université de BORDEAUX

Master in Bioinformatics

Deep Learning

Fruit image predictions with Convolutional-Neural-Network

BOCK Pauline

February, 15th of 2019

Outline

Introduction

Nowadays, we are in a world of huge amount of data . All these data allowed to create and use new algorithms to classify and/or predict new data, thanks to deep learning for instance.

Deep learning regroups several ways to predict data giving a lot of similar data including Convolutional Neural Network (CNN). This last is often used for prediction of images, for example what are the things, animals or people in an image.

To do that , there are several models and layers that can be implemented. Indeed, a CNN is working as next:
Firstly you need to create an input layer with your training data,
secondly you add several layers of treatment that can be convolutional layers or pool layers, and then fully connected layers, densenet and/or softmax,
thirdly at the end of the last layers, there is a final layer containing the output data.
These steps were the forward part of CNN. During the sequence of layers, some parameters are adjusted to have the better prediction for the next part. These parameters are updated at the end and used again to the forward propagation , this updating of parameters is the backward propagation.

The aim of this project is to use fruit pictures and CNN to predict the kind of fruit appearing in a new picture with fruit. The prediction will need to be the best possible, testing several models and parameters to have the best accuracy.

1 Material

1.1 Dataset

The dataset of this project is a set of 83 types of fruit images for the training and test dataset. The images are in jpg format and have a size of 100 per 100 pixels. They are showing one fruit per image and are regroup according to the type of fruit in repositories. There are 48 905 images in the training dataset and 16 421 for the test dataset. This dataset is coming from another project (Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.)

1.2 Language and packages

To compute the CNN, scripts will be made in Python language with the use of Pytorch Framework.

Figure 1: Pytorch logo.

Pytorch is a python framework allowing to train and test several deep learning models rather easily. It is used by some big plants (like Twitter for example) and researchers. It can run on gpu to make more quickly computations.

2 Methods

2.1 CNN Algorithm

Before testing several models, a script has been implemented to create an new model to use at the beginning before making different parameters test.

To implement a CNN, data must be first imported and formatted to be used by the Pytorch library. The amount of data being huge for both training and testing part, it needed to be separated in several sets of images, the number of images in a set defined by the batch size variable. Batch size shouldn't be too small to avoid the decrease of the estimate of the gradient accuracy, and will be limited by the computer memory.

Then a network has to be first written, giving the order of the layers and the layers used, informing about the data and its size, and the number of features.

The network is initialized and the Pytorch process is set to run with the GPU instead of CPU to win some execution and computing time.

The loss function and the optimizer (precising the learning rate) are then given.

After that, the forward and backward process are launched several times, the number of iterations giving by the epoch variable. As the number of epochs increases, the weight is changed and adjusted in the neural network and the model goes from underfitting to overfitting. So number of epochs needs to be scaled.

So for this number of epochs, and for the number of data used for training, the layers are computed as well as bias and weight parameters, and the loss function. Then the backward propagation took place, updating the parameters, computing the derivatives (gradient descent).

The training is then accomplished and the test is launched on test data, computing the prediction and the accuracy of these predictions. For the first 10 images that were passed to the test, their types are printed and the prediction done on it is printed too. It allows to have a little appreciation of the effectiveness of the model.

Another script is used, coming from Pytorch models tutorials and modified to test different parameters and to be launched on the fruit dataset. This script is used to test some already created models of CNN, and work as the previous described above in the main lines, but the different networks models have just to be called by Pytorch models module.

2.2 The different models tested

Several parameters, functions and layers can be changed to adjust the model and find the best accuracy of prediction. There are two ways to design a model: create a new architecture network or use already created ones (which is called Transfer learning).

Only one self created architecture has been tested and two transfer learning have been done.

The new architecture was composed of one convolution layer followed by a max pooling, then another convolution is done followed by a new max pooling, and to finish, three fully

connected layers are added.

