Population Based Training of Neural Networks

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojtek Czarnecki,
Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning,
Karen Simonyan, Chrisantha Fernando, Koray Kavukcuoglu

DeepMind

7th December 2017, NIPS Metalearning Symposium

Motivation

Neural Networks require optimisation to become useful.

The success of a neural network after optimisation is determined by the joint tuning of

- **Model architecture**
- **Data optimised over**
- **Details of optimisation**

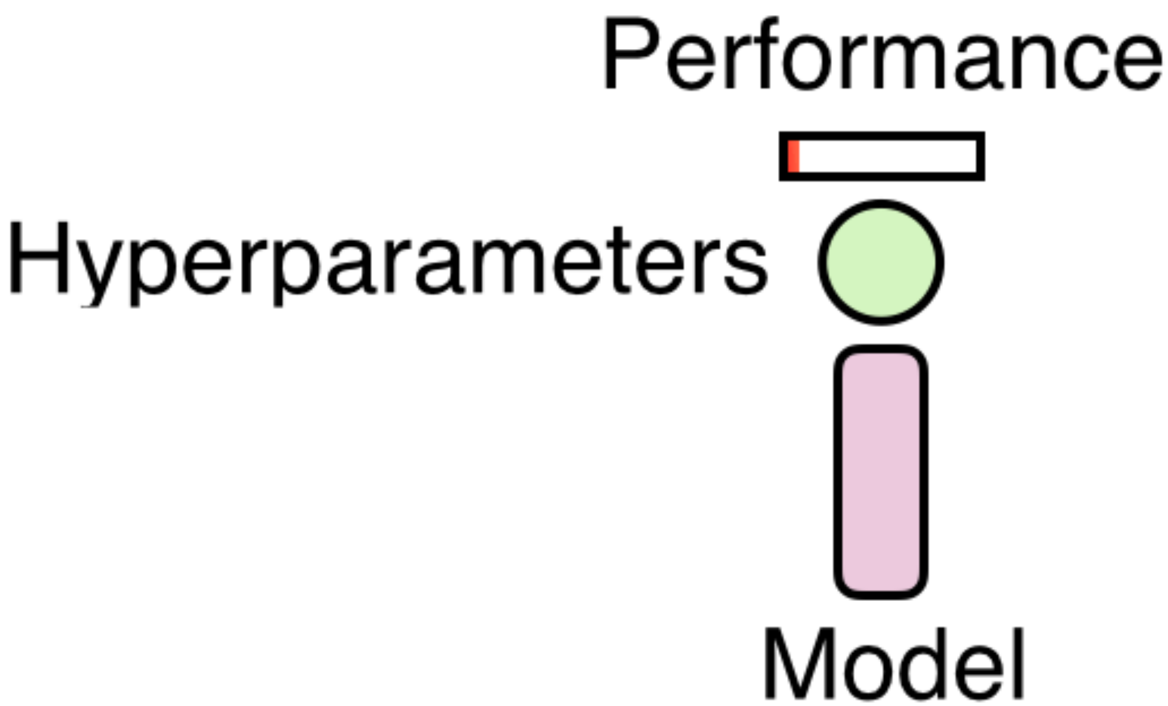
tuneable knobs are ***hyperparameters***

The correct hyperparameters are crucial to success.

