University of Bordeaux

Deep Learning Project

Master Degree Of Bioinformatics

Conception of a Convolutional Neural Network to classify fruits

Author:
Hans Schrieke

Supervisor:
Van Linh Le

February 17, 2019

1 Introduction

During the last years, **Machine Learning** is very developed in many domains and particularly for data mining. It is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

Deep Learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input
- learn in supervised (classification) and/or unsupervised (pattern analysis) manners
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts

Neural Networks are a set of deep learning algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. They are very efficient to cluster and classify.

In this project, we used a category of Neural Networks that have proven very good for image recognition and classification : the **Convolutional Neural Networks** (CNN). They have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars for example.

We want to design a convolutional neural network able to recognize and classify fruits from a fruits images dataset with a correct accuracy.

2 Material

2.1 Dataset

The dataset used during this project contains the images of 95 types of fruits. It comes from the online community of data scientists and machine learners **Kaggle**[1].

2.2 Software

The open source deep learning **Pytorch** 1.0 [2] library has been used to create the convolutional neural network with different packages : **torchvision**[3] that consists of popular datasets, model architectures, and common image transformations for computer vision and **numpy**[4] that is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

The program has been written in Python 3.7.

3 Method

3.1 Structure of a CNN

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, ReLU layer (activation function), pooling layers and fully connected layers.

3.1.1 Convolutional layer

Convolution is a mathematical operation to merge two sets of information. One or more filter is used to extract features from the input and result is a set of filtered images. Each image corresponds to a particular filter. The convolution operation is running by sliding the filter over the input image. At every location, there is a element-wise matrix multiplication and sum the result. The sum goes into the feature map which is the result of convolution step. We then slide the filter and perform the same operation.

3.1.2 ReLU

For any king of neural network to be powerfull, it needs to contain non-linearity.

3.1.3 Max pooling

The most common type of pooling is max pooling which just takes the max value in the pooling window. It's a method to take a large image and reduce its size while preserving the most important information it contains. Because it keeps at each step the maximum value contained in the window, it preserves the best feature of this window. The output will have the same number images but each image will have a lower number of pixels.

3.1.4 Fully connected layers

Instead of treating inputs as 2-dimensional arrays, they are treated as a simple list and identically. This layer uses these features for classifying the input image into various classes based on the training dataset.

