 10000 test images: 2 %
```

Figure 2:

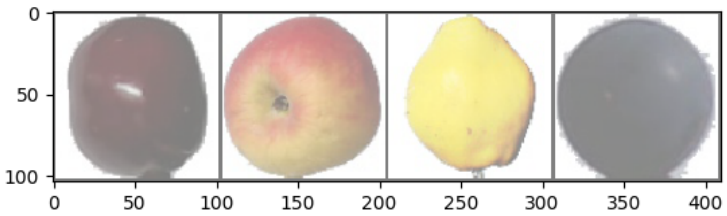

Figure 3:

epochs=10, lr=0.01

```
cpu
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=7744, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=83, bias=True)
)
[1, 2000] loss: 3.838
[1, 4000] loss: 2.277
[1, 6000] loss: 2.203
[1, 8000] loss: 2.145
[1,10000] loss: 2.206
[2, 2000] loss: 2.374
[2, 4000] loss: 2.774
[2, 6000] loss: 3.235
[2, 8000] loss: 3.039
[2,10000] loss: 3.347
[3, 2000] loss: 3.022
[3, 4000] loss: 3.239
[3, 6000] loss: 3.959
[3, 8000] loss: 4.038
[3,10000] loss: 4.420
[4, 2000] loss: 4.412
[4, 4000] loss: 4.414
[4, 6000] loss: 4.414
[4, 8000] loss: 4.414
[4,10000] loss: 4.411
[5, 2000] loss: 4.412
[5, 4000] loss: 4.411
[5, 6000] loss: 4.412
[5, 8000] loss: 4.416
[5,10000] loss: 4.412
```

```
[6, 2000] loss: 4.413
[6, 4000] loss: 4.414
[6, 6000] loss: 4.413
[6, 8000] loss: 4.411
[6,10000] loss: 4.411
[7, 2000] loss: 4.407
[7, 4000] loss: 4.416
[7, 6000] loss: 4.413
[7, 8000] loss: 4.414
[7,10000] loss: 4.414
[8, 2000] loss: 4.410
[8, 4000] loss: 4.415
[8, 6000] loss: 4.414
[8, 8000] loss: 4.411
[8,10000] loss: 4.412
[9, 2000] loss: 4.413
[9, 4000] loss: 4.413
[9, 6000] loss: 4.413
[9, 8000] loss: 4.414
[9,10000] loss: 4.408
[10, 2000] loss: 4.410
[10, 4000] loss: 4.413
[10, 6000] loss: 4.413
[10, 8000] loss: 4.414
[10,10000] loss: 4.412
Finished Training
('Predicted:', 'tensor(82) tensor(28) tensor(81) tensor(64)')
Accuracy of the network on the 10000 test images: 2 %
```

Figure 4:

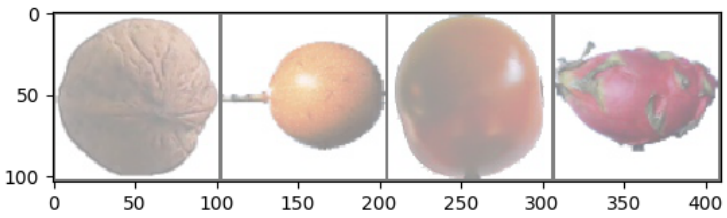

Figure 5:

epochs=2, lr=0.001

```
cpu
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=7744, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=83, bias=True)
)
[1, 2000] loss: 3.610
[1, 4000] loss: 1.820
[1, 6000] loss: 1.069
[1, 8000] loss: 0.798
[1,10000] loss: 0.630
[2, 2000] loss: 0.461
[2, 4000] loss: 0.462
[2, 6000] loss: 0.256
[2, 8000] loss: 0.329
[2,10000] loss: 0.226
Finished Training
('Predicted:', 'tensor(78) tensor(11) tensor(73) tensor(67)')
Accuracy of the network on the 10000 test images: 78 %
```

Figure 6:

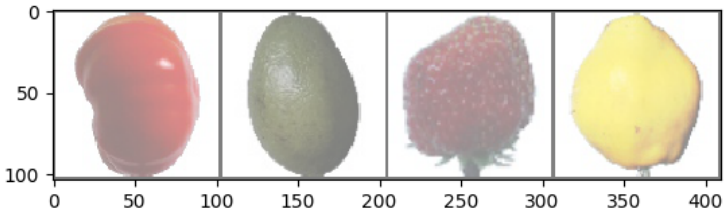

Figure 7:

epochs=10, lr=0.001

