Deep Learning Course 2018 *Vanditha Chandrashekar Rao*
Autonomous Intelligent Systems
University Freiburg *Date: 08.01.2019*

Deep Learning Lab Course 2018 - Exercise 4

Introduction

In this exercise the goal is to implement Bayesian optimization, Hyberband and the combination of Bayesian optimization and Hyberband to optimize the hyperparameters of a convolutional neural network (CNN) on CIFAR-10.

The model consists of three convolutional layers (with RELU activations and batch norm) and one fully connected layer. The weights are optimized using Adam optimizer. In order to avoid spending time training CNNs (original benchmark), surrogate benchmark is optimized.

A surrogate benchmark is a regression model (here, random forest) that was trained on a large set of randomly sampled hyperparameter configurations of the original benchmark. To evaluate a hyperparameter configuration, the prediction of the regression model is used, since it is much faster to evaluate (milliseconds). Surrogate benchmark is only an approximation of the true benchmark.

Bayesian Optimization

Bayesian Optimization is a method that uses some kind of approximation. Bayesian Optimization helps to find a best model among many. It internally uses a model to guide the search. This optimization technique is based on randomness and probability distributions.

Acquisition functions: Proposing sampling points in the search space is done by acquisition functions.

For Bayesian optimization, emukit - a python based toolbox - is used. The configuration space for our CNN surrogate benchmark. In this exercise we will optimize the learning rate, batch size and the number of filters in each of the 3 convolutional layers. These hyperparameters are considered as continuous variables. The configuration space for the above mentioned hyperparameters are:

1. Learning Rate, $\alpha \in [-6, -1]$

2. Batch size $\in [32, 512]$
3. Number of filters, n $\in [4, 10]$

Hyperband

Hyperbands combines random search with successive halving to balance very aggressive evaluation with many configurations on the smallest budget and very conservative runs on the maximum budget. Hyperband runs successive halving with alternating number of configurations and budgets. Successive halving is a bandit-based technique for allocating more resources to promising configurations in a principled manner.

Combining Bayesian Optimization with Hyperband

One of the disadvantages of Hyperband is that it draws configurations randomly and hence might take exponentially long to approach the global optimum. In this part of the exercise, the Hyperband is combined with a kernel density estimator that models the distribution of the good and the bad configurations in the input space. This enable us to find good configurations much faster.

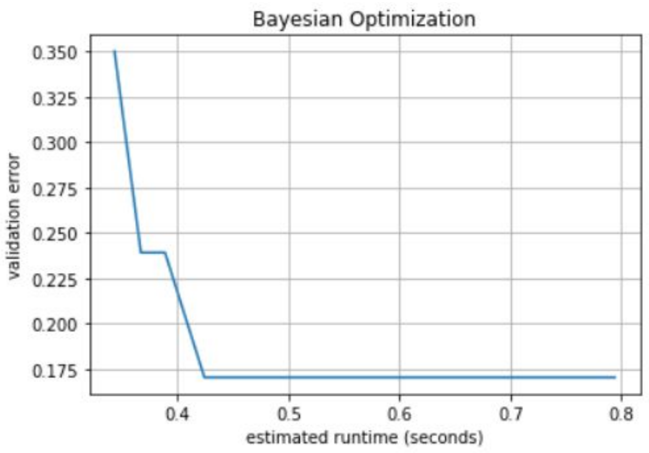
To implement the kernel density estimator, statsmodel is used. A multivariate kernel density estimator is used to fit a distribution over the whole input space.