3.2 Structure of CNN used

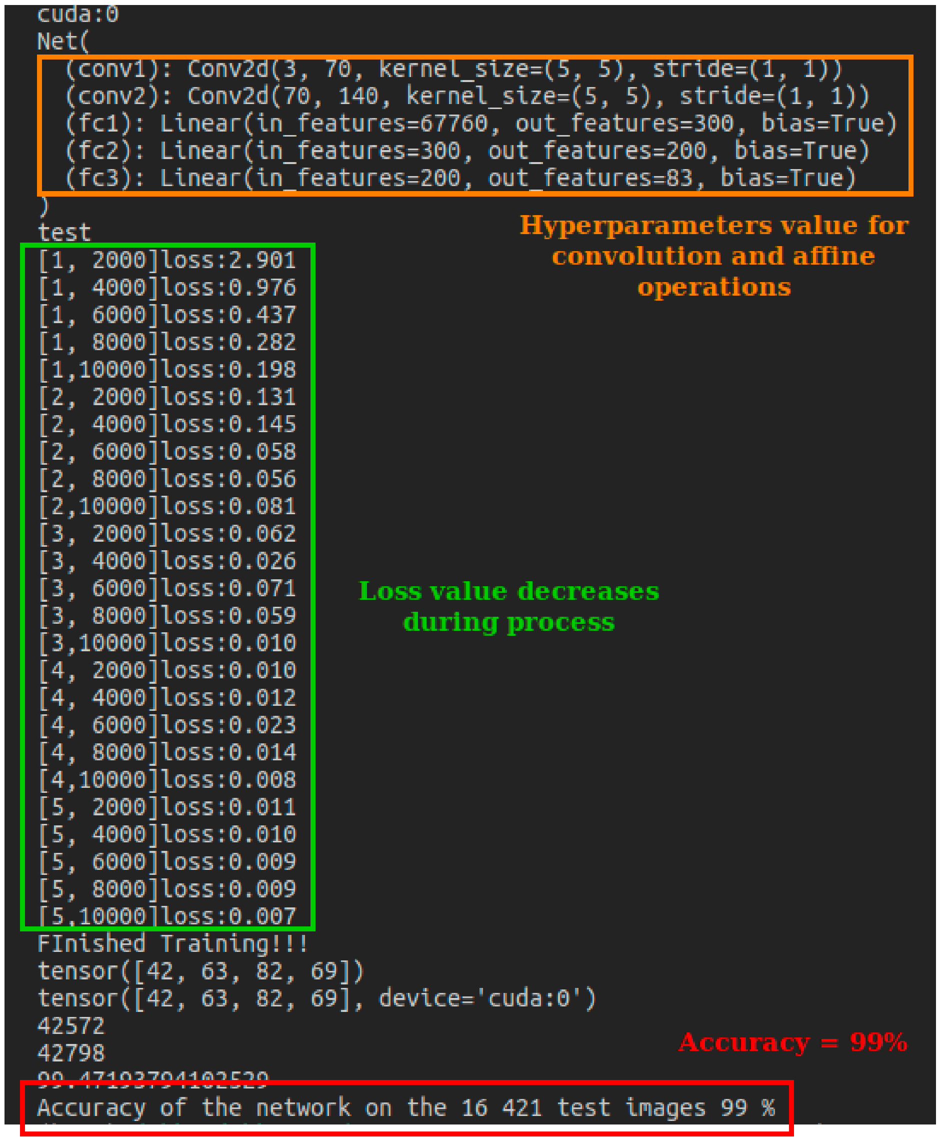
There are different ways to construct a CNN. For this project, I choose to do two convolution layers followed by ReLU and max pooling operations.

After making the CNN, a function allows to train it to recognize the fruits. And finally, a test of CNN shows the accuracy of the model.

4 Results

Depending on the hyperparameters used, the accuracy obtained is different. In fact, with low values of input and output channels in the convolution layers, and low value of size of input and output in affine operations, we obtain a low accuracy : 2%.

By increasing these values a lot, the accuracy reaches a very good value: 99%. It means there is 99% chance that the fruit recognized by the program is well the fruit that we have given as input. Moreover, the loss values obtained during the training are low, it means the fluctuations are well reduced during the process : it helps obtaining a better performance because there is a very low data loss.



```
cuda:0
Net(
  (conv1): Conv2d(3, 70, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(70, 140, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=67760, out_features=300, bias=True)
  (fc2): Linear(in_features=300, out_features=200, bias=True)
  (fc3): Linear(in_features=200, out_features=83, bias=True)
)
test
[1, 2000]loss:2.901
[1, 4000]loss:0.976
[1, 6000]loss:0.437
[1, 8000]loss:0.282
[1,10000]loss:0.198
[2, 2000]loss:0.131
[2, 4000]loss:0.145
[2, 6000]loss:0.058
[2, 8000]loss:0.056
[2,10000]loss:0.081
[3, 2000]loss:0.062
[3, 4000]loss:0.026
[3, 6000]loss:0.071
[3, 8000]loss:0.059
[3,10000]loss:0.010
[4, 2000]loss:0.010
[4, 4000]loss:0.012
[4, 6000]loss:0.023
[4, 8000]loss:0.014
[4,10000]loss:0.008
[5, 2000]loss:0.011
[5, 4000]loss:0.010
[5, 6000]loss:0.009
[5, 8000]loss:0.009
[5,10000]loss:0.007
FInished Training!!!
tensor([42, 63, 82, 69])
tensor([42, 63, 82, 69], device='cuda:0')
42572
42798
99.47193794102529
Accuracy of the network on the 16 421 test images 99 %
```

Hyperparameters value for convolution and affine operations
Loss value decreases during process
Accuracy = 99%

Figure 1: Results of convolution neural network with fruits dataset

References

[1] https://www.kaggle.com/

[2] https://pytorch.org/

[3] https://pytorch.org/docs/stable/torchvision/index.html

[4] http://www.numpy.org/