The two already created models used were AlexNet model and ResNet model. Their network architectures can be seen in the next figures:

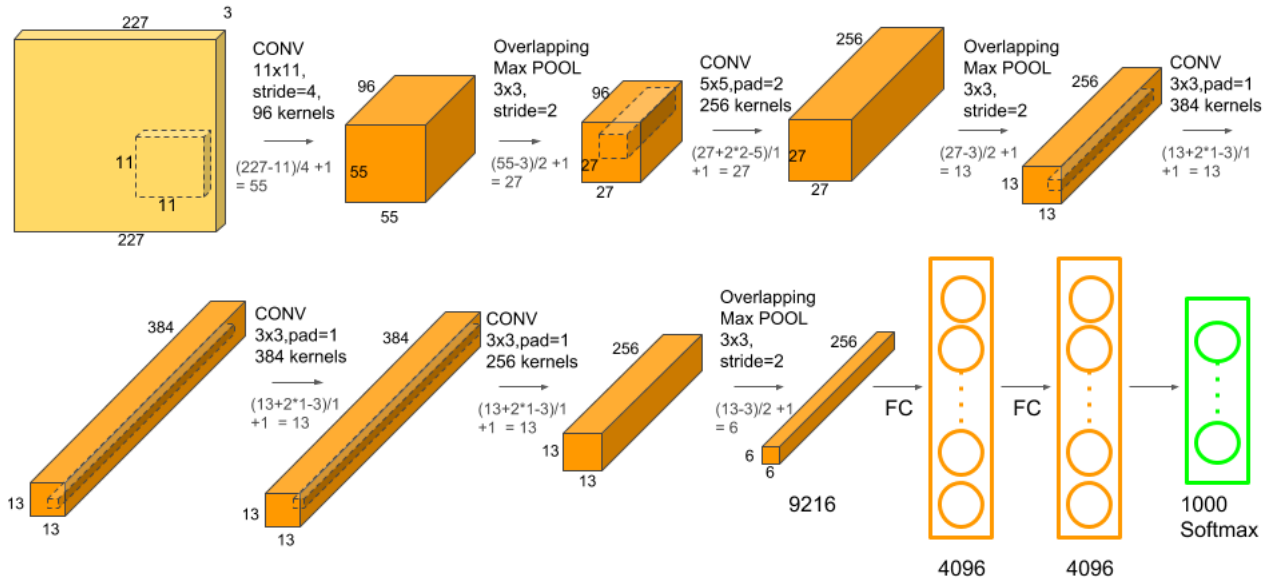

Figure 2: Diagram of AlexNet architecture. "CONV" is for convolution and "FC" for Fully Connected.

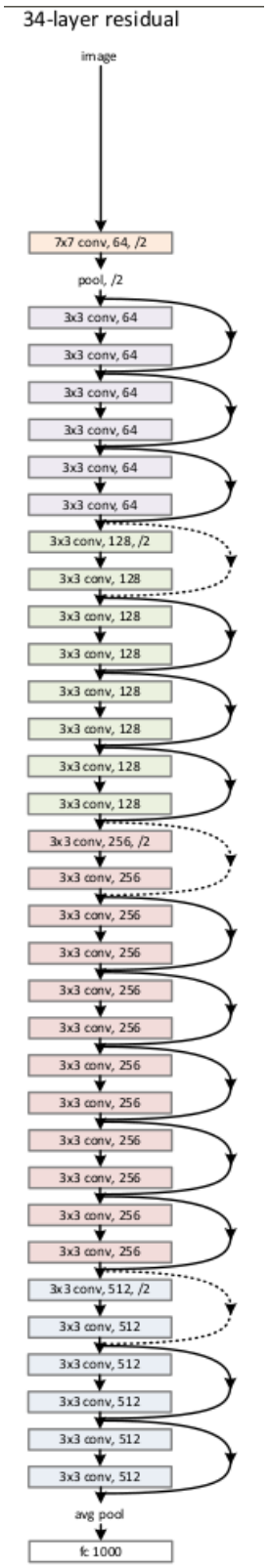

Figure 3: Diagram of ResNet architecture.

The different combinations that have been done can be seen in the next table:

Loss function	Optimizer - Learning Rate	Epochs	Batch size	Layers
New Architecture				
Cross Entropy Loss	SGD (0.001)	2	100 train 80 test	Conv -> MP -> Conv -> MP -> FC * 3
	SGD (0.001)	5		
	SGD (0.001)	10		
	SGD (0.01)	10		
	Adam (0.01)	10		
	Adam (0.001)	10		
Cross Entropy Loss	SGD (0.01)	10	300 train 150 test	Conv -> MP -> Conv -> MP -> FC * 3
	SGD (0.001)	10		
	Adam (0.001)	10		
Transfer Learning				
Cross Entropy Loss	Adam(0.001)	10	4	AlexNet
	SGD (0.01)	10	4	AlexNet
	Adam(0.001)	10	4	ResNet

Figure 4: Table showing the different combinations tested. It is divided into 2 parts, the new architecture tests were done with the selfCNN.py script, and the Transfer Learning part with Transfer_Learning.py script.
For the layers, "Conv" means convolution, "MP" is for Max Pooling, and "FC" is for fully connected.