```
cpu
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=7744, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=83, bias=True)
)
[1, 2000] loss: 3.512
[1, 4000] loss: 1.965
[1, 6000] loss: 1.194
[1, 8000] loss: 0.798
[1,10000] loss: 0.535
[2, 2000] loss: 0.365
[2, 4000] loss: 0.264
[2, 6000] loss: 0.330
[2, 8000] loss: 0.230
[2,10000] loss: 0.145
[3, 2000] loss: 0.247
[3, 4000] loss: 0.191
[3, 6000] loss: 0.104
[3, 8000] loss: 0.145
[3,10000] loss: 0.144
[4, 2000] loss: 0.128
[4, 4000] loss: 0.090
[4, 6000] loss: 0.216
[4, 8000] loss: 0.108
[4,10000] loss: 0.075
[5, 2000] loss: 0.062
[5, 4000] loss: 0.144
[5, 6000] loss: 0.145
[5, 8000] loss: 0.119
[5,10000] loss: 0.017
```

```
[6, 2000] loss: 0.009
[6, 4000] loss: 0.011
[6, 6000] loss: 0.008
[6, 8000] loss: 0.011
[6,10000] loss: 0.021
[7, 2000] loss: 0.011
[7, 4000] loss: 0.009
[7, 6000] loss: 0.007
[7, 8000] loss: 0.008
[7,10000] loss: 0.011
[8, 2000] loss: 0.009
[8, 4000] loss: 0.007
[8, 6000] loss: 0.009
[8, 8000] loss: 0.009
[8,10000] loss: 0.008
[9, 2000] loss: 0.009
[9, 4000] loss: 0.008
[9, 6000] loss: 0.007
[9, 8000] loss: 0.008
[9,10000] loss: 0.008
[10, 2000] loss: 0.007
[10, 4000] loss: 0.008
[10, 6000] loss: 0.007
[10, 8000] loss: 0.007
[10,10000] loss: 0.009
Finished Training
('Predicted:', 'tensor(30) tensor(79) tensor(11) tensor(20)')
Accuracy of the network on the 10000 test images: 93 %
```

Figure 8:

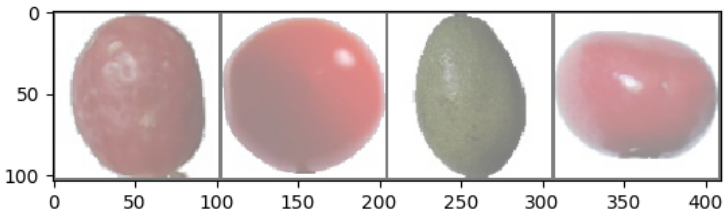

Figure 9:

Conclusion and further work

It is possible to see that when lr=0.01, however epochs are used, the accuracy is bad.
When lr=0.001, it's possible to remark that the number of epochs can have an influence on the accuracy. Also, the lowest is the loss, the better it is for accuracy. When loss>1, then the accuracy is bad.

In this project, it has been possible to design of a CNN to classify Fruits-360 dataset. The network has been implemented using PyTorch framework. Results obtained using different hyper-parameters show that the number of epochs used or variation of update rules can change loss and accuracy.

In order to improve the model, it should be interesting to purpose a new structure of neural network. It already has been done in [2]. It will be interesting to have more than two convolutional layers and compare results and accuracy.

References

[1] L. Hou, Q. Wu, Q. Sun, H. Yang, and P. Li, "Fruit recognition based on convolution neural network," *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 18–22, 2016.

[2] H. Muresan and M. Oltean, "Fruit recognition from images using deep learning," *CoRR*, vol. abs/1712.00580, 2017.

[3] H. Muresan and M. Oltean, "Fruit recognition from images using deep learning kaggle @ONLINE," 2019.

[4] H. Muresan and M. Oltean, "Fruit recognition from images using deep learning github @ONLINE," 2019.