Deep Learning Lab - Report on the 4th Exercise

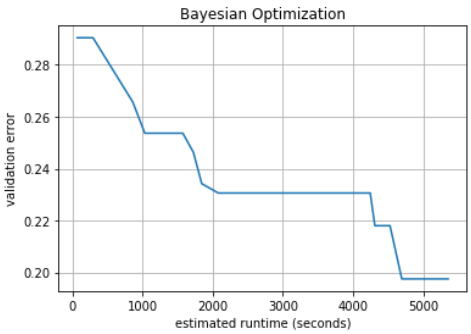
Joshua Heipel

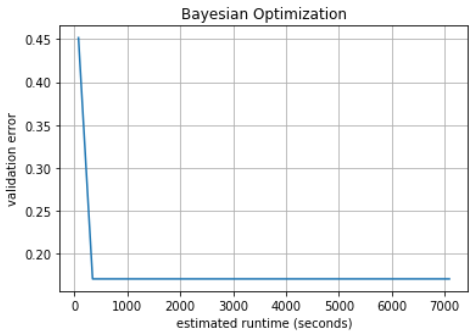
matriculation number: 3706603

Bayesian Optimization (BO)

In the first part of the exercise Bayesian Optimization (BO) method was implemented in order to find the best hyper-parameter settings, that minimize the (approximated) loss of a fully trained CNN. Therefore, after randomly sampling n initial points from parameter space and evaluating the loss function y for each point x_i, a Gaussian Process model (GP) $\hat{y}$ is fitted as an approximation to the (unknown) loss function $y(x^*)$ for unseen configurations x^*. To find a new point x_{new} where, according to the GP model, $\hat{y}(x_{new})$ is expected to be minimal, the so called expected improvement (EI) is maximized. Then the loss function is evaluated at the new point x_{new} and the GP is updated. The steps are finally repeated for a given number of iterations. This optimization procedure tries to balance the exploitation vs. exploration dilemma, searching for configurations that are not too close to already evaluated points and not too far from the current best solution (incumbent). The acquisition function and the kernel of the GP control the amount of exploration respectivly exploitation.

As can be seen in the following figures, the incumbent trajectory depends on the initially sampled parameter settings x_i. Plateaus indicate, that the evaluated loss of new configurations x was higher than the current incumbent (no improvement). In some cases BO finds a good configuration close to the optimum very fast and no improvement is achieved with the following iterations (right). In other cases the incumbent trajectory slightly improves in nearly every optimization step (left).

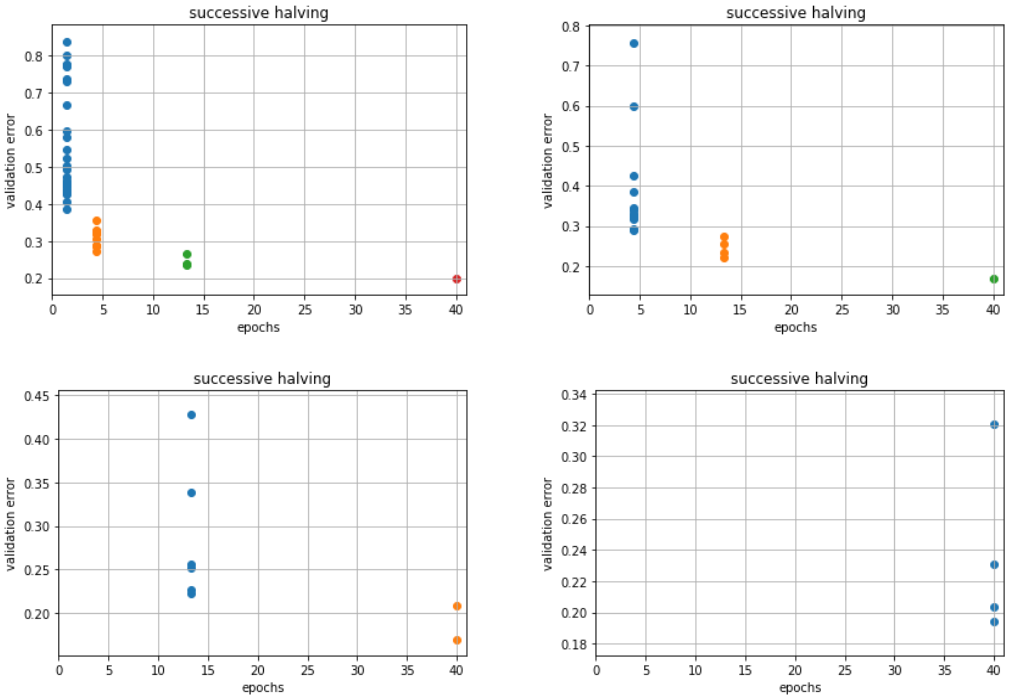




2. Hyperband (HB)

In the second part of the exercise Hyperband Optimization (HB), which combines Random Search (RS) and Successive Halving (SH), is used to optimize the CNN hyper-parameters. Here the outer loop of HB controls a fixed total budget (e.g. total number of epochs) that is distributed over different randomly sampled configurations x and evaluated by SH. Depending on the individual learning curve, training of each configuration x_i is stopped after a fixed number of epochs. As only the most promising configurations are fully evaluated, the method tries to save limited ressources, thereby speeding up the parameter search.

The following figures show how different initial random configurations are successively evaluated. In each iteration i of the SH algorithm (different colors) only the best n_i configurations are retained. The different subplots visualize subsequent steps s of the outer HB algorithm. In each step the total budget is distributed over a different number of random configurations and reduction rates.



3. BOHB

In the last part of the exercise Bayesian optimization and Hyperband (BOHB) were combined to further improve runtime and overall performance of the hyper-parameter optimization procedure. Similar to HB the outer loop of BOHB balances the total budget used for the optimization. Instead of randomly sampling initial configuration for SH from the parameter space, new configurations are drawn by (approximately) optimizing a gaussian process model. Therefore the already evaluated points x are split into "good" x^+ and "bad" x^- configurations according to the evaluated loss $y(x)$ (*update()* method). Then, to find a new paramter setting x the ratio of the kernel density estimators $l(x^+)$ and $g(x^-)$ is (approximately) maximized by drawing a limited number of samples from l and evaluating $\frac{l}{g}$ (*sample()* method). In order to theoretically guarantee finding the optimal solution x_{opt} also a few random samples are generated. The generated initial samples are finally successively evaluated by SH. As I couldn't figure out the given code with the main loop for BOHB (which I think contained some small mistakes) I wrote my own implementation, that makes use of the implementations of BO and HB from exercise (1) and (2).

In the figure below (which shows the overall validation error of different configurations over time), configurations generated by the BO model are successively improving, while randomly drawn samples might lead to bad configurations even when optimization further progresses. This speeds up the optimization procedure compared to BO or HB alone (not shown).