Advanced Topics in Distributed Systems

Moritz Meister
moritz@logicalclocks.com

LOGICAL CLOCKS

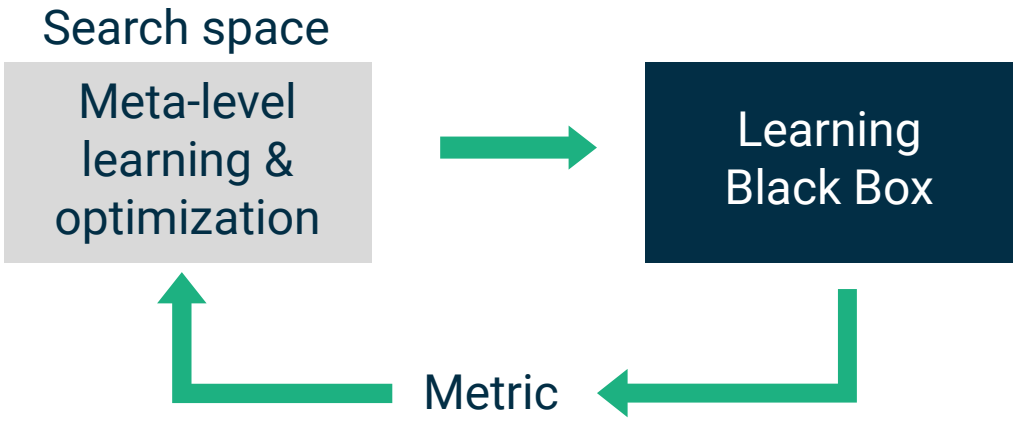
Papers

S. Falkner et al. **BOHB: Robust and efficient hyperparameter optimization at scale**.
In International Conference on Machine Learning, pp. 1436–1445, 2018.
[BOHB]

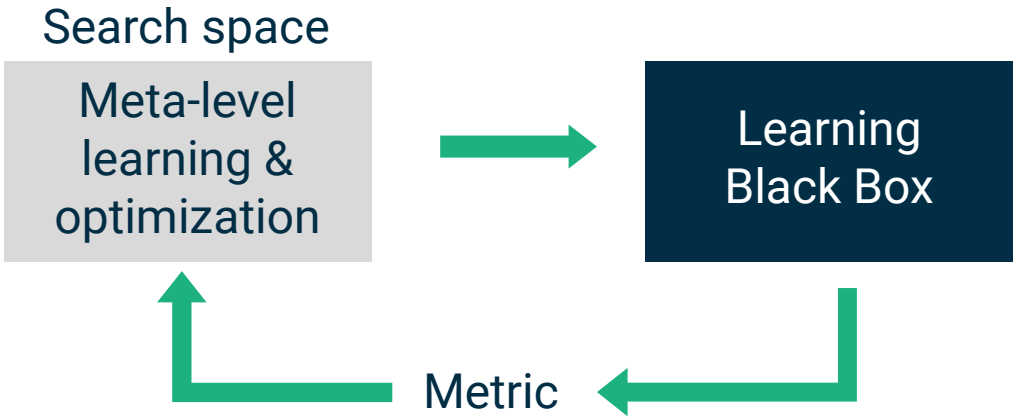
M. Jaderberg et al. **Population based training of neural networks**. arXiv:1711.09846, 2017.
[PBT]

Liam Li et al. **Massively Parallel Hyperparameter Tuning**. arXiv:1810.05934, 2018.
[ASHA]

Black Box Optimization



Black Box Optimization



How to scale this iterative loop?

What if search space is high dimensional?

Additional hyperparameters introduced?

Are different algorithms comparable?

How to define a baseline?

PBT: Problem Definition

Goal: Optimize both, the parameters and the hyperparameters jointly on the actual metric we care about.

