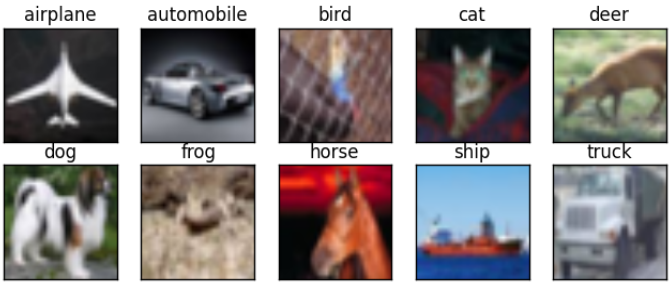
Neural Networks for Images - Exercise #1

submission date: 11/4/2021

Programming Task: Classification using convolutional neural networks (CNNs)



In this exercise you will explore and evaluate the importance of various components of a classification CNN. This will be done by adding, modifying and removing different components. In each of these experiments you will need to implement the change, train the network, rationalize the results and document your conclusions as to what happened and why in a pdf report that you will submit (see Submission Guidelines below).

The baseline network for this exercise is this vanilla network trained over the CIFAR10 dataset. This dataset consists of 10 object classes with 6000 example images from each class. The images are fairly small, 32-by-32 RGB pixels.

Formally,

1. **Architecture Fine-Tuning.** Plot the train and test losses and explore how they change as you change the total number of filters (neurons) in the network to the levels of overfitting and underfitting. Find the configuration resulting with the lowest test error. Report the final results in your report, including your other experimental results.
2. **Linear Model.** What happens to the performance of the network once you remove all the non-linear components it contains. List these components and explain the results obtained. Increase the dimension of the last FC layers. Did this improve the results? Explain.
3. **Locality of the Receptive Field.** As we discussed in class, in order for the network to capture local dependencies in the image (corresponding to the shape of particular objects) it uses conv. layers with neurons with a small and localized receptive field. We would like to evaluate the importance of this locality. Simply using larger filters will not allow us to test this aspect - **why?**
   Instead we will randomly reshuffle the spatial locations of pixels in the image. This shuffling should be the same for all the images, and therefore we will sample a single random permutation over a grid of 32x32 pixels **once** (at the beginning of the code) and use it to permute the pixels of each batch before using it (basically, anywhere a batch is extracted from the dataset in the code!).
   In this test, we will keep the same number of weights in the network, yet we will de-localize the receptive field. Explain what happens to the network performance, and

why you think this is the case (trying to capture local versus global structure/correlations in images makes more sense).
Note that it might be easier to implement this pixel reshuffling in1D, over vectors of dimension 1024. Recall that the image batches in this code are of [4,3,32,32] which corresponds to four images with three channels and 32x32 pixels.

4. **No Spatial Structure.** It will be interesting to see how far the network can get by operating over the input pixels without their spatial structure. Modify the above test to break the image structure completely, i.e., it should receive the pixel values but not over a consistent order by sampling the shuffling permutation **every time it is used**. Describe the results obtained and whether there is any class that did surprisingly well - why is that?

We expect you to report and elaborate on every practical task in the report, using your own words and your own analysis of what you've done. Include everything that you think is crucial for us to understand your way of thinking.

Theoretical Questions:

1. Formally show that the condition L[x(t+s)](y) = L[x(t)](y+s) over a linear operator L receiving a 1D signal x(t) with offset s (the outer parentheses refer to the output signal), corresponds to a convolution. Hint: decompose the signal x(t) into a weighted sum of translated delta functions, and then use the linearity of L. What input signal will output the underlining filter of the convolution?
2. When reshaping a 2D activation map (resulting from a convolutional layer) and feeding it into an FC layer, how important is the order?
3. The convolution operator is LTI (linear translation invariant). What about the following operators:
    a. The ReLU activation function
    b. The strided pooling layer (i.e., the one that always picks the top-right pixel in each block).

    Explain your answers.

Submission Guidelines:

The submission is in **pairs.** Please submit a single zip file named "ex1_ID1_ID2.zip". This file should contain your code, along with an "ex1.pdf" file which should contain your answers to the theoretical part as well as the figures and analysis for the practical part. In addition, please include in this compressed archive a README with your names and cse usernames.
Please write readable code, with documentation where needed, as the code will also be checked manually.