Experiments and Results

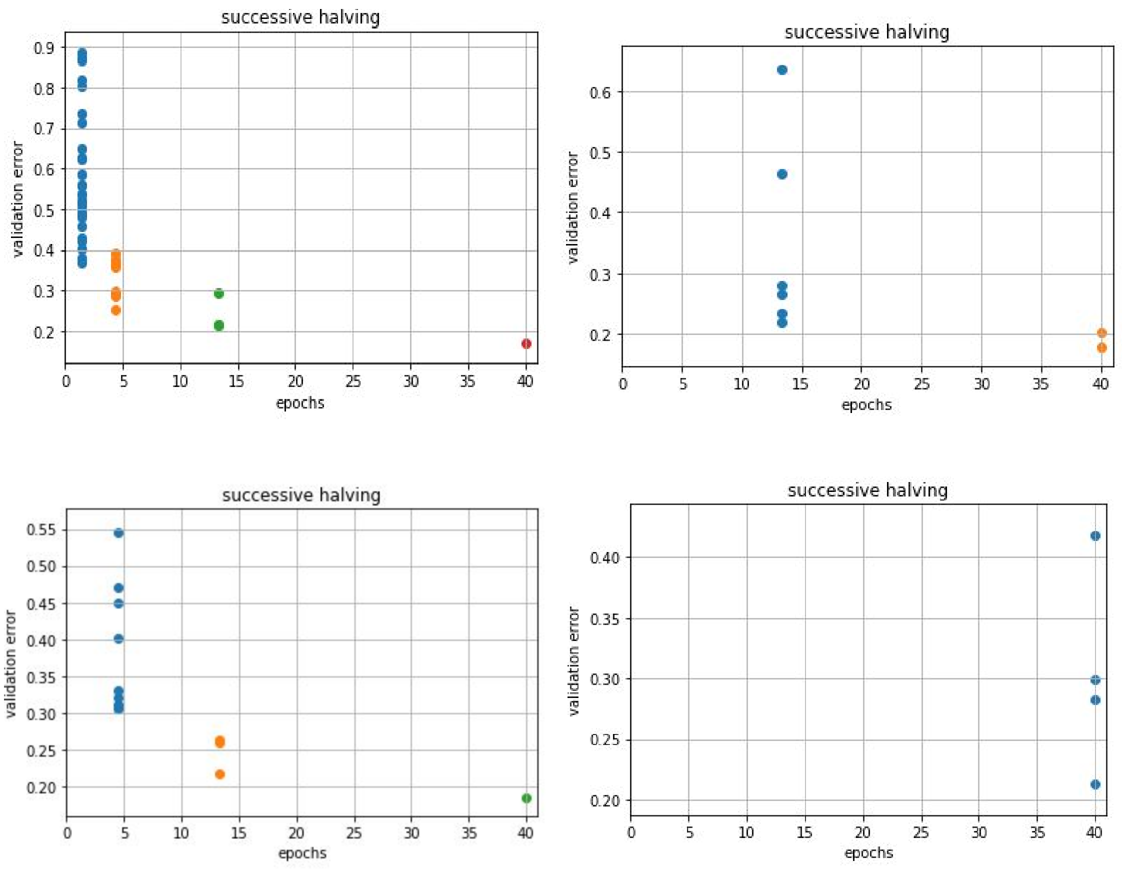
The Bayesian optimization procedure is as follows.

1. Create Initial design to guide the search.
2. Find the next sampling point by optimizing the acquisition function over the Gaussian Process (the model)
3. Obtain a possibly noisy sample y from the objective function.
4. Add the sample to initial design and update the model.

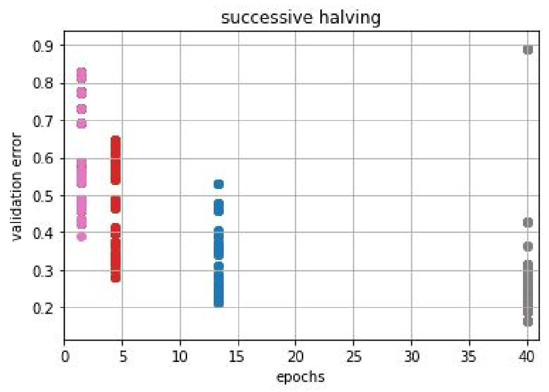
Additionally, in this exercise we will also keep track of the current best solution we have found known as incumbent an runtime. The plot of the incumbent performance over the cumulative runtime is as shown below:

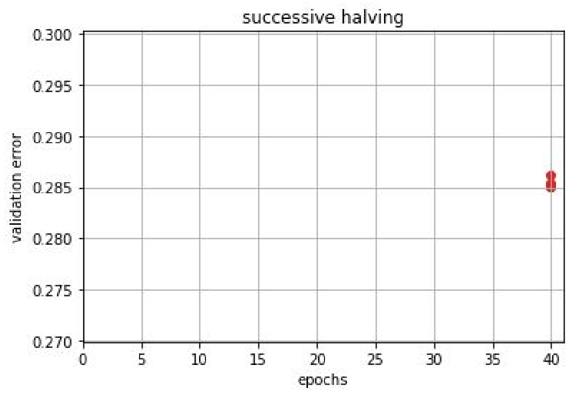


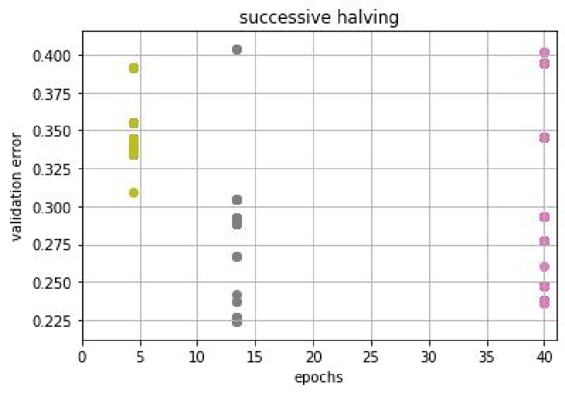
The effect of successive halving on Validation error as the number of epochs increases is shown as below:

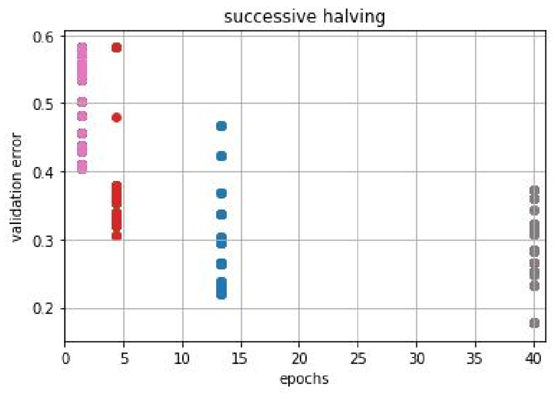


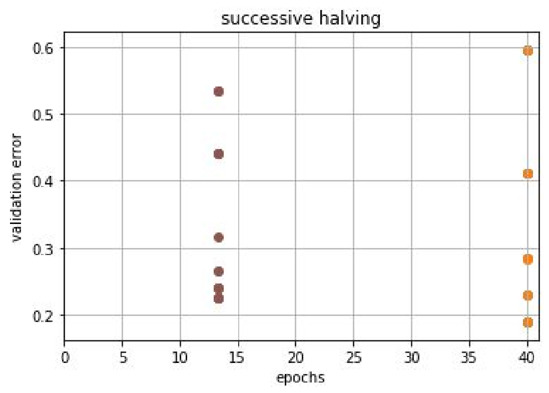
The plot below shows the incumbent trajectory after running BOHB and the visualization for each configuration whether it was sampled randomly or from our model.