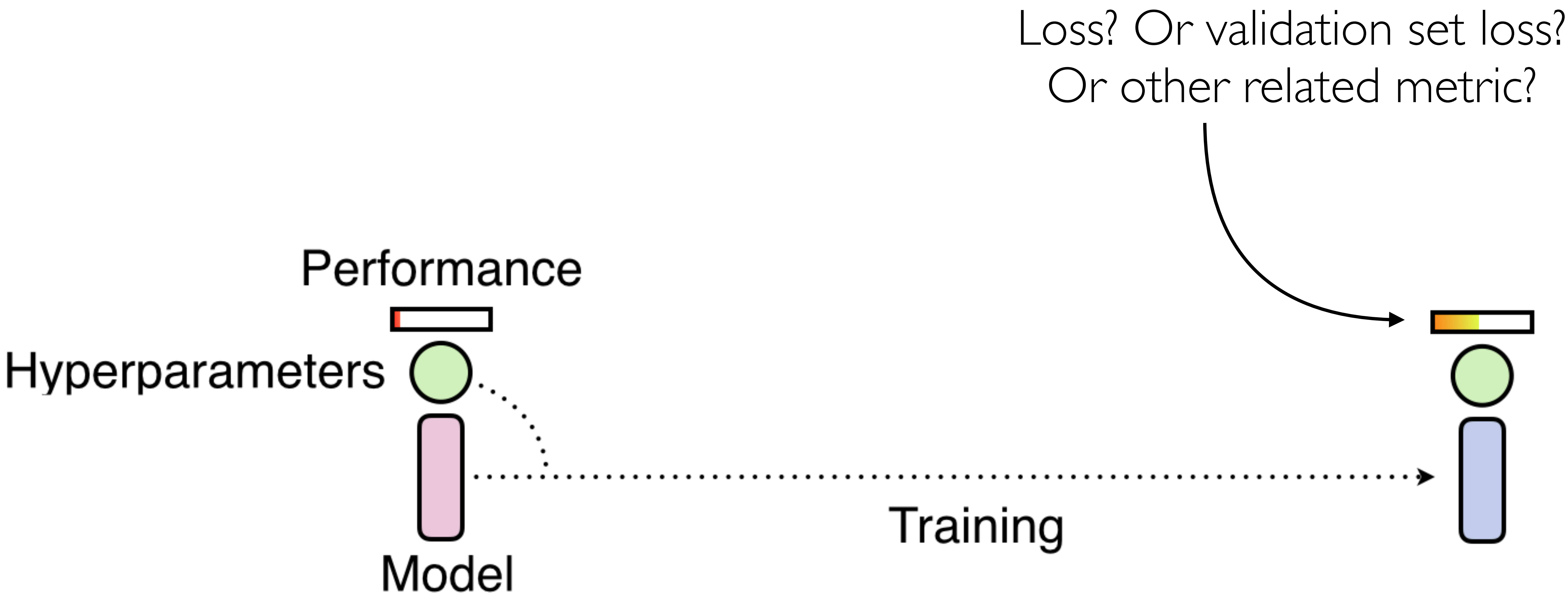
Machine learning includes tuning hyperparameters: expensive, slow.
Biases our model selection to favour tuneable algorithms.

Reinforcement Learning (RL) is highly non-stationary, requires non-stationary hyperparameters.

