# Assignment 1

# Convolutional Neural Networks for Image Classification

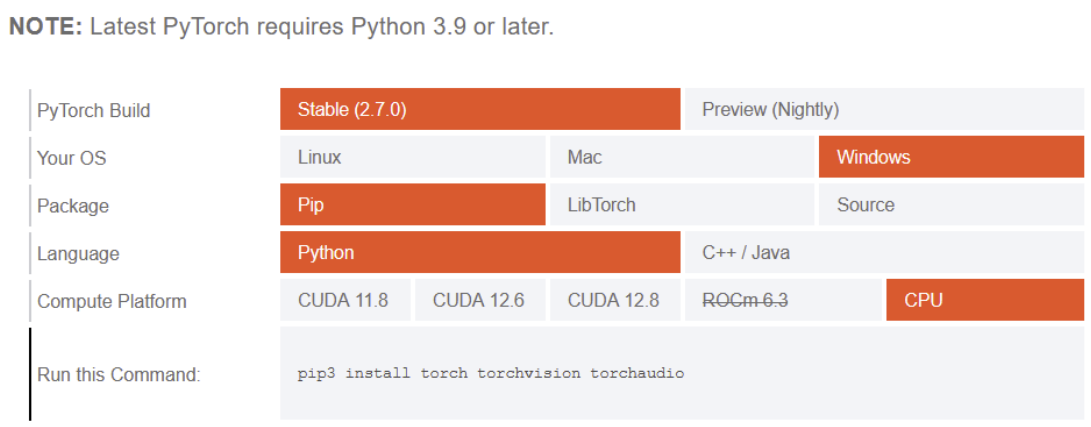
Exercise for the course *Deep Learning in Computer Vision and Remote Sensing*

Summer term 2025

**PD Dr.-Ing. Martin Weinmann** (martin.weinmann@kit.edu)
**M.Sc. Dennis Haitz** (dennis.haitz@kit.edu)

## 1 Data and Training (10 P)

For the task of training a Convolutional Neural Network (CNN) as an image classifier, the Pytorch framework is used in this assignment. Please make sure to have it installed using your preferred package manager (e.g. pip or conda). If you have a capable Nvidia Graphics Card, you can use the GPU version of Pytorch with CUDA versions 11.8, 12.6 or 12.8 depending on your device. Otherwise install the CPU version. The CPU version will be sufficient for this task, the training and prediction however will take longer. See the Pytorch website for details (https://pytorch.org/get-started/locally/).



**NOTE:** Latest PyTorch requires Python 3.9 or later.

| | | | | | |
|---|---|---|---|---|---|
| PyTorch Build | [✓ Stable (2.7.0)] | Preview (Nightly) | | | |
| Your OS | Linux | Mac | [✓ Windows] | | |
| Package | [✓ Pip] | LibTorch | Source | | |
| Language | [✓ Python] | C++ / Java | | | |
| Compute Platform | CUDA 11.8 | CUDA 12.6 | CUDA 12.8 | ~~ROCm 6.3~~ | [✓ CPU] |
| Run this Command: | `pip3 install torch torchvision torchaudio` | | | | |

Figure 1: Options of Pytorch installations.

The first task is to train an image classifier to separate between two classes (Figure 2). Load the training data from the provided folder (./data/train). In order to do that, write a *custom* dataset and dataloader class that is able to load the data and later provide it for training. See this Pytorch tutorial for implementation details: https://pytorch.org/tutorials/beginner/introyt/trainingyt.html . Test the dataloader by calling the *next*-function if it is working properly. The data needs to be accessed **randomly**. Before training starts, you need to implement a model (i.e. Neural Network) architecture. Implement the following architecture:

1. Convolution layer (32 kernels, 5x5 kernel-size)
2. Activation layer (ReLU)
3. Max pooling layer (pooling size 2x2, stride 2)
4. Dropout layer (dropout probability 0.1)
5. Linear layer (256 neurons)
6. Activation Layer (ReLU)
7. Linear layer (2 neurons)

