Image classification on CIFAR10 with Convolutional Neural Networks

CSC_5RO13 TP1

David FILLIAT

2 janvier 2025

1 Introduction

In this practical work, we will build different deep neural network structures to work on image classification with the CIFAR10 dataset. This dataset (figure 1) contains small images (32x32 pixels) of objects from ten classes (plane, car, bird ...). The aim is to understand the structure of convolutional networks and the influence of parameters on learning.

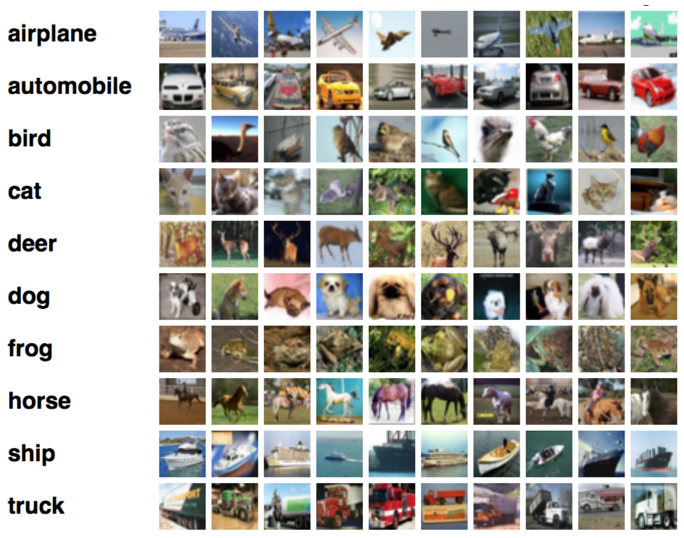

FIGURE 1 – Some images from CIFAR 10

To do this, we will use the pytorch library and the Google Colab platform for execution using this Jupyter Notebook : https://colab.research.google.com/drive/12wJgvLSYqlxzfNYzSmi4iSA15LNjj-r5?usp=sharing.

Follow the first two sections of the notebook (I. What is pytorch, and II. Training a classifier on CIFAR10) to understand the basics of pytorch and how to build and train a network on the CIFAR10 dataset. Then, follow the instructions below to complete the practical work in the third section of the notebook. Note that some of the parameters that you have to modify are in the section II.

2 Question 1 - Hyperparameters tuning

Modify the default parameters in the notebook to get the best possible performance on the validation set. You can play on :

1. `learning_rate` : the step size of the gradient descent. A too high value can lead to divergence, a too low value can lead to slow convergence.
2. `n_epochs` : the number of epochs. A too low value can lead to underfitting, a too high value can lead to overfitting.
3. `batch_size` : the number of examples used to compute the gradient. A too low value can lead to slow convergence, a too high value can lead to divergence.
4. `n_training_samples` : the size of the dataset. The default code uses a dataset of 20,000 examples.
5. `momentum` : the momentum of the gradient descent, when changing the gradient descent method in the `createLossAndOptimizer` function. A too high value can lead to divergence, a too low value can lead to slow convergence.
6. `weight_decay` : add weight decay to the loss function, and tune the coefficient.

3 Question 2 - Network structure

The structure of the network is defined in the `MyConvolutionalNetwork` class in the notebook. Modify this class to add new convolutional layer. Many architectures are possible (number of features at each layer, pooling or not, stride or not, use of fully connected layer or not...). You must build structures that respects the input size constraints (32x32x3 images) and the output size constraints (vector of size 10). Each time, test the performance of the network on the validation set with training on a small dataset. When you have found a good structure, train the network on the full dataset and compare the performance with the previous network.

You can try to :

1. Modify the kernel size of the convolutional layers
2. Modify the number of features of the convolutional layers
3. Add more convolutional layers
4. Add more fully connected layers
5. Add batch normalization layers
6. Add residual connections

The current state of the art on CIFAR10 is around 99.5% accuracy (https://paperswithcode.com/sota/image-classification-on-cifar-10). With a simple CNN, you should be able to reach at least 90% accuracy.