DATA.ML.200 Pattern Recognition and Machine Learning

Exercise week 4: MLP and CNN in PyTorch

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. **pen&paper** Count the number of trainable parameters (2 points)

   a) First study the MLP model in Figure 1. The input is a 64×64 RGB image that represents one of the two different traffic signs.

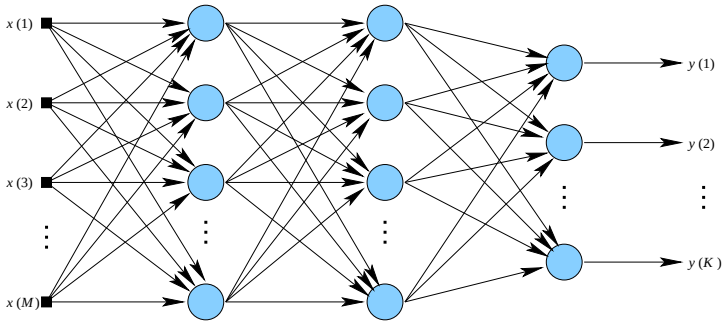

Figure 1: Vanilla neural network.

   The network structure is

   - The $3 \times 64 \times 64$-dimensional is flattened
   - in the 1st layer there are 100 nodes (marked in blue)
   - in the 2nd layer there are 100 nodes (marked in blue)
   - in the 3rd (output) layer there are 2 nodes (marked in blue; one for each class)

   Compute the number of trainable parameters (weights).

2. **python** *Load Traffic sign data for deep neural network processing.*

   Download the two class German Traffic Sign Recognition Benchmark (GTSRB) dataset (GTSRB_subset_2.zip) from the course Moodle page. Images are color images and there are about 400 of each class. Split your data into two parts - 80% for training and 20% for testing. Note that there are ready-made functions for that.

3. **python** *PyTorch implementation* (13 points)

Define the above network in your code. You may in the beginning reduce the number of neurons from 100 to 10 in the two layers. Use the class structure and write the `forward()` function.

Write the training loop and train the network.

- **Loss:** Cross-entropy
- **Optimizer:** Stochastic gradient descent (SGD)
- **Number of epochs:** 10

Your code should print the loss after each epoch and the test set accuracy after training.

Return the following items:

- Python code: mlp.py that does all above.
- PNG image: your full desktop screenshot that includes a terminal where the python file is executed and it prints all above numbers: mlp_screenshot.png (see the example)

4. **pen&paper** Count the number of trainable parameters (2 points)

   a) The MLP - "vanilla" - neural network for $64 \times 64 \times 3$ RGB input is replaced by a CNN.

   The CNN model consists of

   - the first layer is 2D convolution layer of 10 filters of the size 3×3 with stride 2 and ReLU activation function.
   - The first layer is followed by a 2×2 max pooling layer.
   - The max pooling layer is followed by another convolutional layer with the same parameters as the first.
   - The second convolutional layer is followed by another max pooling layer of the same parameters.
   - The max pooling output flattened and followed by a full-connected (dense) layer of two neurons with sigmoid activation function.

   Compute the output size of each layer. Also compute the total number of parameters (weights).

5. **python** *PyTorch implementation* (13 pts)

Implement the CNN class in PyTorch. Training and test data is the same GTSRB dataset and the same train-test split is used (use a fixed random seed).

Write the training loop and train the network.

- **Loss:** Cross-entropy
- **Optimizer:** SGD
- **Batch size:** 32 (implement a data loader)

- **Number of epochs:** 20

Your code should print the loss after each epoch and the test set accuracy after training.

Return the following items:

- Python code: cnn.py that does all above.
- PNG image: your full desktop screenshot that includes a terminal where the python file is executed and it prints all above numbers: cnn_screenshot.png (see the example)