Teachers: J. Hennebert, M. Melchior
Assistants: C. Gisler, M. Dia

Practical work 08 – 7th of November 2019
Convolutional Neural Networks with Keras

Objectives

The first objective of this PW is to understand the principles of Convolutional Neural Networks (CNN). Another objective is to experiments with CNN using Keras on top of TensorFlow.

As for the last practical work, we ask you to submit the solution for next week.

Submission

— **Deadline** : Monday 18 November, 12am (noon)

— **Format** : Zip with report and iPython notebook.

Exercise 1 Computation of convolutions

Revise the slides on convolution layer and make sure you understand the principles and the different steps that were described.

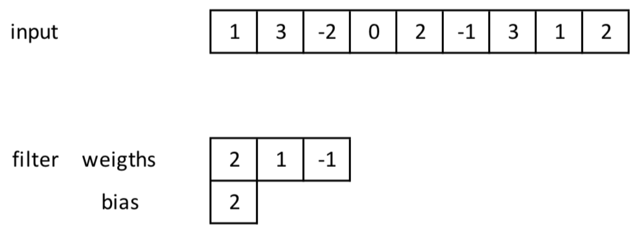


FIGURE 1 – 1D convolution

a) Given the 1D example in Figure 1 compute the output of the convolution by hand considering the following stride and padding values :

— $S = 1$, $P = 0$

— $S = 2$, $P = 0$

— $S = 4$, $P = 0$

— $S = 1$, $P = 1$

— $S = 4$, $P = 1$

In the previous examples, for which values of S and P do we get an output of same dimension as the input ?

b) Given the 2D example in Figure 2 :

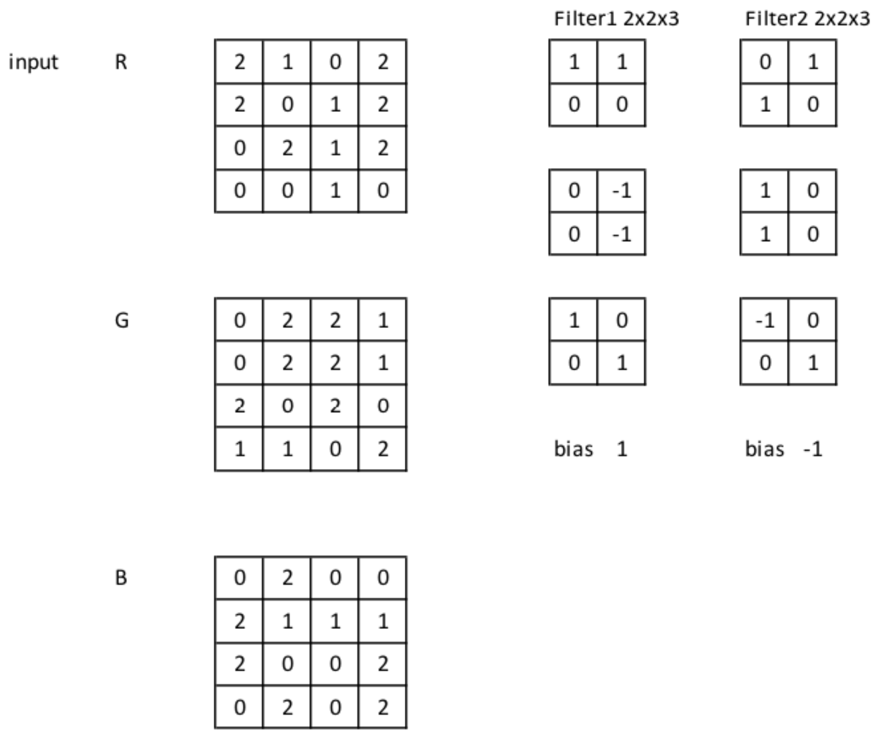
— How many activation maps will we obtain ?

— With $S = 1$ and $P = 0$, what will be the dimension of the output volume ?

— With $S = 2$ and $P = 0$, what will be the dimension of the output volume ?

— Give a filter size, padding value and stride value that will preserve the spatial dimension of the input.

— Use the excel file `cnn-ex1-stud.xlsx` provided to compute the values of the output with $S = 1$ and $P = 0$.



input

R $\begin{bmatrix} 2 & 1 & 0 & 2 \\ 2 & 0 & 1 & 2 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

G $\begin{bmatrix} 0 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \\ 2 & 0 & 2 & 0 \\ 1 & 1 & 0 & 2 \end{bmatrix}$

B $\begin{bmatrix} 0 & 2 & 0 & 0 \\ 2 & 1 & 1 & 1 \\ 2 & 0 & 0 & 2 \\ 0 & 2 & 0 & 2 \end{bmatrix}$

Filter1 2x2x3

$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & -1 \\ 0 & -1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

bias 1

Filter2 2x2x3

$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

bias -1

FIGURE 2 – 2D convolution

Exercise 2 Shallow networks with Keras on CIFAR10

Modify the example given in the class on the MNIST dataset to train and evaluate a shallow networks on the CIFAR10 dataset. The CIFAR10 dataset is composed of 60'000 images equally spread into 10 classes, i.e. with 6'000 images per class. The Figure 3 illustrates some images for each classes. Each images are 32×32 pixels with RGB channels. A single image can then be stored in a numpy array of shape $32 \times 32 \times 3$. Use the loader provided in TensorFlow that will split the set into 50'000 images for training and 10'000 images for testing.

