TP RF : Convolutionnal Neural Network (CNN)

Hala DJEGHIM

December 19, 2021

Please find our implementation in this link https://github.com/hala-djeghim/TP-RF-ImageClassification/blob/main/Image_classification.ipynb

1 Question 1

Ideal epoch number is 30 epochs, after this number of iteration the model overfitt i.e the loss increases and the accuracy decreases.

2 Question 2

After adding a validation set we notice that the model don't overfit as quickly it overfitted before adding it.

3 Question 3

The scores evolution after each network definition is showed in 1

We first changed the output channels of each convolutionnal layer, by lowering it, the higher accuracy score was **0.5466**, while by increasing it the accuracy reached **0.6581**. The initial accuracy before modifying the architecture was **0.5927**.

We then defined another network by adding a third convolutionnal layer, and changing the kernel size and the padding, we choose **3x3** as a kernel size and **1** as padding. The network didn't reach a higher accuracy but it converged in a more "stable" way.

Since our last network definition seemed to give our model stability, we choose to define a CNN with 3 convolutionnal blocks, and a fully connected layer, where each convolutionnal block had two conv layers, each one followed by a Relu and then a a max pooling layer at the end of the block. This network reached the higher score so far, **0.6918**, but it converged quickly, even tho the loss and the accuracy had more stable variations.

4 Question 4

Kernels output the of the first layer. 2

5 Question 5

We first tried to fine tune squeezenet to CIFAR10 accuracy with a learning rate equal to **0.01**, the training and validation loss had both **nan values** at each epoch ! The accuracy stayed the same (0.10). We found out that the parameters of the model were all equal to nan values, after each epoch, so the issue was certainly the learning rate, it was way too high for the model. We lowered it to **1e-8** and the problem was solved, but the model needed way more epoch to converge, we trained it with 200 epochs, and it only reached as accuracy, it certainly needs way more iterations to converge.

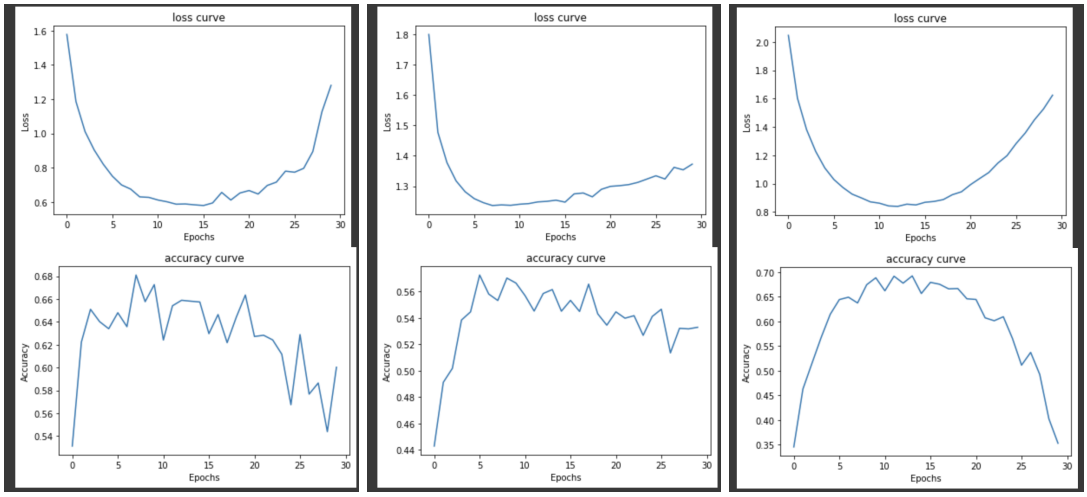

Figure 1: Improvements results at CNN definition, evolution from left to right.

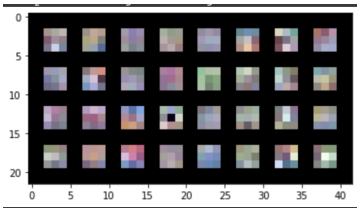

Figure 2: Kernel output

6 BONUS : Results improvements

The goal of this section is to get the more stable and "beautiful" curves of the loss and the accuracy. To do so, we used some very known strategies : Data augmentation, Scheduler, Dropout and batch normalization.
The scores evolution after each step is defined in 3

6.1 Data augmentation

Data augmentation helps increasing the number of the samples so that we have more data to train the model on. We did a horizontal, vertical and random rotation flips. We can notice a small improvement in the scores.

