Deep Learning Assignment 6 in FS 2022

Convolutional Networks

Microsoft Forms Document:
https://forms.office.com/r/89d0K2m5yr

Manuel Günther

Distributed: Friday, April 1, 2022
Discussed: Friday, April 8, 2022

The goal of this exercise is to show the difference between fully-connected networks and convolutional networks. We will make use of the MNIST dataset of handwritten digits, which are provided through the `torchvision.datasets.MNIST` interface. Also, we will learn how to use input transforms and batch processing in PyTorch.

1 Dataset

In PyTorch, a dataset stores a list of input and target tensors $(\mathcal{X}^{[n]}, \mathcal{T}^{[n]})$. In case of MNIST, the inputs are $\mathcal{X}^{[n]} \in \mathbb{R}^{28\times28}$ and $\mathcal{T}^{[n]} \in \{0, \dots, 9\}$. More precisely, the data in the dataset is provided in form of `PIL.Image.Image`, which represents an image class with some more functionality, and pixel values in range $[0, 255]$. In order to convert these images into `torch.Tensor`'s in range $[0, 1]$, we can use the `torchvision.transforms.ToTensor` transform. Furthermore, in PyTorch batches are created from datasets using the `torch.utils.data.DataLoader` class.

Task 1: Dataset Implement a function that takes a given transform object and returns two datasets, one for training and one for testing.[1]

Test 1: Data Types Call this dataset generation function with `transform=None` to avoid the data type conversion. Assure that the returned image is of type `PIL.Image.Image`, and the target is of type `int`.

Task 2: Data Loaders Call the dataset generation function with the `ToTensor` transform. Select a training batch size of $B = 64$, and a test batch size of your choice. Instantiate data loaders for the two datasets. Decide if data shuffling is required for either of the two.

Test 2: Batches Check that all batches of the training set are in the required batch size – except for the last batch. Also check that all input and target batches are of type `torch.Tensor`. Check that all inputs are in range $[0, 1]$ and that all target values are in $\{0, \dots, 9\}$.

[1] In our small example, we will directly use the test data for validation purposes.

1. flatten layer to convert the 28×28 pixel input into a $28 * 28$ element vector
2. fully-connected layer with D inputs and K outputs
3. activation function tanh
4. fully-connected layer with K inputs and K outputs
5. activation function tanh
6. fully-connected layer with K inputs and O outputs

(a) Fully-connected Network

1. 2D convolutional layer with Q_1 channels, kernel size 5×5, stride 1 and padding 2
2. 2D maximum pooling layer with kernel size 2×2 and stride 2
3. activation function tanh
4. 2D convolutional layer with Q_2 channels, kernel size 5×5, stride 1 and padding 2
5. 2D maximum pooling layer with kernel size 2×2 and stride 2
6. activation function tanh
7. flatten layer to convert the convolution output into a vector
8. fully-connected layer with the correct number of inputs and O outputs

(b) Convolutional Network

Table 1: Network configurations of the fully-connected and the convolutional network

2 Networks

As last week, we will rely on `torch.nn.Sequential` to create networks with particular lists of consecutive layers. Particularly, we will investigate two different versions of networks, one fully-connected network and one convolutional network, with the same number of learnable layers. The two network topologies are provided in Tab. 1.

Task 3: Fully-Connected Network Although we have seen that two layers can approximate any function, we use a three-layer fully-connected network in this assignment. Implement a function that takes parameters D, K and O to produce a three learnable layers, where the two hidden layers have the same number of K hidden neurons (for simplicity). Between these layers, use tanh as activation function. In order to convert the 28×28 pixel input into a $28 * 28$ element vector, we can make use of the `torch.nn.Flatten` layer.

Task 4: Convolutions Output (theoretical question) We need to define the size of the vector resulting from the flatten layer (7) in Tab. 1(b), in order to know the size of the input of the fully-connected layer (8). Provide a theoretical analysis of the output size of the flatten layer, given that the input to the network is of size 28×28.

Task 5: Convolutional Network For the convolutional network, we will use two convolutional layers including pooling and activation and variable numbers Q_1 and Q_2 of output channels for the convolutions, as given in Tab. 1(b). Implement a function that generates this network.

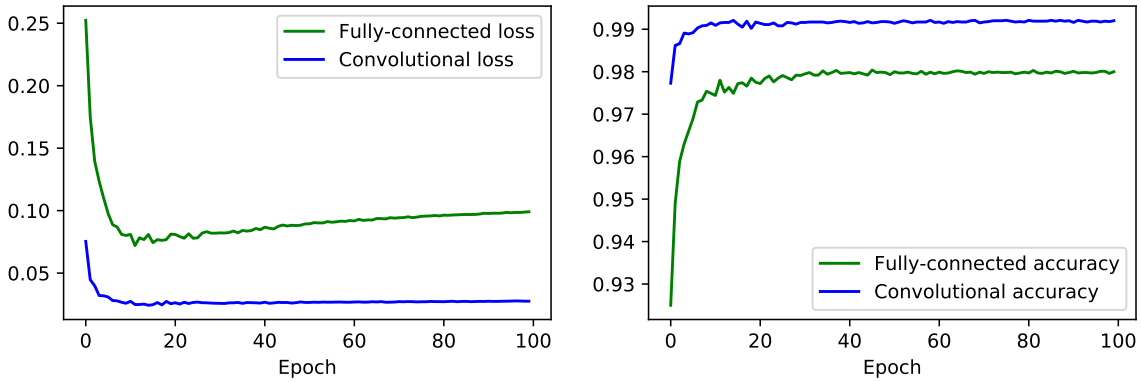

Figure 1: This figure shows some exemplary progression plots of loss values and accuracies for fully-connected and convolutional networks.

3 Network Training

For training and evaluating the network, we will rely on standard functionality in PyTorch. We will use the standard categorical cross-entropy loss together with a stochastic gradient descent optimizer. For training, we will use the batched implementation of the dataset, for which we perform one update step for each training batch. After having gone through the full training dataset, we will compute test set accuracy and loss. What do we need to consider when implementing this in a batched way, too?

Task 6: Training and Validation Loop Implement a training function that takes the network, a given number of epochs and a learning rate. The computed test set loss and accuracy values should be stored in two separate lists, and they should be returned after the training is complete.

Task 7: Fully-Connected Training Instantiate a fully-connected network with $K = 100$ and $O = 10$. Train this network for 10 epochs, with a learning rate of $\eta = 0.01$. Store the resulting lists of loss and accuracy values.

Task 8: Convolutional Training Instantiate a convolutional network with $Q_1 = 32$, $Q_2 = 64$ and $O = 10$. Train this network for 10 epochs, with a learning rate of $\eta = 0.01$. Store the resulting lists of loss and accuracy values.

Task 9: Plotting Plot the loss values achieved over the epochs for both networks into one plot. Plot the accuracy values achieved over the epochs for both networks into another plot. Exemplary results can be found in Fig. 1. Note that this plot includes results for 100 epochs.

Task 10: Learnable Parameters Estimate analytically how many learnable parameters are contained in each of the two networks. Implement a function to count the number of parameters that are actually contained in the networks.