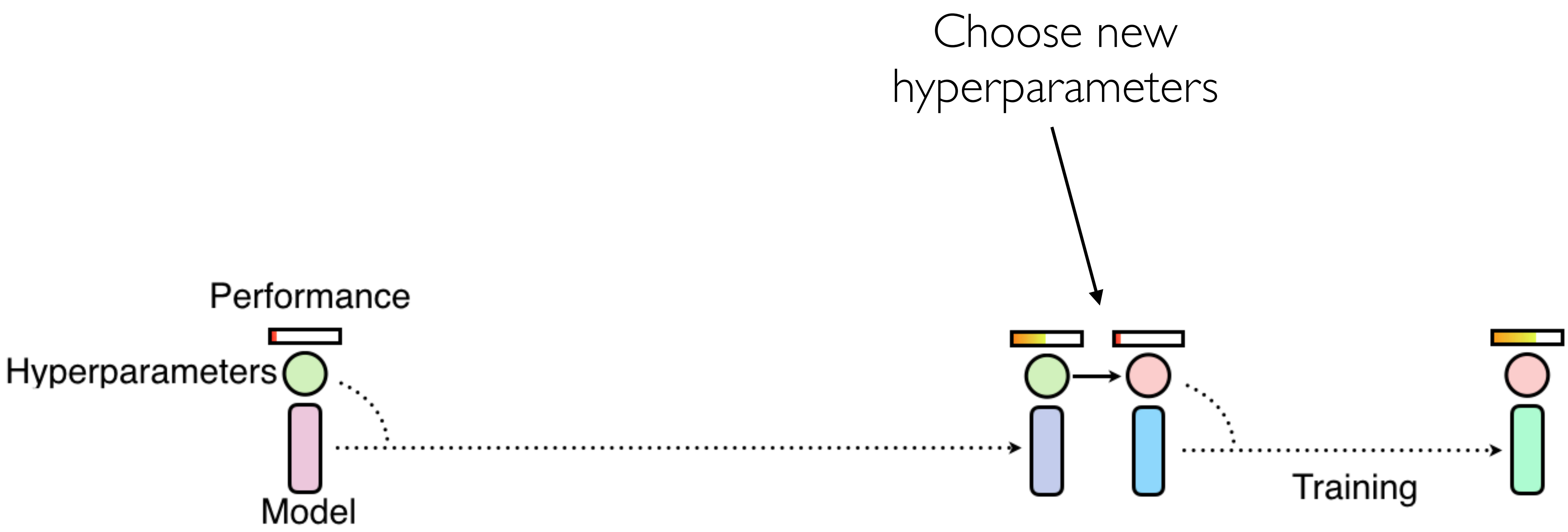
Sequential Optimisation



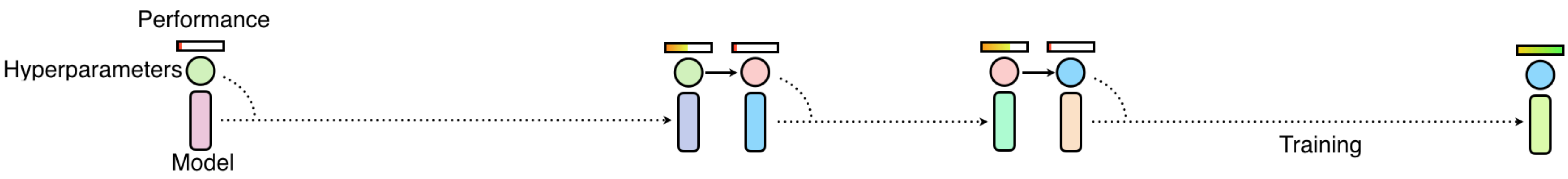
SEQUENTIAL OPTIMISATION



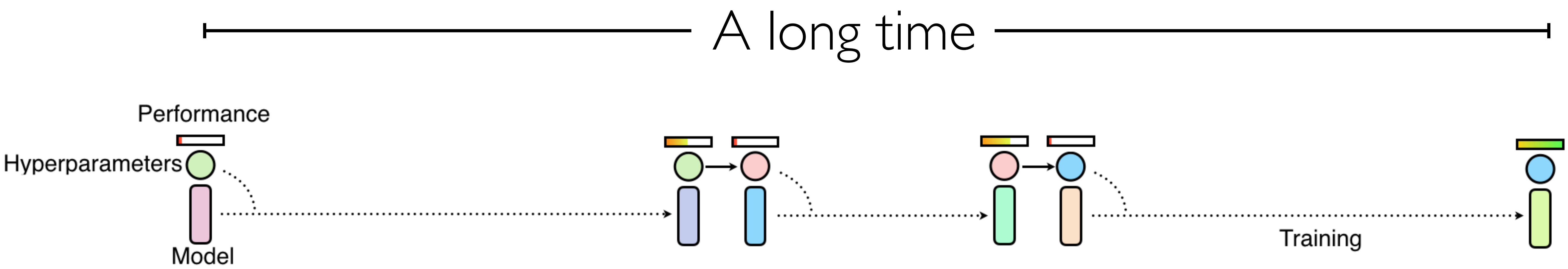
Sequential Optimisation