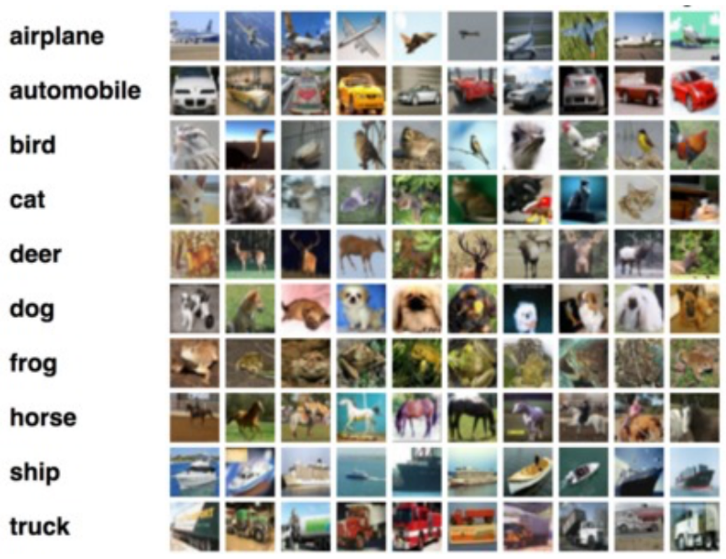


FIGURE 3 – Sample images of CIFAR10 dataset

One-layer network and weight visualisation

Train a simple 1-layer dense network using the following parameters : 5 epochs, batch size = 128, L2 regularizer (value of 0.05), `softmax` activation, `categorical-crossentropy` loss and `adam` optimizer. Report on the performance on the test set.

In a similar way as seen in class for the NIST database, plot the weights of the network. Pay attention that here, the normalization of the weight values to 0-256 values should be done on a per-channel basis (RGB). The L2 regularizer is actually used to impeach a too strong saturation of the weights, allowing a smoother rendering of the plots, as illustrated in Figure 4.

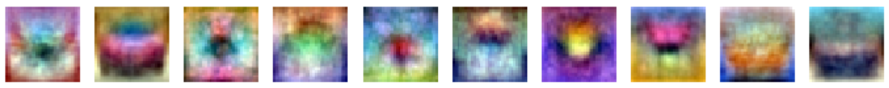


FIGURE 4 – Weight visualization - 1-layer network - CIFAR10 dataset

Two-layers network

Start with the following configuration : 128 neurons on the hidden layer, 30 training epochs, a batch size of 128 images, a ~~eclu~~ `relu` activation on the hidden layer, a `softmax` activation at the output layer, `categorical-crossentropy` loss and `adam` optimizer. You don't need to use a L2 regularizer in this model. This configuration should give you an accuracy around 45% on the test set.

Make some tuning on the hyperparameters in order to improve the performances, e.g. by increasing the number of neurons or by adding fully connected layers. Report your best performance.

Exercise 3 CNN with Keras on CIFAR10

Modify your MLP version from the previous exercise towards Convolutional Neural Networks.

Simple CNN

Start with the following configuration :

— Keep your input shape equals to (32x32x3), i.e. you don't need to re-shape your input into a flattened vector. A given training sample X_i will hold the raw pixel values of the ith image of width 32, height 32, and with three color channels R,G,B.

— Insert a first convolutional layer `Conv2D()` with 32 filters and stride $S = 1$. Let Keras compute the padding for you using the option `padding='same'`. This layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This will result in an output volume (32x32x32) as we decided to use 32 filters.

— Insert a ReLU layer that will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged (32x32x32).

— Insert a max pooling layer `MaxPooling2D()` of size 2. This layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in a volume (16x16x32).

— CNN usually ends with a fully-connected (dense) layer that will compute the class scores, resulting in a volume of size (1x1x10), where each of the 10 numbers correspond to a class score. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume. For this reason, in Keras, you first need to "flatten" the output volume of the previous layer, in our case from (16,16,32) towards (1,8192). To do that use the Keras layer `Flatten()`. You may finally plug the dense layer `Dense()` with a softmax activation.

Already with this simple CNN, you should observe a gain in your performance of about 10% against your best MLP version of the previous exercise.

Deeper CNN

From the previous CNN network, investigate with deeper architectures adding couples of `Conv2D()` and `MaxPooling2D()` layers. A rule of thumb is to try to extract higher level information going deeper in the network, therefore increasing the number of filters and the size of the kernel could be a good idea.

Report on your experiments and describe your best configuration through experimenting with 3-5 different architectures. Use a table similar to the example given below and provides the hyper-parameters used for your configurations.

CNN	Architecture	Acc. train %	Acc. test %
1	Layer 1 : CONV D=32, w=h=3, S=1, P='same' ; Layer 2 : MAXPOOL ... ; Layer 3 : DENSE ...	68.2%	62.4%
2	...	...	...
3	...	...	...
4	...	...	...
5	...	...	...

TABLE 1 – Performances on CIFAR10 with different CNN configurations. In the Architecture column, D is the number of filters, w h are the width and height in pixels, S is the stride value, P is the padding value, etc.

Exercise 4 Optional : Review Questions

a) What is the difference between a Dense and a Conv2D layer in Keras ?

b) In CNN, explain what it means to "preserve the spatial structure" ?

c) In CNN, explain the different operations taking place when performing a convolution of a filter on an input, e.g. an image. How is the filter defined in terms of hyper-parameters ?

d) Define what is meant by receptive field and activation map.

e) Explain the effect of a given stride or zero-padding value.

f) For a given configuration of a convolution layer, compute the number of parameters (weights, bias).

g) Define what is a max pooling layer and what is its role in a CNN architecture.