TECHNISCHE UNIVERSITÄT DRESDEN

Professur für Informationstheorie und maschinelles Lernen, Fakultät Elektrotechnik und Informationstechnik

Prof. Rafael Schaefer, Dr. Martin Mittelbach, Muah Kim, Dr. Rick Fritschek, Max Wolf 13.06.2024

2. Homework of the Course
Einführung in das maschinelle Lernen

The homework can be submitted by ***08.07.2024, 23:59****.*

Please upload your results via the OPAL repository of this course using the available upload folder named "2. Hausaufgabe".

Upload your results as a single ZIP-file with the name firstname-lastname-homework-02.zip.

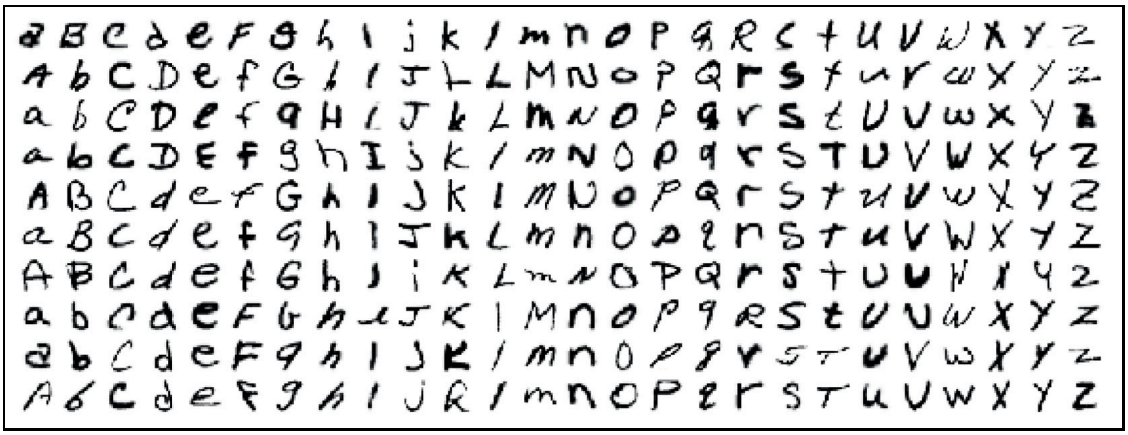
Homework Problem 3: (EMNIST image classification with CNN and PyTorch)

We consider the EMNIST dataset. It is similar to the MNIST dataset but contains not only labeled images of handwritten digits but also of handwritten letters. Please download the dataset using the following link

https://tud.link/ajpy8t

and unpack it into the folder where you execute the PYTHON code to be developed for this homework problem.

The EMNIST dataset can then be handled similar to the MNIST dataset using PYTORCH (module `torchvision.datasets`). We consider a subset containing only letters, which is loaded using `torchvision.datasets.EMNIST(...,split='letters',...)`.



The dataset has 26 classes, one class for each letter, where corresponding small and capital letters are joined to one class. For each class there are 4800 training images and 800 testing/validation images such that the training set contains 124,800 samples and the testing/validation set 20,800 samples. All images are grayscale of size 28×28 pixels.

The goal is to implement and train a convolutional neural network (CNN) with PYTORCH such that the images of handwritten letters are classified with high accuracy, where the training image set is used for network training and the testing/validation image set for validation.

The CNN architecture to be considered has 3 convolutional layers (2-dimensional) followed by 3 fully connected linear layers with the following specifications:

- *Convolutional layers*:
  - Each convolutional layer has kernel size 5×5, stride being equal to 1, and `ReLU` activation at the output.
  - The padding of the first convolutional layer is such that the output image size is equal to the input image size. The padding in the remaining convolutional layers is equal to 0.
  - Max pooling is applied (only) prior to the input of the 3rd convolutional layer and prior to the input of the 1st linear layer to reduce the image resolution by a factor of 2 in each dimension.
  - The number of output channels doubles from one to the next convolutional layer starting with 8.
- *Linear layers*:
  - The number of output nodes in the 1st and 2nd linear layer are 128 and 64 with `ReLU` activation.
  - At the output layer an activation function suitable for the considered classification task is used.

Sub-problems:

a) (0.5 points) Visualize a selection of sample images from the training dataset and the testing/validation dataset. Use the numerical class labels *and* the corresponding letter labels as image captions.
Note: To display the images of the considered dataset correctly they need to be 'transposed' (horizontal flip and then 90-degrees-anti-clockwise rotation).

b) (1 point) Calculate the size of all layers' dimensions correctly. Calculate the total number of trainable parameters.

c) (1.5 points) Implement the CNN and the training algorithm written below by using PyTorch. Everything that is not specified is of your choice.
  - Adam optimizer
  - training with mini-batches
  - default PyTorch initialization of network parameters
  - cross-entropy loss

  Note: Be careful! After loading the EMNIST training and test datasets the numerical labels of the handwritten letter images are in the range $1, 2, \ldots, 26$!

d) (2 points) Train the CNN to classify the 26 classes of handwritten letter images using the *training* dataset. Train the CNN until the validation classification accuracy of the *testing/validation* dataset is at least 92%. Plot the classification accuracy of the *training* dataset and the *testing/validation* dataset over the number of epochs. Provide the model parameters after successful training.

e) (1 point) For the trained model illustrate the classification result of the *testing/validation* dataset with a *confusion matrix* displaying the number of correctly and incorrectly classified samples for all class combinations. You can use the module `sklearn.metrics` for this task. On the axes of the displayed confusion matrix use the letters $\{A, B, C, \ldots, Y, Z\}$ as labels instead of numerical values.

f) (1 point) Implement a function that allows to visualize a selection of images from a single class of the (*testing/validation* or *training*) dataset, where the (true) class label is a parameter of the function. Use the true and the predicted letter labels (for the trained model) as image captions. Show example illustrations.

g) (1 point) Implement a function that allows to analyze misclassifications of the trained model based on the confusion matrix of a considered dataset. The function should determine for the (training or testing/validation) dataset and a specified (true) class label the most frequently misclassified predicted class. The function should allow to visualize a selection of images of these misclassifcations with the true and the predicted letter labels as image captions. Show example illustrations.

h) *Extra task*: (1 extra point) Find a CNN architecture with at least 20% fewer trainable parameters, which achieves a validation classification accuracy of at least 92% for the *testing/validation* dataset. Provide the model parameters after successful training.

Notes

- What should be submitted is your calculations of Task b), the source code you implemented and used, the trained model(s) (use corresponding PyTorch functions for saving) of Task d) (and maybe Task h)), and the illustrated figures of Tasks a), d), e), f), g) you plotted.