The final project requires you to put together most of the topics covered in the lecture. Your task is to train a recurrent neural network on time-series. More specifically, you will try to predict the training performance of a CNN on the CIFAR-10 dataset based on its hyperparameter configuration and intermediate training results. This project was chosen for several reasons:

1. The problem amends itself to different tasks. You will look at regression and time-series prediction, but one could think of other things, too.
2. The dataset is rather small to keep computational costs low and allow you to go through multiple iterations of models of growing complexity.
3. The problem is easy enough to not require very complex models to achieve good performance, but easy enough to yield good results within the scope of this project.

The data consists of learning curves of neural networks trained with different hyperparameters. The names of all hyperparameters and their ranges are summarized in the following table:

Table 1: Hyperparameters of each learning curve.

parameter	min value	max value	log
batch size	32	512	no
learning rate	10^{-6}	10^{0}	yes
number of units in layer 1	16	1024	yes
number of units in layer 2	16	1024	yes
number of units in layer 3	16	1024	yes

In this project, you will tackle two problems:

prediction of the final error In this scenario, the error of a given configuration after a fixed number of epochs should be predicted. This problem occurs, for example, in hyperparameter optimization of neural networks when one tries to model the error rate after training.

extrapolation Here, a given part of a learning curve has to be extended beyond the observed points. Early stopping would be a good example where this problem becomes relevant.

On page 2 you find a detailed list of tasks, and we will release a list of plots you should prepare for the oral exam by the end of the semester. For all steps, do some basic hyperparameter optimization for all models for a fairer comparison. This avoids problems where two methods require very different hyperparameters to achieve their best performance. For our problem this is feasible, but for more resource-demanding applications in deep learning it might not.

1. Download the data and the corresponding notebook from ILIAS. It will show you how to load the data. Bring the data into a suitable format by converting it into arrays.

2. A MLP for the *prediction of the final error* task

   - Implement a simple MLP with two hidden layers with 64 units each, and ReLU activation functions.
   - Train it on the raw data, and on a preprocessed version where every input dimension has zero mean and unit variance. Compare those to a baseline of your choice using 3 fold cross validation (CV).
   - Study the influence of at least 2 regularization methods from the book.
   - Implement an exponential learning rate schedule (and tune it) to achieve better performance.

3. A recurrent network for the extrapolation task

   - Implement a 2 layer LSTM with a 2 layer MLP on top, both with 64 units each. Naturally, the configuration should be considered by the RNN. Implement and discuss the way you provide the configuration and the last observed value to the network as inputs and how they are used for prediction.
   - Train a network using only the first 5, 10, and 20 epochs of your training curves, respectively. Measure the models' MSE for the final point at epoch 40 on the training and test data for a varying input length (5, 10, 20, 30). As a baseline, train a model that predicts the final error rate from a fixed number of observations. Again, use 3-fold CV to split into train and test data.
   - Train another model, but this time use a randomized length (uniform between 5 and 20 epochs) for the sequences during training. Reevaluate the MSE as above.
   - As another baseline, train a simple regressor that uses the configuration and the last 4 points to predict the next one. Train it on the whole training data and compare to the RNNs trained earlier.