Unnecessarily Complicated Addition and Subtraction of Two Digits

A typical task of an AI engineer is to create prototypes to evaluate new ideas. Therefore, we have designed a task where you have to implement a small project independently. In this task, your solution will be evaluated holistically. It doesn't just matter how well or poorly your approach solves the problem. Rather, it also depends on whether the chosen solution approach (e.g., network architecture) is appropriate, whether typical best practices are followed, what the quality of the source code is, whether you have **comprehensively** evaluated your approach and can present it appropriately. To help us evaluate this, please prepare in addition to your implementation a short report (written document, slides or Jupyter notebook) that shows your thought process and includes an analysis of the data, your approach and results.

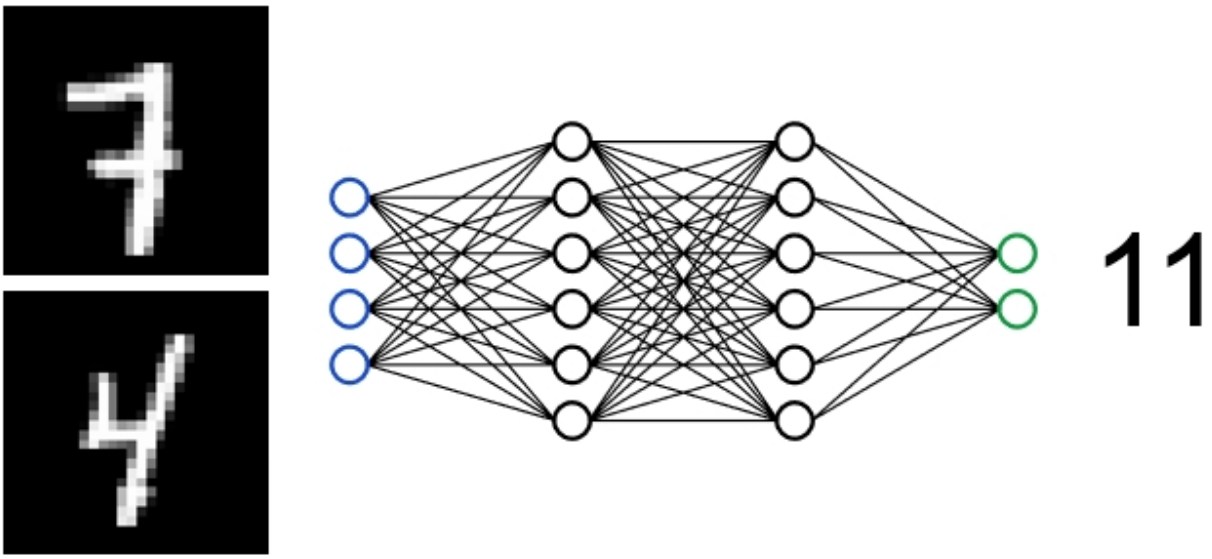
Although it is a toy problem, you should treat it as if it was given to you on your job by your boss and the code is intended to be used in the future for other projects. Don't implement the most creative/interesting approach you can think of, but implement the most appropriate/best approach from your point of view, just like in a real scenario. We assume that the solution you send us reflects the way you would normally code, structure projects and follow best practices. Please be careful not to allocate too little time to individual aspects such as the evaluation or the report.

The task was designed in such a way that no GPU is absolutely necessary for training the required neural networks. Typically, applicants spend about 8 hours to complete this task, including the report. Please email your solution (code & report, but no training data or trained networks) to the recruiter/interviewer as a zip file. Be prepared to present/discuss your approach and results during a review session. We'll ask you to share your screen during this one, so we can look at your code, results, and report together. So before the review session begins, make sure MS Teams is installed on your computer and your IDE is started and working.

Please read the full document (including the requirements at the end) before you start working on this task.

Subtask 1: SuMNIST

Design, implement and evaluate an approach based on a *single* neural network that outputs the sum of two handwritten digits given as individual images (see image).



The network should be trained in an end2end fashion. This means that your approach should consist of a single neural network that processes both images simultaneously and outputs the sum, i.e. `sum = net(image_a, image_b)`. Accordingly, the network may only use image pairs as input and sums as output during training. It is completely fine, if e.g. the network output needs a simple 1:1 mapping to be interpretable. How exactly both images are given to the network and what the network architecture & output looks like is part of the task. Any network architecture and any way handling both images is allowed.