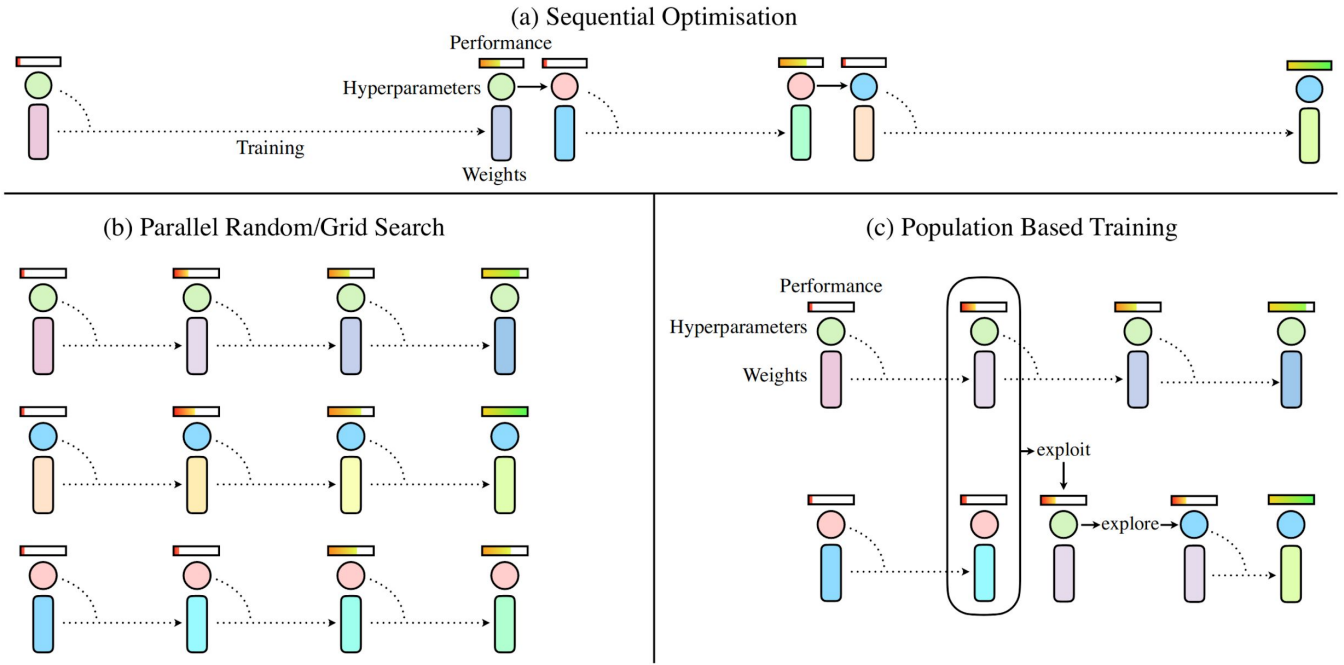
$$\theta^* = \texttt{optimise}(\theta|\boldsymbol{h}^*),\ \boldsymbol{h}^* = \arg\max_{\boldsymbol{h}\in\mathcal{H}^T} \texttt{eval}(\texttt{optimise}(\theta|\boldsymbol{h})).$$

To achieve this, a population **P** of **N** models are optimised with different hyperparameters.

PBT: Goals

1. Bridge and extend parallel search methods and sequential optimization methods
2. Wall clock run time in the orders of a single optimization process
3. Robustness for non-stationary hyperparameters
    a. Learning a hyperparameter schedule

PBT: Approach



PBT: Algorithm

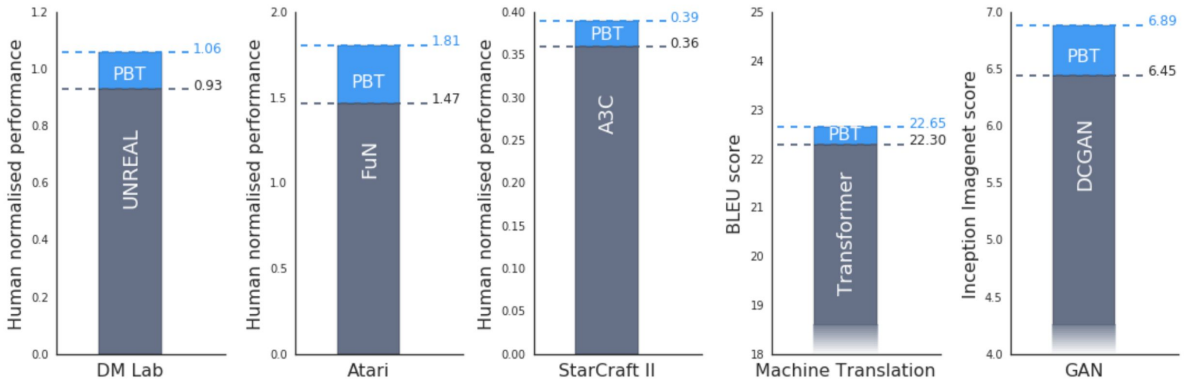
Algorithm 1 Population Based Training (PBT)