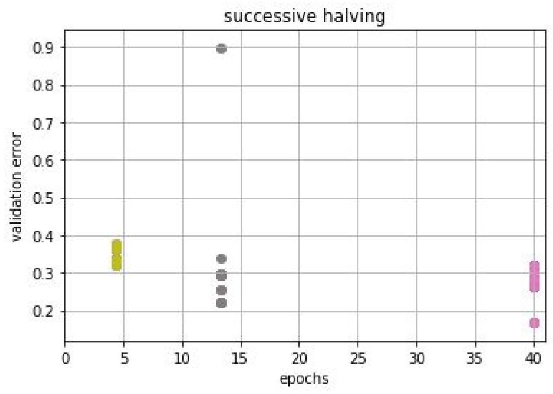

successive halving
validation error
epochs

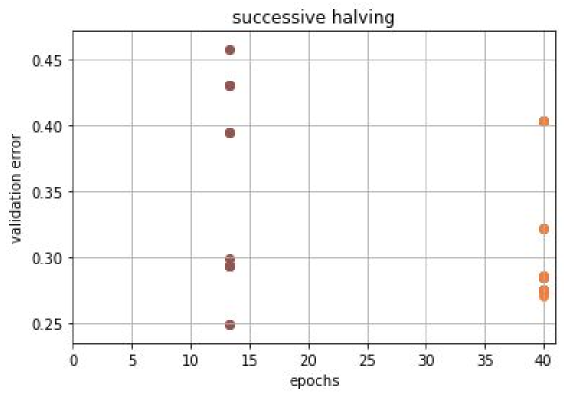

successive halving
validation error
epochs

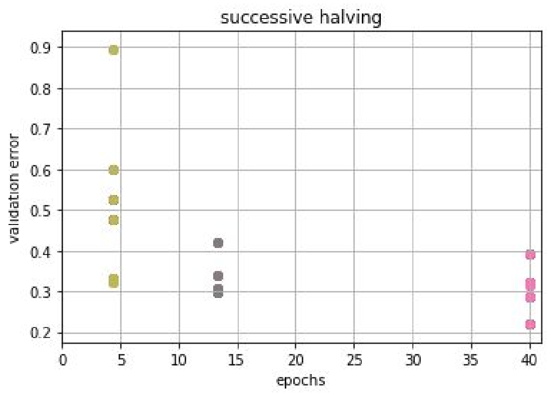

successive halving
validation error
epochs

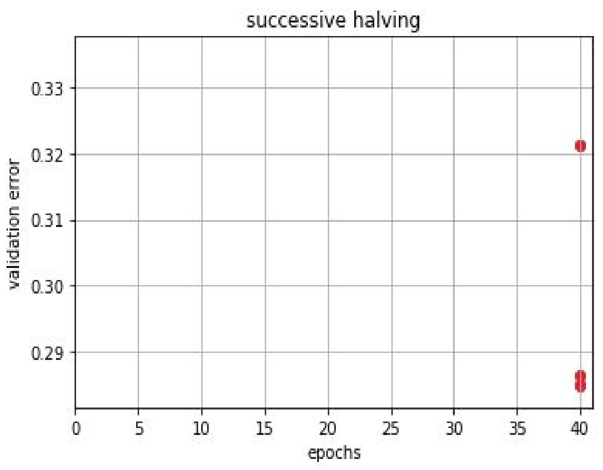

successive halving
validation error
epochs

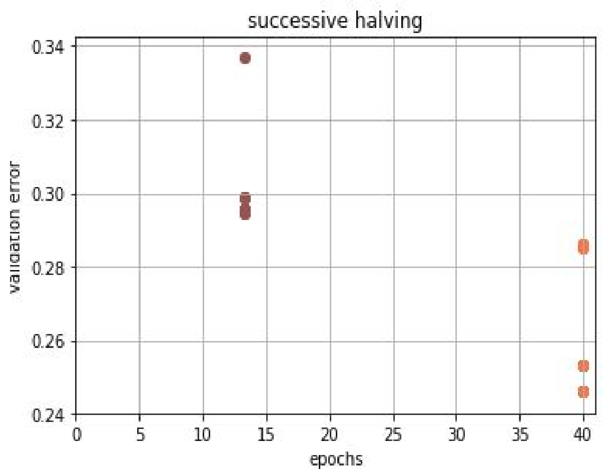

successive halving
validation error
epochs

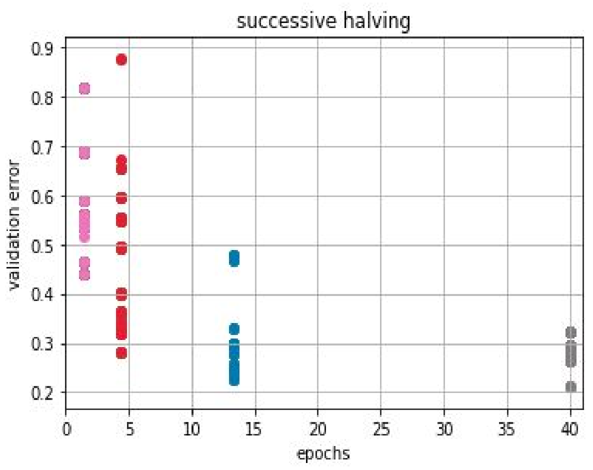

successive halving
validation error
epochs

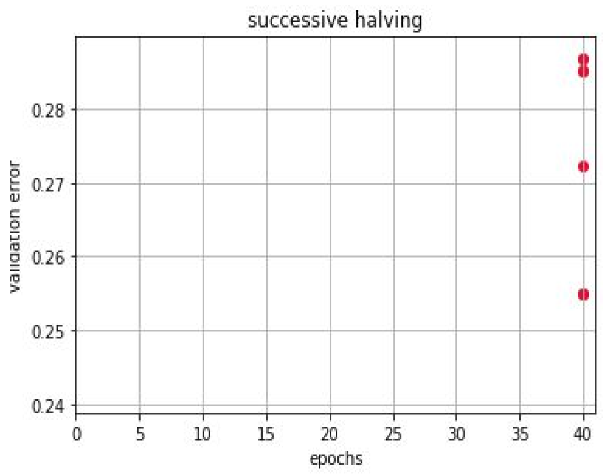

successive halving
validation error
epochs

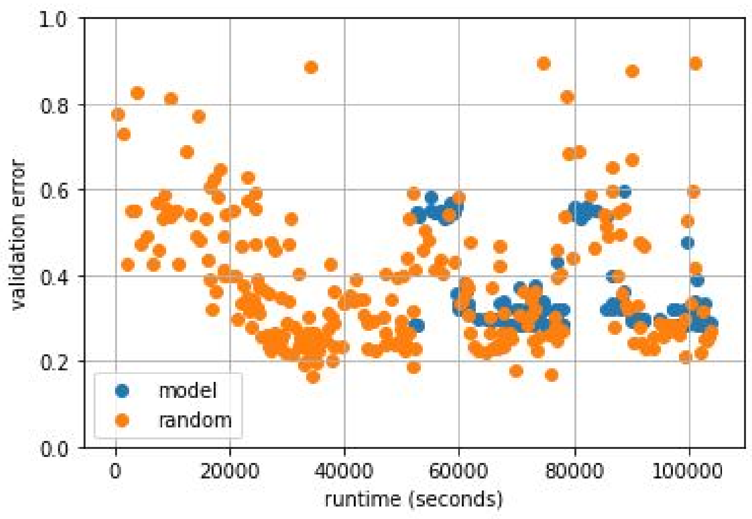

successive halving
validation error
epochs











It can be observed that BOHB converges to minimum fast and also achieves global optimum