Technische Universität Bergakademie Freiberg
Institut für Numerische Mathematik und Optimierung
Prof. Dr. O. Rheinbach
Dr. S. Köhler

Wintersemester 2024/25
14.01.2025

Praktikum Wissenschaftliches Rechnen / Scientific Programming Project

Programming Project

Introduction

For the scientific programming project, you should work in groups of 2-3 people. Your task is to implement and train a *convolutional neuronal network* for image classification. The network should be trained on self-generated data sets (10 classes), where each group should contribute images to the whole database. Furthermore, the network should be able to classify new images correctly.

The project is divided into several steps:

1. Data generation: All groups should provided 15-20 images for each class. The classes are:

1. bottles	3. spoons	5. forks	7. T-shirts	9. chairs
2. mugs/cups	4. knifes	6. shoes	8. plants	10. bikes

    For more details, see the introcdution exercise sheet.
2. Data preparation and pre-processing (normalization, augmentation).
3. Implementation of a given CNN model from scratch in PyTorch.
4. Train the model, hyperparameter tuning and evaluation of the results.
    - Please make your program reproducible (also the data augmentation), e.g., set seeds for the random number generators.
    - Report your results, e.g., intermediate results, results of the hyperparameter tuning, different performance measurements and so on.
    - Use a 80% of the database as your training data and the rest as test data.
    - Use *k-fold cross-validation* for hyperparameter tuning.
    - Prepare your python code in a way that you can handle complete new images for a classification task (plot the classified label and image), which wil be handed to you at the time of your presentation.

Architectures of the Neural Network

The groups should implement different classical CNN-architectures of neuronal networks. However, each group should use a simple *LeNet-5*-like network as a starting point, which is then developed into a more complex architecture through parameterization or expansion. The aim is not to re-create the models *one-to-one*, but to understand their basic structure and concepts. Use simplified versions of the models if the full architecture is too complex.

LeNet-5 (as a basis)

LeNet-5 is on of the earliest CNN-architecture (Yann LeCun et al., ca. 1998), originally for MNIST hand written number recognition. It consists of a few convolutional layers, followed by pooling and simple fully connected layers. The architecture is relatively small and straightforward, which makes it ideal for getting started and learning the basic building blocks of a CNN.

Main features:

- Input: 32x32-pixel-image (adapt the data set)
- Two to three convolutional layers with subsequent pooling layers
- simple fully connected layer at the end

AlexNet Light

AlexNet (Krizhevsky et al., 2012) was a breakthrough in the ImageNet competition. It is larger and deeper than LeNet and relies more on ReLU activation functions and dropout. For our purposes, create a *simplified* version ("AlexNet Light") that has fewer parameters and layers, but retains the basic principle of deeper CNNs with ReLU and dropout.

Main features:

- more convolutional layers with differen filter sizes
- ReLU activation function for faster convergenceKonvergenz
- dropout layer for regularization

VGG-16

VGG networks (Simonyan & Zisserman, 2014) are characterized by a very deep but uniform architecture: many 3x3 convolution layers in a row, followed by pooling. To get started, we choose a simplified version of VGG-16, e.g. fewer repetitions. It is important to understand the idea of deep stacking of small 3x3 filters.

Main features:

- many 3x3-convolution-layer consecutively
- pooling layer after some convolutional layers
- deep architecture with uniform layer design

ResNet-50

ResNets (He et al., 2015) are known for the introduction of so-called *residual blocks*, which alleviate the vanishing gradient problem in deep networks with skip connections. For this exercise, a simplified version of ResNet (e.g. ResNet-18 or a reduced ResNet-50 version) is used to demonstrate the concept of residual blocks.

Main features:

- residual blocks with skip connections
- significantly deeper architecture than LeNet/AlexNet
- simplified training through direct connections from previous layers

Assignment

a) **Data set creation:** Take 15-20 images of each class, see above. These images have to be from pairwise disjoint objects, i.e., not only the same object from different perspectives (use different perspectives for different objects). Take the images by your own and do not use pre-existing images from the internet.

b) **Preprocessing:** Normalize the full image database and use data augmentation techniques to enlarge the database, e.g., cropping, flipping, rotating, and chage of colors of the images. Use the standard transform operations of PYTORCH and PIL

c) **Implement your model:** Use PYTORCH and orient yourself on examples from the lectures or documentation. Each group is assigned one of the three archtiectures:

   - AlexNet Light
   - VGG-16 (simplified)
   - ResNet-50 (simplified)

   Important: Implement the network architecture from scratch by yourself. Do not use already implemented models like `torchvision.models.alexnet`. Make everything reproducible.

d) **Training and Evaluation:** Train the model and tune the hyperparameters, e.g., learning rate, batch size, number of epochs, optimizer. Keep in mind that the hyperparameter tuning is with respect to the training data and **not** the test data. Use *k-fold cross-validation* for the hyperparameter tuning. Visualize different perfomance meassurements, e.g., trainings- and validation-accuracy, confusion matrix. Visualize the loss and accuracy curves.

e) **Documentation:** Explain your approach and decisions. Describe the data, the architecture of your model, the hyperparameters and the results. Discuss possible approaches for improvement.

Submission

Submit your code (Python-scripts), your data set by a link, e.g., TUBAF ownsky-cloud, and your documentation as a PDF. The documention should contain at least:

- description of the data set
- description of the network architecture (incl. a figure)
- training setup (hyperparameters, number of epochs, optimizer, learning rate, etc.)
- result in form of the accuracy and confusion matrix
- short discussion of the result

Literature & Ressources

- LeNet-5: `http://yann.lecun.com/exdb/lenet/`
- AlexNet: Krizhevsky, Sutskever, Hinton (2012),
  `https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-ne`
  `pdf`
- VGG: Simonyan, Zisserman (2014), `https://arxiv.org/abs/1409.1556`
- ResNet: He et al. (2015), `https://arxiv.org/abs/1512.03385`
- PyTorch Dokumentation: `https://pytorch.org/docs/stable/index.html`