Chapter 3

Convolutional Neural Network

Part 1

Professor:

José Oramas Mogrovejo

Teacher Assistants:

Hamed Behzadi Khormouji, Arian Sabaghi

Artificial Neural Networks

University of Antwerp

March 9, 2023

1 Introduction

This section introduces a Convolutional Neural Network in practice. Here, we cover the practical implementation of some aspects discussed in the theory lectures, namely **2-D convolution operations with different configurations, and pooling operations**.

1.1 Learning Outcomes

a) You will obtain an intuition of 2D convolutional operations in practice.

b) You will obtain an intuition on the effect of different kernel sizes, i.e., $k = 1$ and $k>1$.

c) You will get familiar with architecture design.

d) You will get familiar with architecture modification tips in order to obtain a trained model with higher accuracy.

e) You will obtain an intuition of transformation effects on training phase.

2 Environment

In this session, you are free to choose the environment that is more convenient for you. This could be either Google-Colab, IDLab's GPULab, computers from the lab, or your local machine.

3 Dataset and Code

You will find code for this exercise in the BlackBoard platform located at the address *UA_2500WETANN: Artificial Neural Material/Practical Session/Code/Chapter 3:Convolutional Neural Networks-Part1*. The zip file contains the 15 scenes dataset as well. This dataset includes 15 classes of different scenes.

4 Convolution Operation

In this section, you will work with a very shallow feature extraction model composed by just one convolutional layer that receives a 3-channel input and outputs a three channels tensor. Each channel can be separately visualized to illustrate what is the output of each filter in this convolutional layer. The script file *conv_operation.py* implements such a model.

4.1 Problem 1

a) Run this script file several times to see visualizations in each run. Now the first question is that what is your interpretation from generated visualizations? and as a second question, discuss why does the visualization of convolution operation output change in each run?

5 Convolutional Neural Networks - Part 1

In this section, you will train a Convolutional Neural Network (CNN). This section is composed by several sub-problems. For each sub-problem, in addition to running the produced code, you should also **save the obtained results for further comparisons and discussion**.

5.1 Problem 2 - CNN Schematic

The script file *Model.py* implements the architecture of a CNN model via the class *Network*. The *Constructor __init__* of the class defines the layers of the model and the *Forward* function implements the computational graph of the model. Before running the code, in this problem, you are asked to draw on a paper the schematic visualization of the model implemented in the code. Your schematic should specify the arrangement of the layers, the order of their connection, along with the configuration of each layer. For example, the schematic should be similar to the one shown in Figure 1.

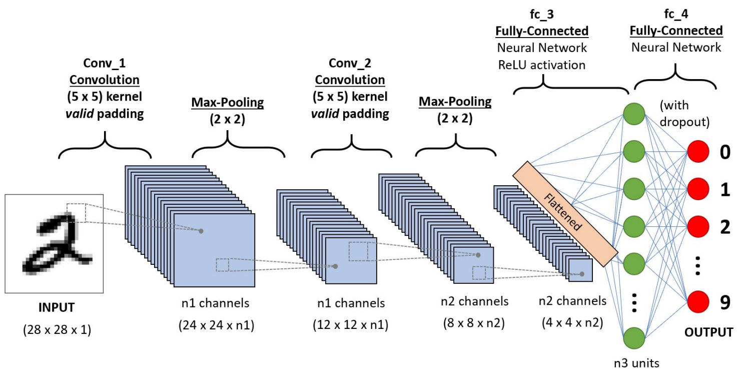

Figure 1: Example of Schematic Visualization of an arbitary model

5.2 Problem 3 - CNN Implementation

As a reminder, save the obtained results after each experiment. The goal is to get familiar with the reason(s) for the trends we observe in the results when we modify different parts of the model or the training procedure.

a) First, run the code. You will face an error. What the does error show? What would be the reason for such error? Provide a code snippet to solve it (one possible place for implementing this code snippet is inside the *feature_extraction* function in the class *Network* from the file *Model.py*).

b) After fixing the error, run the code each time with different learning rate values *1e-3, 1e-1*. How do you justify the train loss and accuracy trends observed in both experiments?

c) In the next step, Add *transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])* to the data transformation processes and test the program and record the results. How do you justify the results you have achieved so far?

d) In the next experiment, add *transforms.RandomHorizontalFlip()* to the transformation procedure. Is there any change in the accuracy and loss of train and test data? How do you justify the obtained results?

e) Set the kernel size of the convolutional layers to 1? What does kernel size = 1 mean? Run the program and record the accuracy and loss performance.

f) Set the kernel size to 9 or change the stride to a value larger than 1. For each configuration, take an experiment from the code and record the accuracy and loss performance. Have you encountered any errors? What has caused such an error? Based on the results of this experiment and the previous one, what might be the effect of kernel size = 1?

g) If you were to use a convolutional layer instead of a fully-connected layer, how would you change the implementation? What would be the type of convolution operation, i.e., 1-D or 2-D? And what would be the configuration of the convolution layer that makes it equivalent to the currently implemented fully-connected layer?

h) Finally, according to the modification applied to the model and the results obtained, what is your suggestion for improving the performance of the model?