procedure TRAIN($\mathcal{P}$) ▷ initial population $\mathcal{P}$
  for $(\theta, h, p, t) \in \mathcal{P}$ (asynchronously in parallel) **do**
    while not end of training **do**
      $\theta \leftarrow \texttt{step}(\theta|h)$ ▷ one step of optimisation using hyperparameters h
      $p \leftarrow \texttt{eval}(\theta)$ ▷ current model evaluation
      if $\texttt{ready}(p, t, \mathcal{P})$ **then**
        $h', \theta' \leftarrow \texttt{exploit}(h, \theta, p, \mathcal{P})$ ▷ use the rest of population to find better solution
        if $\theta \neq \theta'$ **then**
          $h, \theta \leftarrow \texttt{explore}(h', \theta', \mathcal{P})$ ▷ produce new hyperparameters h
          $p \leftarrow \texttt{eval}(\theta)$ ▷ new model evaluation
        end if
      end if
      update $\mathcal{P}$ with new $(\theta, h, p, t+1)$ ▷ update population
    end while
  end for
  return θ with the highest p in $\mathcal{P}$
end procedure

PBT: Experiments

Empirical results on wide range of tasks:

1. Deep reinforcement learning
2. Machine translation
3. Generative adversarial networks



PBT: Meta-Hyperparameters

Population size

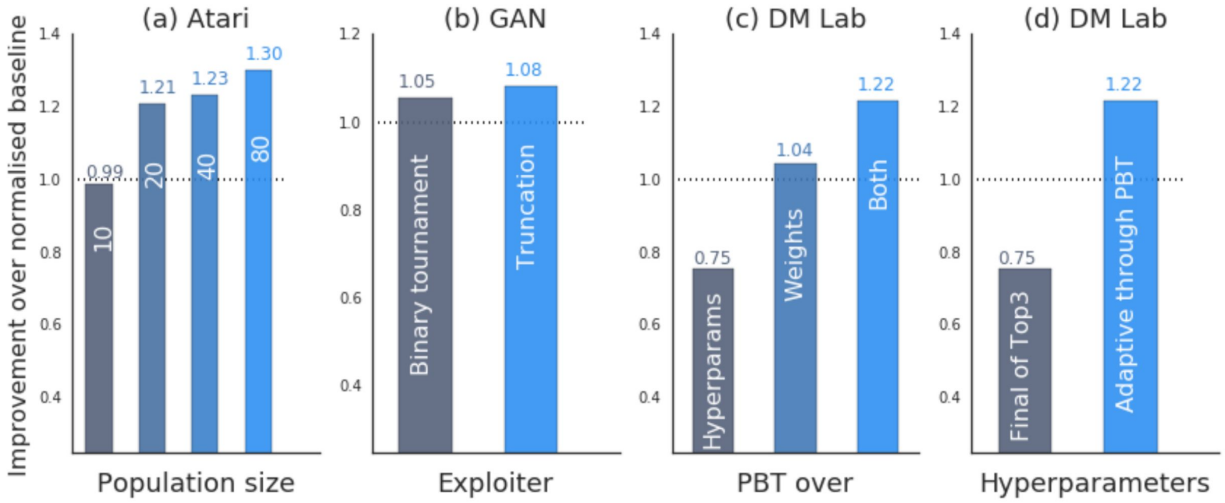
Step

Eval

Ready

Exploit

Explore



PBT: Contributions

1. Practical way to augment standard training of neural network models
2. Consistent improvements in accuracy, training time and stability across a wide range of tasks
3. Joint optimization of parameters and hyperparameters
4. Adaptive schedules
5. Decentralised and Asynchronous

Disadvantages:

1. Introduction of many new meta-hyperparameters
2. Not applicable to Neural Architecture Search, because partial solutions can’t be shared

BOHB: Problem definition

Validation performance of a machine learning algorithm can be modeled as a function

$$f : \mathcal{X} \rightarrow \mathbb{R}$$

of their hyperparameters. The hyperparameter space ***X*** can include continuous and discrete dimensions.

