Preston Robertson

IE 8990
Spring 2022
Homework #2
Due Date: 3/8/2022 5PM CST
Submission: Please put your answer and code in a PDF file and upload on Canvas

---

Q1. Consider a 1-dimensional time-series with values 2, 1, 3, 4, 7. Perform a convolution with a 1-dimensional filter (1, 0, 1) and zero padding with stride of 1.

- The answer is (3,4,7)
    - The pattern is noticeable, that the farther numbers on the right are higher so as the stride moves along this 1D set of values the stride will pick up the right most value. Which is plausible because the final filter number is 1.

Q2. Consider an activation volume of size 13 × 13 × 64 and a filter of size 3 × 3 × 64. Discuss whether it is possible to perform convolutions with strides 2, 3, 4, and justify your answer in each case.

- Output Size = ((Input Size + (2 * Padding Size) – Filter Size)/stride) + 1
    - Using this equation, we can see the resulting matrix to verify if it is possible.
- In the next table I used the above matrix to see if the results are a whole number, if it is yes then it is possible to perform convulsions
    - In some instances, it is important to change the padding size to allow for the convolutional layer to be used.

Table 1: Answer to Question 1

Stride	With No Padding	Verification	With Padding of 2	Verification
2	6	Yes	N/A	N/A
3	4.333333	No	5	Yes
4	3.5	No	4	Yes
5	3	Yes	N/A	N/A

Q3. Based on the input image in Table 1:

Table 1: Input

6	3	4	4	5	0	3
4	7	4	0	4	0	4
7	0	3	4	4	5	2
3	7	0	3	5	0	7
5	8	2	5	5	4	2
8	0	0	6	6	0	0
6	4	3	0	0	4	5

Table 2: Horizontal edge detection filter

1	1	1
0	0	0
-1	-1	-1

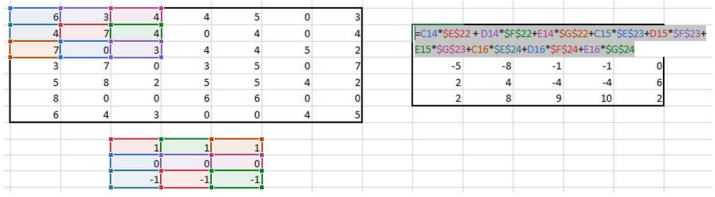
- Computer the convolution of the input volume with the horizontal edge detection filter in Table 2. Use stride of 1 without padding
    - Output Size = ((Input Size + (2 * Padding Size) – Filter Size)/stride) + 1
        - Using this equation, we can see that a 5x5 matrix is the result.
- I used Excel to solve this problem, the equation is here:
    - C14*E22 +D14*F22+E14*G22+C15*E23+D15*F23+E15*G23+C16*E24+ D16*F$24+E16*$G$24
    - 

Table 3: Question 3 Part 1

3	4	2	-4	-3
5	1	0	-4	-4
-5	-8	-1	-1	0
2	4	-4	-4	6
2	8	9	10	2

- Perform a 4×4 pooling at stride 1 on the result from the previous question
    - I used the max pooling technique and got the results below.
    - Output Size = ((Input Size + (2 * Padding Size) – Filter Size)/stride) + 1
        - Using this equation, we can see that a 2x2 matrix is the result.

Table 4: Question 3 Part 2

5	6
10	10

Q4. Randomly select 20% of the CIFAR-10 data as test data. Train the network on subsets of varying size (10%, 50%, 80%) from the remaining CIFAR-10 data. Perform the following tasks:

- Train DenseNet121 architecture. Plot the top-1 error with data size.
- Design a CNN model based on your own choice. Plot the top-1 error with data size.
    - At the bottom of the document.

Q5. Review: Select one article from below and summarize it (1 page, single-space, font size 12pt)

- Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., Kolter, Z. (2019). Differentiable convex optimization layers. arXiv preprint arXiv:1910.12430.

This article is about how Agrawal et al. applied the principles of Domain-Specific Languages (DSLs) to differentiable convex optimization problems. The DSL allowed for non-experts in optimization to have access to convex optimization. One way this paper accomplishes this is through making a standard language for differentiable convex optimization, they call the grammar "disciplined parametrized programming". One intention of this standardized grammar is to allow for use of differentiable convex optimization in standardized programs such as PyTorch and TensorFlow 2.0. This lowers the barrier entry significantly. To clarify, an optimization problem is like problems from linear programming. It is minimizing or maximizing the problem with some constraint. In practice, this way of solving problems can mathematically solve problems far faster than other methods. This is due to multi-threading on CPU and GPU. The convex optimization problem is one of the solutions to our new activation functions tunable parameter. Back to the summary, another novel point from the papers is the use of abstract forms instead of conic forms. This has the advantage of being able to quickly experiment through new problems that did not have conic forms. This method still allows for conic form problems to be solved. Using abstract forms means the user does not need to pre-process data in conic forms allowing for more ease of use for non-experts. This abstract form allows for dynamic learning. This dynamic learning comes from the ability to minimize across the entire set of options. Whereas stochastic gradient descent in neural networks is short sighted based on mini-batch size. The convex optimization problem can understand all options and pick the most optimal choice given the problem domain. This is one of the reasons why optimization problems have such popularity as a modeling tool.

One of the main difficulties when using optimization for a problem is tracking change. In stochastic gradient descent, how you optimize your problem is by showing how the changes in each weight effects the loss. This change is derivation to show the impact/importance of a specific weight. Optimization lacked the ability to show the change in the solution based on certain parameters since it changes for each problem. This paper combats this by showing a standardized way to take the derivative of any differentiable convex optimization problem to map the importance of specific inputs. This allows for the optimization problem to be solved much more quickly than before, making it a viable solution. Since the entirety of neural network training is essentially an optimization problem, this method is now a potential solution to training speed and accuracy. The use of convex optimization will allow for quicker grid search commands for non-neural network based machine learning techniques as well. How are the values retrieved from the solver? The solver finds the optimal input that would satisfy the conic form of the problem then reshaping and scaling to the original problem.

Overall, this paper sought to explore the use of non-conic optimization problems in machine learning. This would significantly reduce training time if ever implemented; however,

the optimization problems were too complex for non-experts, so the paper made its own grammar that would work with TensorFlow and PyTorch. This gives the ability to use these optimization problems to non-experts. Finally, this paper found a way through the use of derivates that it is possible to solve non-conic optimization problems. This will have large impact when grid-searching hyper parameters for several hyperparameters.