Approaches that are **NOT** allowed:

- Train a part of your network or a completely separate network on MNIST only (i.e. single image -> [0-9])
- Perform the addition outside your network in Python, e.g. after classifying both digits individually, i.e. `result = net(image_a) + net(image_b)`
- Use pretrained weights

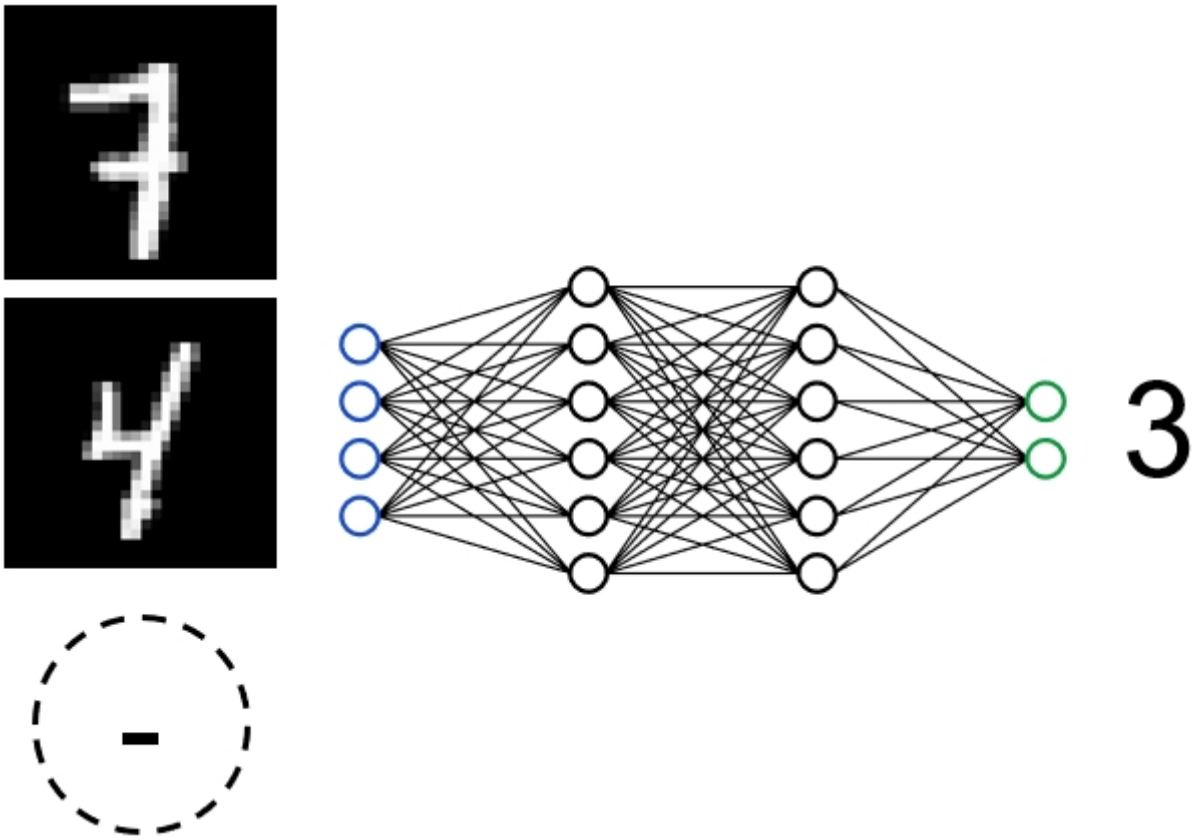
We expect that your approach covers at least the following points:

- Based on the MNIST Dataset, create an appropriate custom datasets for this subtask. Write code that allows using these datasets for training and evaluation.
- Extend and adapt the Pytorch MNIST Example code to train one or more custom network architectures that solve the given problem. This sample code serves only as a starting point for your implementation. Don't feel bound to decisions, approaches and parameters of the sample code, but change everything you want and follow best practices you know. We prefer clean and elegant approaches that solve the problem to unnecessarily complex ones.
- Design and implement appropriate ways to evaluate your approach. Carefully choose descriptive and intuitive metrics and visualizations that are particularly well suited to this subtask. Your evaluation should reveal the strengths and weaknesses of your trained network.

Subtask 2: DiffSuMNist

Extend your code to train a *single* network that is capable of both adding and subtracting two digits. In addition to the images of both digits, a third input (**NOT an image**) tells the network to either add or subtract both digits. In case of subtraction, the digit in the second image should be subtracted from the digit in the first image. Please let the solution of this subtask be based on the source code and the results of the first task. Adapt the dataset generation, network architecture and the evaluation process. Otherwise, the same requirements apply.

Warning: Do **NOT** learn two separate networks for this subtask! Also, the network should have a shared head/output for adding and subtracting images.



Requirements

- Use Pytorch for the implementation of your solution.
- Use an appropriate project structure for your solution. We should be able to reproduce your results on our machines.
- Don't put all your code into a single jupyter notebook or python file, but use an appropriate project file structure. For the report, you can use a jupyter notebook.
- Comment your code accordingly, so that we are able to understand the steps you are taking and the reasoning behind them.