Objective:

$$\boldsymbol{x}_\star \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

Through noisy observations:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2_{noise})$$

BOHB: Goals

1. Strong Anytime Performance
2. Strong Final Performance.
3. Effective Use of Parallel Resources
4. Scalability in terms of number of hyperparameters
5. Robustness & Flexibility
6. Simplicity
7. Computational efficiency

BHOB: Bayesian Optimization + HyperBand

Bayesian Optimization:

- Strong final performance
- Do not scale well to high dimensions (Gaussian Processes)
- Infeasible for models with long training time (deep learning)

HyperBand:

- Exploit cheaper fidelities
- Strong anytime performance
- Scalability, robustness and flexibility

BOHB: Bayesian Optimization

BO uses a probabilistic model to model the objective function based on the set of already observed data points.

1. Select the point that maximizes the acquisition function
2. Evaluate the objective function at this point
3. Add the new observation to the data and refit the model

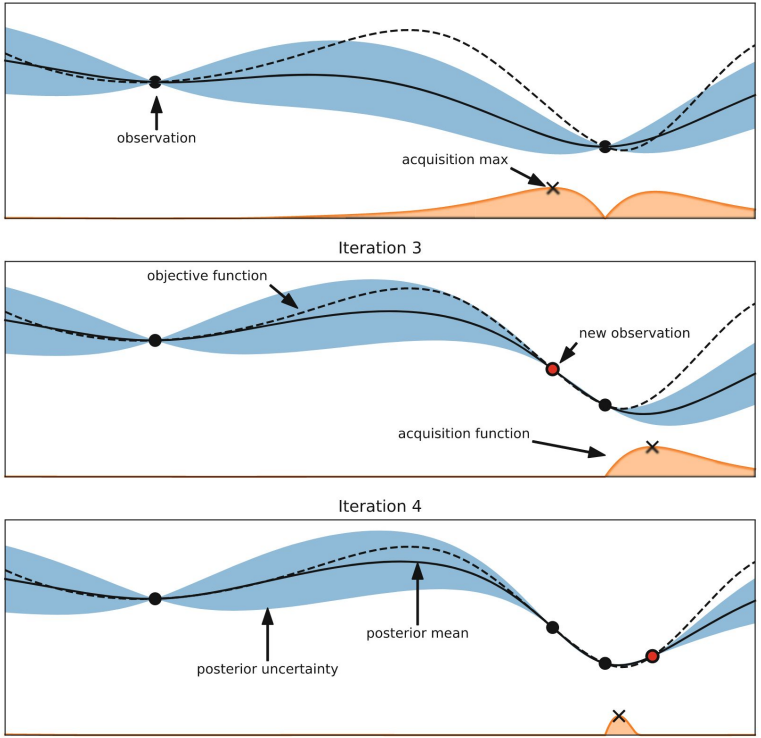
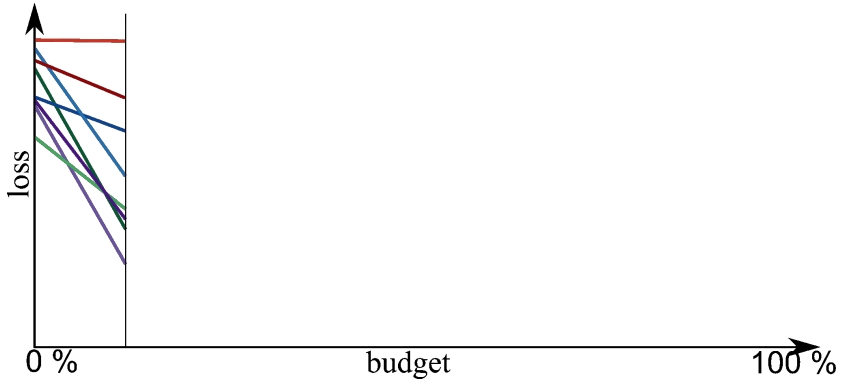


Figure: “AutoML: Methods, Systems, Challenges (Book)

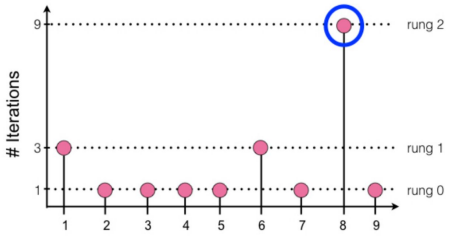
BOHB: HyperBand and Successive Halving

Hyperband (HB; Li et al., 2016) uses cheap-to-evaluate approximations of the objective function on smaller budgets.