6.2 Optimizer

We used a scheduler to modify the learning rate during training by decreasing it, the learning rate starts from **0.1** and decreases

We initially trained the model with 30 epochs, we noticed that it needed more epochs, so we tried with 50 epochs. The model is way more stable and don't overfit as quickly as before, and it reaches a higher accuracy score **0.7986**, it still need a higher epochs number to overfit.

6.3 Dropout

The number of parameters increases at each layer, the model is more likely to overfit with a huge number of parameters To remedy this problem, we are going to add a layer allowing a very efficient regularization for neural networks : the dropout layer. The general principle of this layer is to randomly deactivate at each pass a part of the neurons of the network to artificially reduce the number of parameters. The accuracy reached was **0.8011**.

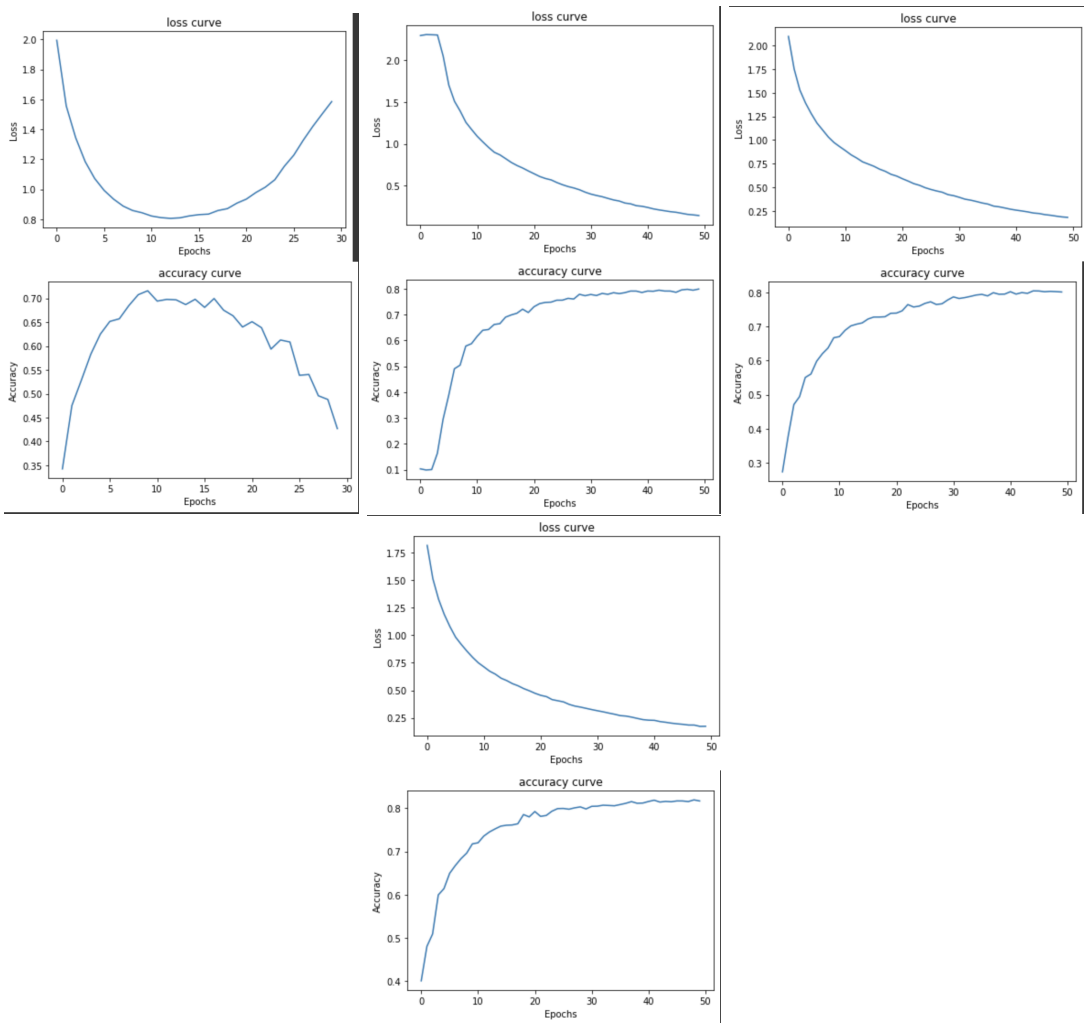

Figure 3: Improvements results at each step, evolution from left to right : Data augmentation, Scheduler, Dropout and Batch normalization

6.4 Bath normalization

This is a layer that learns to renormalize the outputs of the previous layer, allowing to stabilize the learning. A higher score was reached more quickly then without this layer !