Only the ReLU (Rectified Linear Unit) will be used as an activation function, like it is faster than tanH function for hidden layers.

Furthermore, only cross entropy loss function have been tested. The batch size for Transfer Learning is lesser than for the new architecture, as the models had more layers and were more slow and more memory expensive.

3 Results and discussion

The accuracy for each test model can be seen in the next table:

Loss function	Optimizer - Learning Rate	Epochs	Time	Batch size	Layers	Accuracy
New Architecture						
Cross Entropy Loss	SGD (0.001)	2	<60 min	100 train 80 test	Conv -> MP -> Conv -> MP -> FC * 3	0.59*
	SGD (0.001)	5	<60 min			0.73*
	SGD (0.001)	10	~30min			0.81*
	SGD (0.01)	10	~30min			0.93*
	Adam (0.01)	10	~30min			0.78
	Adam (0.001)	10	~30min			0.94*
Cross Entropy Loss	SGD (0.01)	10	~60min	300 train 150 test	Conv -> MP -> Conv -> MP -> FC * 3	0.88
	SGD (0.001)	10	~60min			0.69
	Adam (0.001)	10	~60min			0.93*
Transfer Learning						
Cross Entropy Loss	Adam(0.001)	10	185 min	4	AlexNet	0.96
	SGD (0.01)	10	>185min	4	AlexNet	0.95
	Adam(0.001)	10	332 min	4	ResNet	0.89

Figure 5: Table showing the results according to the different combinations tested. It is divided into 2 parts, the new architecture tests were done with the selfCNN.py script, and the Transfer Learning part with Transfer_Learning.py script.
Time and Accuracy columns have been added. For accuracy, the green value is the best founded and the red is the worst. A "*" symbol next to accuracy values means that it is a mean value and that the model have been tested 3 times.

Transfer learning takes more time than the new architecture tests, as the AlexNet and mainly ResNet models have more layers in their network. For this reason, they have not been tested several times to do a mean value of accuracy.

Number of epochs seems to be really important, as it is showed between the first and the third model tested. The 2 epoch model had an accuracy value of only 59% and that was the lowest accuracy values between all the combinations tested.

For the optimizer, it seems that SGD works better with a learning rate of 0.01 than with a learning rate of 0.001, unlike Adam that is the opposite.

Five combinations have shown an accuracy value equal or higher than 93% which seems a great value

As one can see, the best tested model is the AlexNet architecture with CrossEntropy Loss function and Adam optimizer with a learning rate of 0.001. Indeed, its accuracy has reached 96%.

In spite of the low number of epochs and batch size chosen, the accuracy values seems rather high and so the models tested seems rather effective to recognize the type of the fruits in an image. It could be possible that these results are like that because of the dataset. The dataset used is rather big, and is coming from 360 degrees photos of each fruit, without a background and without noise. Moreover, the test images really look like the training ones. Another possibility can be the batch size, that is rather low because of the computer memory limits and the lack of time.

Conclusion

Some models of CNN, newly or already created, have been tested to recognize type of fruits, using 360-fruits dataset. One model have been more effective than the other ones, with an accuracy of 96%.

It could be interested for a next part to have some new test images, with some unclear background or with some noise for example, to see if the specifity of the model is not too high and if it can recognize a type of fruit in a more difficult picture. So, it could be also interested to compute specificity and sensibility of each models, to do more statistics and better qualified them.

Bibliography

Data :
Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, (2018).

Method:
SAGAR SHARMA, Epoch vs Batch Size vs Iterations, Towards Data Science website, Sep 23, (2017).
https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Convolutional Neural Nets in PyTorch, Algorithmia website, april 10, (2018).
https://blog.algorithmia.com/convolutional-neural-nets-in-pytorch/

David Mack, Octavian, How to pick the best learning rate for your machine learning project, Medium website, Apr 9, (2018).
https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

Nathan Inkawhich, Finetuning Torchvision Models, Pytorch website
https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

Krizhevsky, Alex. "One weird trick for parallelizing convolutional neural networks." arXiv preprint arXiv:1404.5997 (2014).

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. (2016).

Figures:
SUNITA NAYAK, Understanding AlexNet, Learn OpenCV website, June 13, (2018).

Baiba Opule image copyright owner (Pinterest).