Jacob introduced us to Pytorch and data parallelism within the last few lectures. To get familiar with both writing Pytorch models and data parallelism, in this assignment we are going to write a Pytorch model from scratch to be trained on the MNIST dataset. We will then wrap this model in Pytorch's data parallel abstractions, and train it on Isaac four times, increasing the number of ranks used. We will then visualize the results of this model training and discuss how the model scales with the number of ranks. Please create a GitHub repository for your work and share a link to it on Canvas. This will help you go between Isaac and your machine, and it will help us guide you if you run into any issues. This GitHub repository with the work from the following two problems will be your official submission.

1. Let's write a data parallel convolutional net to train on the MNIST dataset. Our goal is to see how data parallel training affects two things: our model's accuracy, and its runtime. Due to some limitations on Isaac, we are going to train this model using a single node, but we will increase the number of processes per node to see how things scale. Use the number of processes $p \in \{1, 2, 4, 8\}$ and train the model for 30 epochs each time. We want to keep track of two things:

   a. Our training accuracy at every epoch
   b. The total time it takes the model to train

   > **Note:** *I recommend saving your training accuracy at every epoch as a numpy array. You should have a numpy array for each run where p number of processes were used. In the next step, we will plot these accuracy curves for each of the runs to compare how our model accuracy changes as we vary the number of processors (and the effective batch size as a result).*

2. After training the model for 30 epochs for each of the p number of ranks in problem one, let's compare both our models' accuracy and their total runtime for each run. Create two plots:

   a. A line plot showing the epoch accuracy for each of the runs
   b. A second line plot that shows the total runtime by the number of processes used

   What is the effect of training time as we increase the number of processes available? How does this line up with your expectations of the scalability laws that we discussed? Which scalability law is most applicable in this case? How is our model accuracy affected by the increase in the number of processes running? Present your figures and answers to these questions in either a short document or a Jupyter notebook in your repository.