UNIVERSITY OF MINNESOTA
Driven to Discover℠

MNIST Classification using PyTorch

EE 5561
Fall 2023

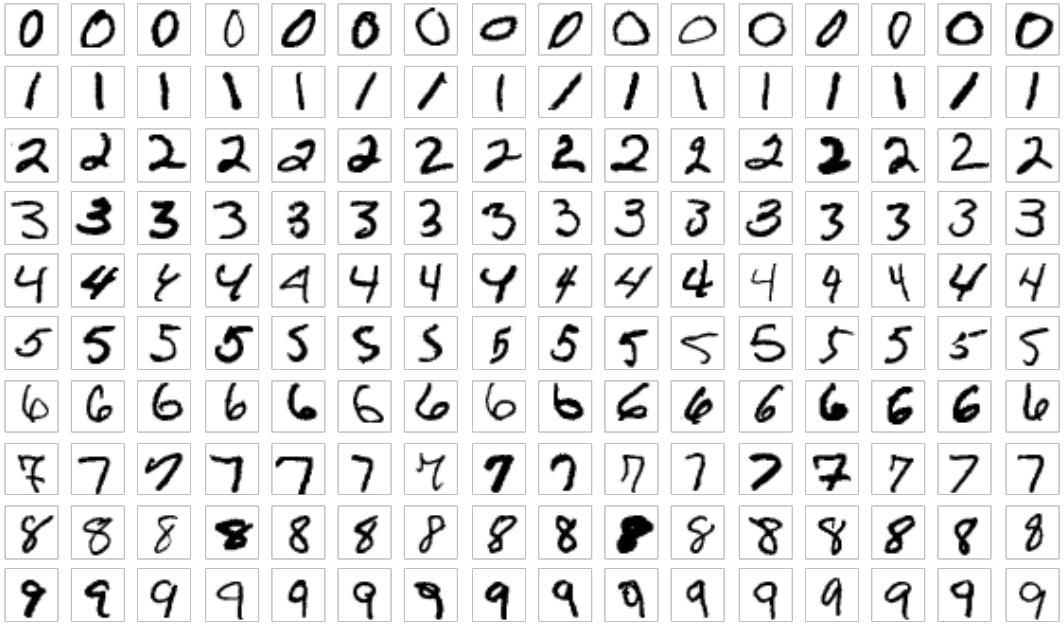
Introdution

- Open-source machine learning framework
- Think like Numpy but with strong GPU support

- Setup:
  - `conda install pytorch torchvision -c pytorch`
  - `pip3 install torch torchvision`

MNIST Database (Modified National Institute of Standards and Technology Database)

- MNIST dataset
  - Handwritten digits
  - 60k images for training
  - 10k images for testing
  - with labels (0-9)
  - 28 × 28 pixel grayscale images