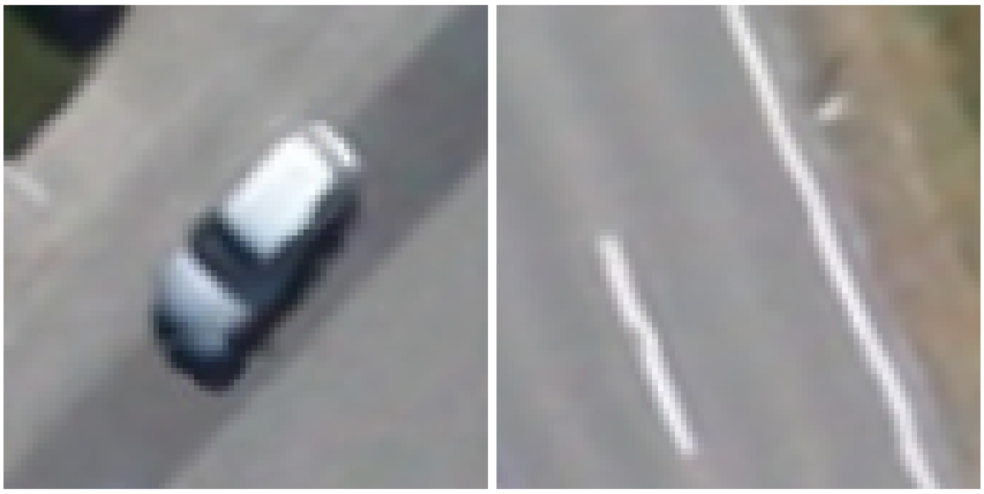


Figure 2: Samples of the dataset. Left: Example of the car class. Right: Example of the not_car class.

Train the model within a training routine, using your model and the dataloader class. Also perform a dataset normalization based on the mean and standard deviation of the dataset. Look for the normalization function in the torchvision library. The images need to be scaled from [0, 255] to [0.0, 1.0]. Run the training with the following settings:

- 40 epochs
- Adam-Optimizer
- Batch-size: 32
- Learning-Rate: 0.0001
- Cross Entropy loss function

The Pytorch tutorial (https://pytorch.org/tutorials/beginner/introyt/trainingyt.html) shows how to prepare and execute the training. Print the training loss every 50 iterations to the console. Print the average loss after each epoch.

## 2 Predefined Architectures (5 P)

Load a small network (e.g. MobileNet, SqueezeNet, ResNet18) from the torchvision library. Train this network according to task 1. If applied on CPU this might take a while, you can try to reduce the number of epochs to 30. If the input images do not match the network architecture in size, resize the images to the size the network accepts. As a side note: Always keep in mind that resizing includes interpolation (Nearest Neighbour, Bilinear, etc.) which removes information from the original images.

## 3 Testing (5 P)

For testing, classification metrics need to be calculated. Use the trained models from Tasks 1 and 2 and perform predictions on the test data (.data/test). Note that the raw output of the network is not a classification result. From the predicted output, calculate

- Confusion Matrix
- Precision
- Recall
- $F_1$-Score
- Overall Accuracy.

You can use TorchMetrics (https://lightning.ai/docs/torchmetrics/stable/) or the Scikit-Learn metrics module (https://scikit-learn.org/stable/index.html) for convenience.

## 4 Questions (10 P)

Answer the following questions and send them as a PDF with your code.

1. What is the difference between an iteration and an epoch in this training scenario? (0.5 P)
2. What do the values in the last (linear) layer tell you? How are they called? (0.5 P)
3. Which function do you need to apply in order to receive an actual classification result? (0.5 P)
4. Why is Cross Entropy an adequate loss function for classification? Could you use an L2-Loss? Why/why not? (0.5 P)
5. In Pytorch, a Softmax-Function is integrated into the Cross Entropy loss. What is a Softmax-Function and what is its effect on the output? (1 P)
6. Explain briefly and precise what leads to model overfitting. (1 P)
7. How would an overfitting of the classifier affect the classification metrics? (1 P)
8. Which model performed better in your tests? Why did it perform better? Explain in detail. (5 P)