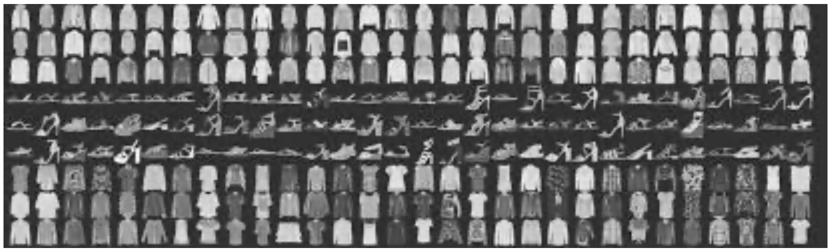
Hyper-parameter tuning with Pytorch

The objective of this lab is to learn how to tune the hyper-parameters of a simple network to classify clothes items in the Fashion MNIST benchmark, a revisitation of the famous MNIST hand-written recognition benchmark.



The dataset is available at `torchvision.datasets.FashionMNIST`

Preliminaries

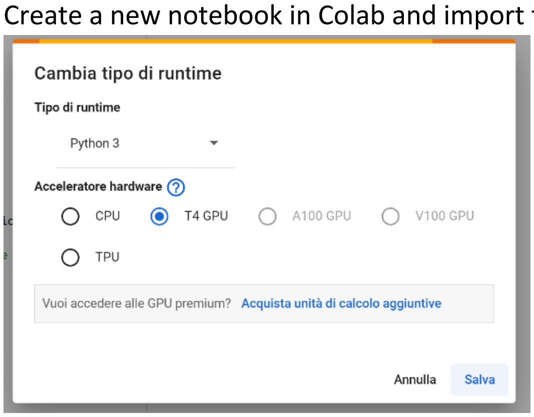
Before starting, review or have ready the following material:

- Review introduction to Pytorch and MNIST notebook
- Review lesson on gradient descent 03. Parameter learning - Gradient descent .pdf
- Review the concepts of under-fitting and over-fitting (04. Model and cost function.pdf – slides 52-62)
- Information about layers available in Pytorch is found in the documentation https://pytorch.org/docs/stable/nn.html
- (Bonus) view additional material (slides and video) on optimizers and hyper-parameter optimization

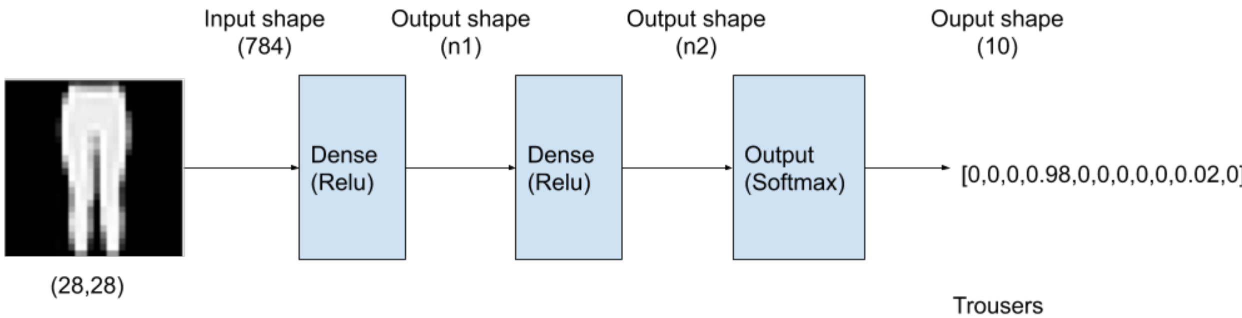
Lab activity

Follow these steps, complete the notebook and submit your answers:

1. Create a new notebook in Colab and import the libraries. Remember to set the runtime to GPU!



2. (Optional) Fashion MNIST only includes a training and test set. In order to monitor overfitting, it is better to also create a validation set. A possible way to construct the validation set is to reserve 10% of the training set as validation set. Hint: look up torch.utils.data.random_split
3. First, we can solve the problem using a classical Multilayer Perception (MLP) with 2 hidden layers and 1 output layer. The network should look something like this:



   Choose different values of n1 and n2 (number of units in the first and second layer)
   Hint: Consider that linear layers are designed to work on 1-dimensional input.
4. Manually select three learning rates and train the model for 30 epochs, or until the accuracy on the validation set stops improving (early stopping). Plot the loss and accuracy on the training and validation set.
   Hint: Pay attention to the batch size -
   Hint: verify if 30 epochs are sufficient for the training loss to reach a plateau
   Hint: What happens if you increase/decrease the selected learning rate by one or two order of magnitude? (e.g. from 0.1 to 1 and 0.01)
5. Second, let's build a convolutional neural network (CNN) by adding two 3x3 convolutional layers with n1 and n2 channels, respectively (each followed by ReLU activation and 2x2 max pooling).
   Study question: If n1 = 64 and n2 = 128, what is the number of hyperparameters for the MLP and the CNN architectures?
6. Now let's start hyperparameter optimization! Examples of hyper-parameters that you can optimize include batch size, learning rate, optimizer, regularization, etc. Look at the learning curves on the training and validation set, watch out for signs of over- and underfitting, and decide your next step.

Optional questions

7. Select the configuration with **the best validation loss**, **run at least 3 trainings** with the same hyper-parameter configurations.
   Study question: Why and how much does the accuracy change between different training runs?

Submit your best score(s)

- Submit your best score (scores if you were able to repeat the run three times) here https://forms.gle/ktNrCS85sU4ttunH9
- Best scores will be discussed in the next lab!