SHA requires: number of configurations **n**, a minimum resource **r**, a maximum resource **R,** reduction factor **η** and a minimum early stopping rate **s.**

Hyperband then just loops over brackets.



bracket s	rung i	n_i	r_i	total budget
0	0	9	1	9
	1	3	3	9
	2	1	9	9
1	0	9	3	27
	1	3	9	27
2	0	9	9	81

Animation: https://www.automl.org/blog_bohb/

BHOB: Observations

Important:
Goal is to optimize on the largest budget

- Always compute the model on the largest budget for which enough observations are available, eventually relying only on the high fidelities

Keep theoretical guarantees of HB:
- sample some of the configurations still at random
- adds to global exploration and guarantees that it is only by a constant factor slower than random search but still converges eventually

BOHB is insensitive to its own hyperparameters.

Use of surrogate model to simulate model training.

BOHB: Parallelization

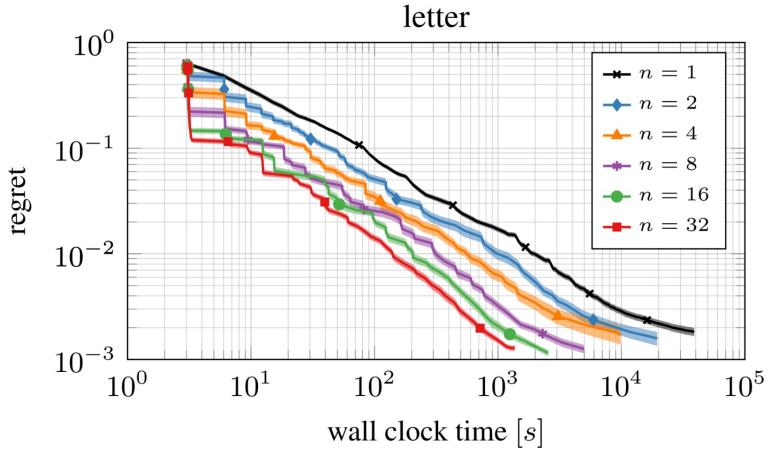
1. Start with first SH run until
   a. All workers are busy

      -> Wait until worker becomes idle to sample more

   b. Or enough configurations have been sampled for SH run

      -> Start next SH run in parallel

Observations are shared across all runs.



Performance of BOHB with different number of parallel workers for 128 iterations on the surrogate letter benchmark. The speedups are close to linear.

BOHB: Experiments

Empirical evaluation on wide range of tasks:

1. High dimensional toy functions
2. SVMs
3. Bayesian neural networks
4. Deep reinforcement learning
5. Convolutional neural networks

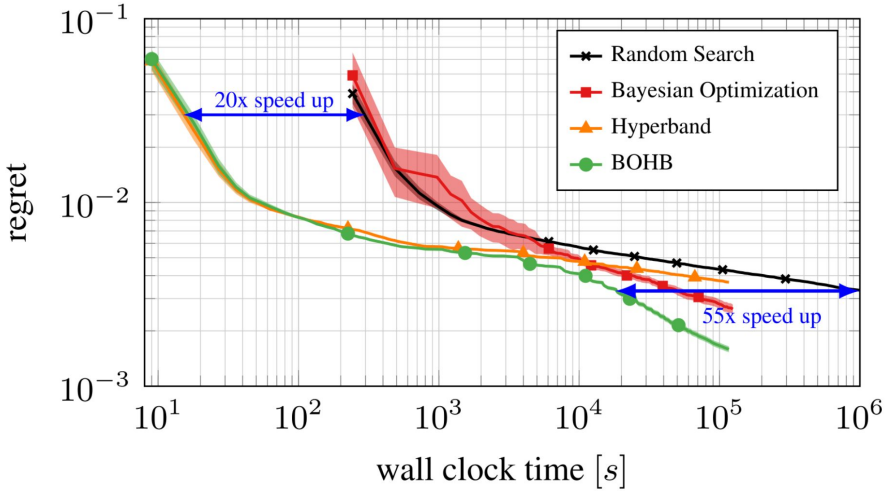


Illustration of typical results obtained exemplary for optimizing six hyperparameters of a neural network. The curves give the immediate regret of the best configuration found by 4 methods as a function of time.