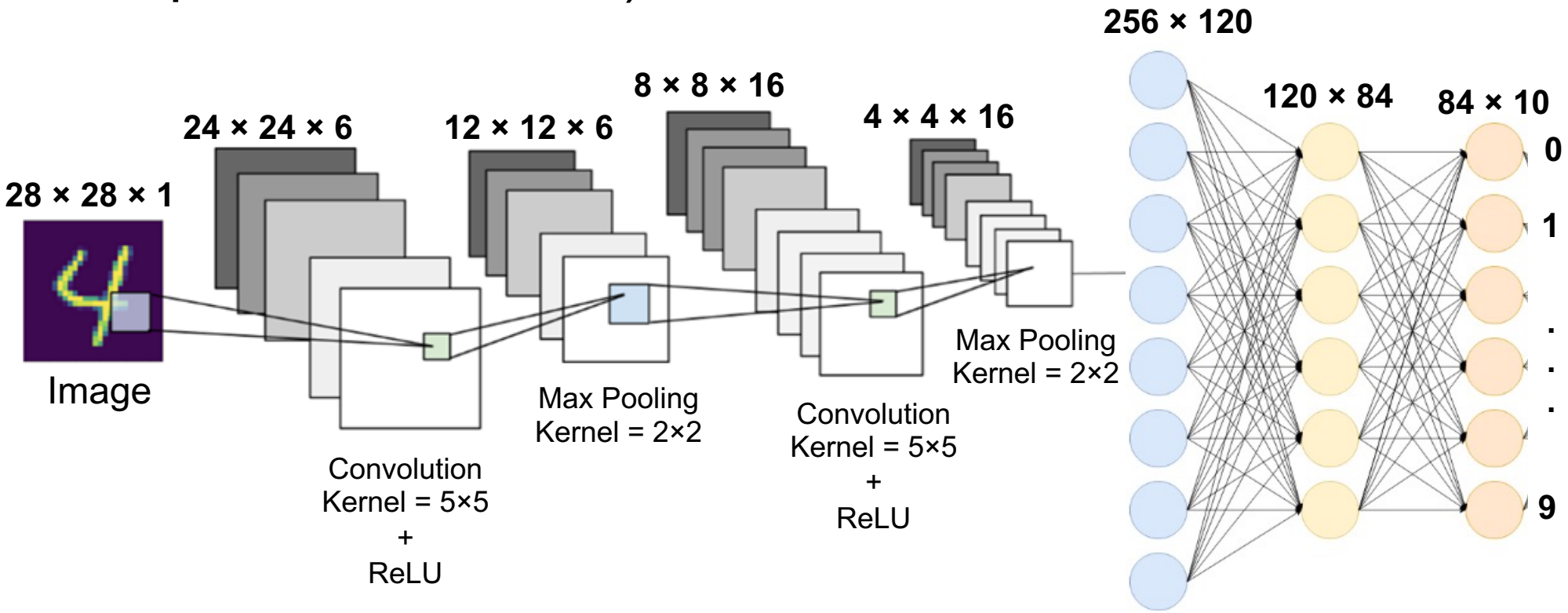
MNIST Classification

- **Task:** Classify a given handwritten image into one of 10 classes
- **Classes:** Representing the integer values from 0 to 9

- **Model:** LeNet

LeNet

- A CNN structure
- A simple well-known network used in MNIST classification (once upon a time, 1989!)



Loss Criteria

- https://pytorch.org/docs/stable/nn.html

TORCH.NN

These are the basic building blocks for graphs:

torch.nn

- Containers
- Convolution Layers
- Pooling layers
- Padding Layers
- Non-linear Activations (weighted sum, nonlinearity)
- Non-linear Activations (other)
- Normalization Layers
- Recurrent Layers
- Transformer Layers
- Linear Layers
- Dropout Layers
- Sparse Layers
- Distance Functions
- Loss Functions
- Vision Layers
- Shuffle Layers
- DataParallel Layers (multi-GPU, distributed)
- Utilities
- Quantized Functions
- Lazy Modules Initialization

Loss Functions

`nn.L1Loss`	Creates a criterion that measures the mean absolute error (MAE) between each element in the input x and target y.
`nn.MSELoss`	Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y.
`nn.CrossEntropyLoss`	This criterion computes the cross entropy loss between input and target.
`nn.CTCLoss`	The Connectionist Temporal Classification loss.
`nn.NLLLoss`	The negative log likelihood loss.
`nn.PoissonNLLLoss`	Negative log likelihood loss with Poisson distribution of target.
`nn.GaussianNLLLoss`	Gaussian negative log likelihood loss.
`nn.KLDivLoss`	The Kullback-Leibler divergence loss.
`nn.BCELoss`	Creates a criterion that measures the Binary Cross Entropy between the target and the input probabilities:
`nn.BCEWithLogitsLoss`	This loss combines a *Sigmoid* layer and the *BCELoss* in one

Optimizers

- https://pytorch.org/docs/stable/optim.html

`ASGD`	Implements Averaged Stochastic Gradient Descent.
`LBFGS`	Implements L-BFGS algorithm, heavily inspired by minFunc.
`NAdam`	Implements NAdam algorithm.
`RAdam`	Implements RAdam algorithm.
`RMSprop`	Implements RMSprop algorithm.
`Rprop`	Implements the resilient backpropagation algorithm.
`SGD`	Implements stochastic gradient descent (optionally with momentum).

Reminder

- Group project proposal deadline: Nov. 20 (**will not be extended**)
- You should be
  - Forming your groups (if you haven't done so already)
  - Coming up with project ideas
- If you do not have a group yet, email me & Merve, also include
  - What type of task you are interested in (e.g. denoising or segmentation, etc)
  - What type of application you are interested in (e.g. medical or astronomical, …)
  - So far only heard from one person