BOHB: Contributions

1. A practical method satisfying the previously defined goals
2. Combination of strong anytime and final performance
3. Wide applicability, mixed discrete/continuous hyperparameter search spaces
4. Few additional hyperparameters

Disadvantages:

1. Other works find that most aggressive stopping usually performs best
2. Wall clock run time larger than single model training

ASHA: Goals

A method applicable to large-scale regime for hyperparameter optimization:

1. High dimensional hyperparameter spaces
2. Increasing training times
3. Rise of parallel computing

Evaluate orders of magnitude more hyperparameter configurations than available parallel workers in a small multiple of the wall-clock time needed to train a single model.

ASHA: Synchronous parallel SHA

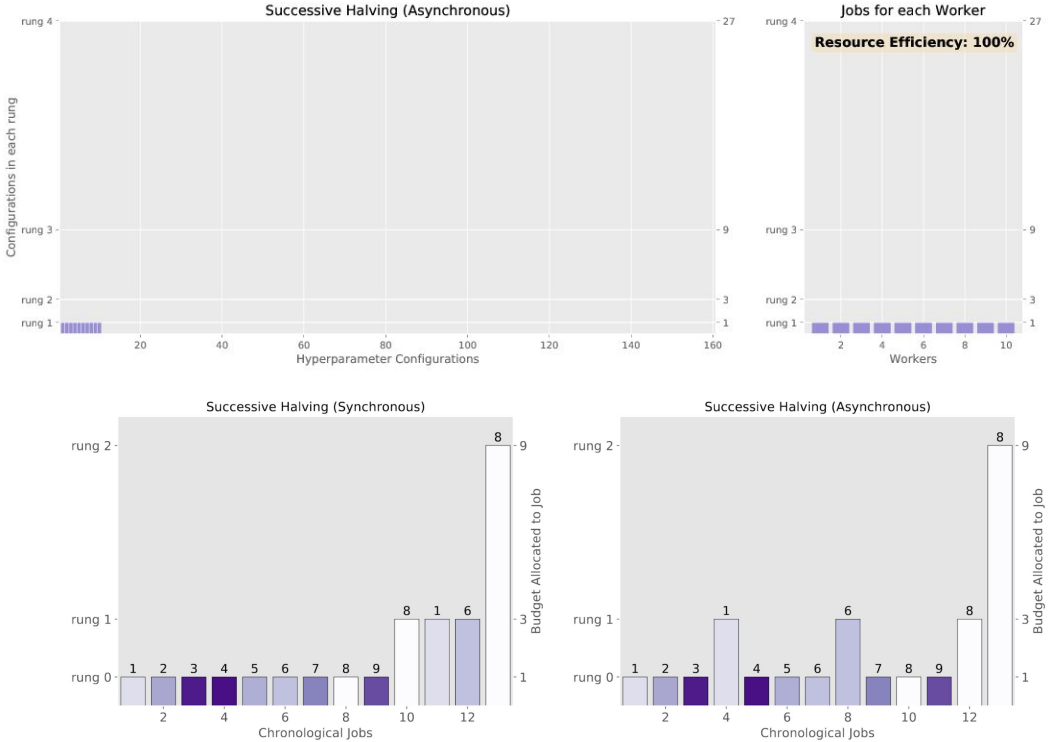
Using the parallelization scheme of Falkner et al. has two shortcomings:

1. Sensitivity to stragglers and dropped jobs.
    a. All trials in a rung need to be finished before proceeding
2. Promotions within brackets are performed independently for each bracket

Embarrassingly parallel implementation of SHA is not applicable to the large scale regime:

- Will require much more time than training single configuration

ASHA: Algorithm



* Color coding indicates performance

Animation: https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/

ASHA: Observations and Algorithm Analysis

1. Ability to give upper bound on time to return a configuration trained to completion: *2*time(R)* (possibly *time(R)* if training can be resumed)

2. Remove the bottleneck of synchronous promotions at the cost of a small number of incorrect promotions

3. Applicable in an infinite horizon setting, experiment can simply be continued if more resources (time) is available.

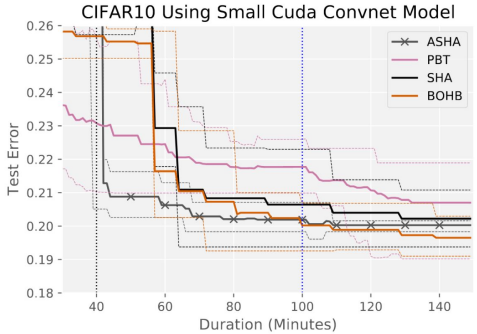
ASHA: Experiments

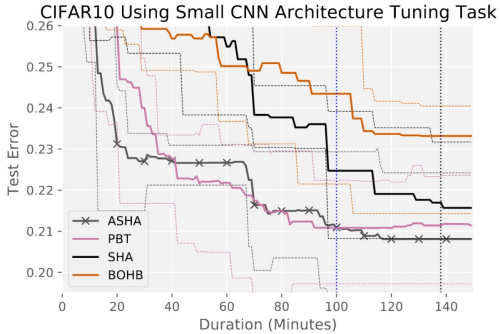
Empirical results on four different tasks:

1. Sequential comparison
    a. CIFAR10 using small cuda convnet model
    b. CIFAR10 using small CNN Architecture tuning task
2. Limited scale distributed experiments
    a. Same as above
3. Large scale ASHA benchmark
4. Modern LSTM benchmark

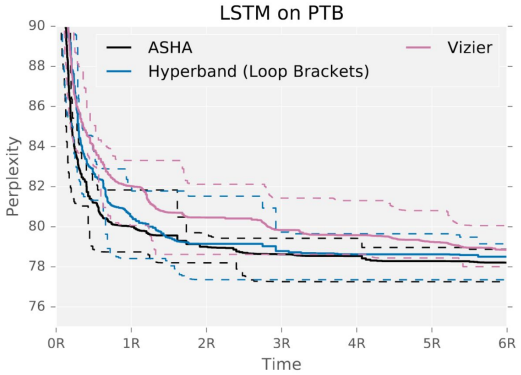
ASHA: Results

Limited-scale distributed experiment with 25 workers.

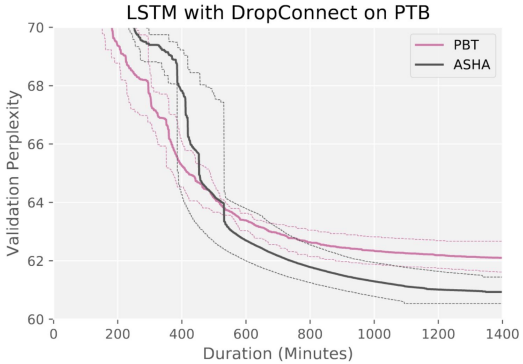




Large scale ASHA benchmark with 500 workers.



Modern LSTM benchmark with 16 GPUs. Population size 20 for PBT.



ASHA: Contributions

1. A practical method for the large-scale regime that exploits parallelism and aggressive early-stopping.
2. Motivation for the use of SHA in parallel settings.
3. ASHA finds a good configurations in approx. the time it takes to train a single model and ASHA scales linearly with the number of workers.
4. ASHA outperforms Vizier.
5. ASHA outperforms PBT on state-of-the-art LSTM model.

ASHA: Contributions

1. A practical method for the large-scale regime that exploits parallelism and aggressive early-stopping.
2. Motivation for the use of SHA in parallel settings.
3. ASHA finds a good configurations in approx. the time it takes to train a single model and ASHA scales linearly with the number of workers.
4. ASHA outperforms Vizier.
5. ASHA outperforms PBT on state-of-the-art LSTM model.

BUT: *Isn't the algorithm the main contribution and the rest are results that show the validity of this contribution?*

Future Directions

& points for discussion

- Do we need adaptive hyperparameter schedules?
- How can experiments be designed to be more comparable?
- Combining ASHA with directed samping (BO)
- Systems support for scalability