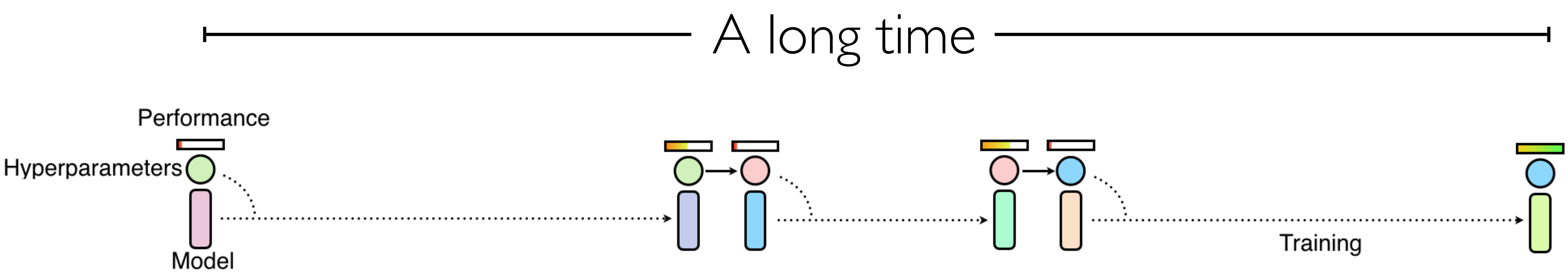
Sequential Optimisation



Sequential Optimisation



SEQUENTIAL OPTIMISATION



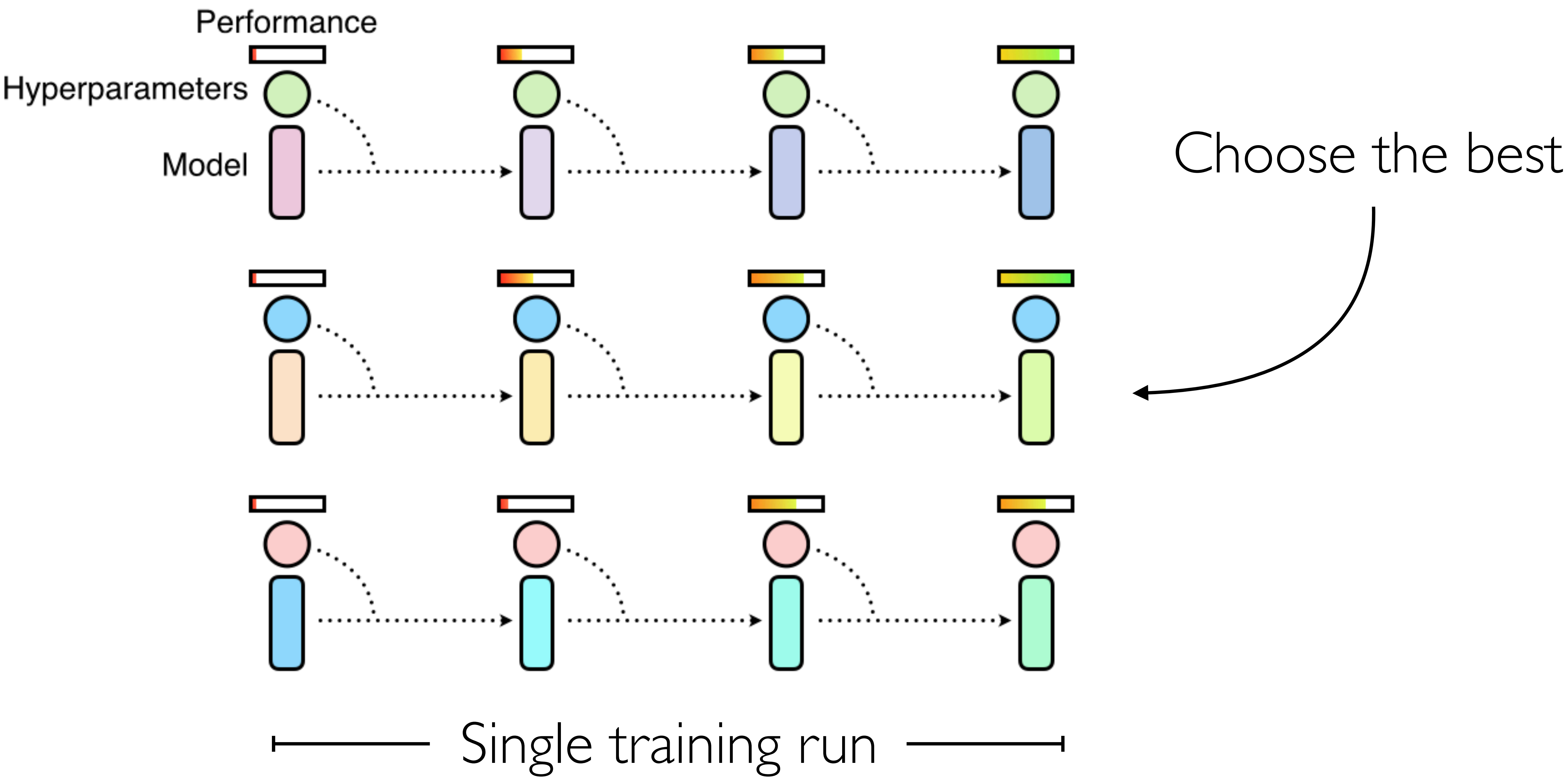
Automate with Bayesian optimisation:

GP-UCB [Srinivas '09], TPE [Bergstra '11], Spearmint [Snoek '12], SMAC [Hutter '11]
Speed up process [Gyorgy '11, Agarwal '11, Sabharwal '16, Swersky '13, Swersky '14, Domhan '15, Klein '16, Snoek '15, Springenberg '16] or use parallel bandits [Li '16].
Lead to SOTA performance e.g. language models [Melis '17]

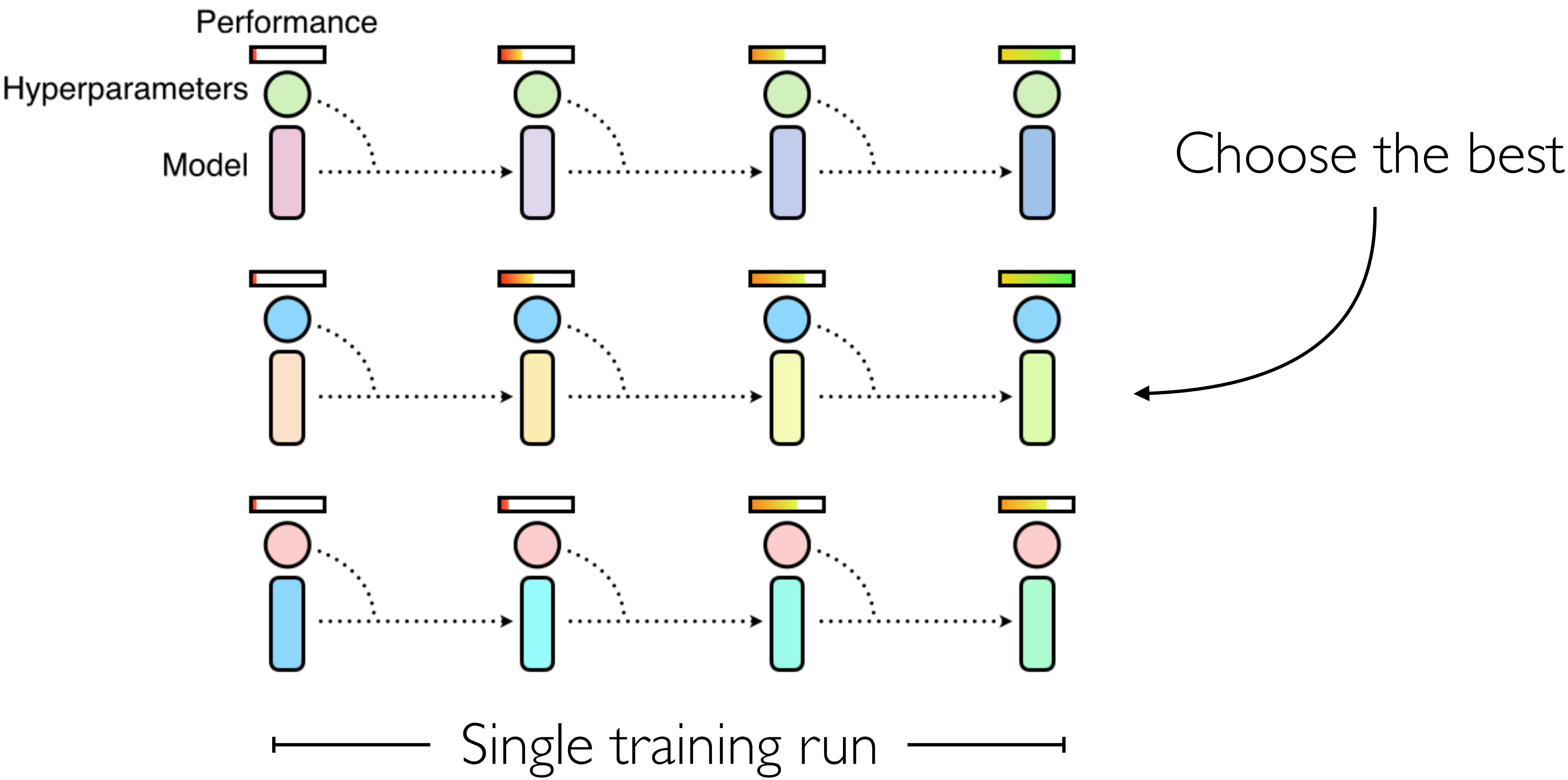
Or genetic algorithms:

[Young '15, Whiteson '06, Miikkulainen '11, Schmidhuber]

Random Search

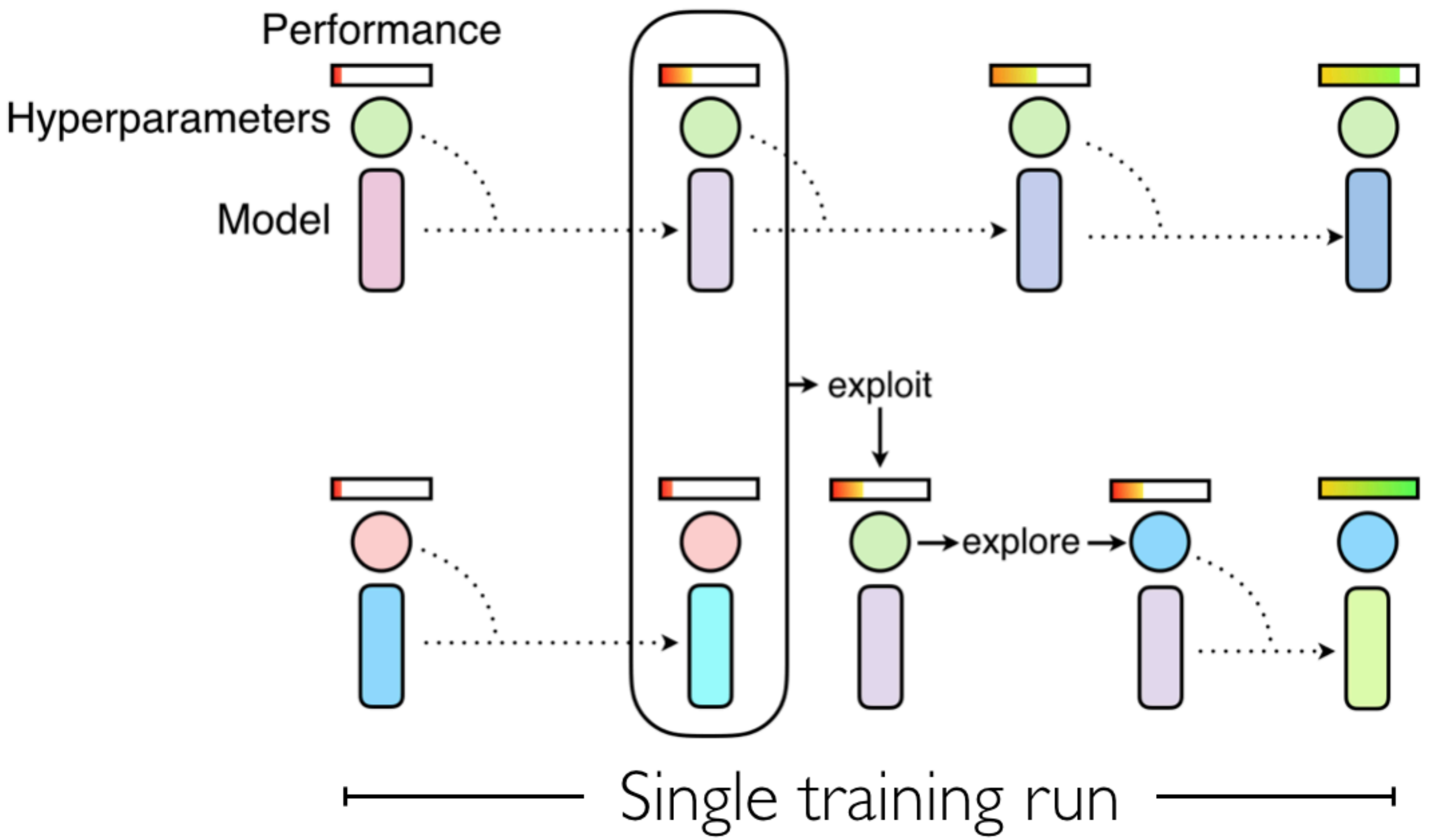


Random Search



Unreasonably effective [Bergstra '12].
Easy to parallelise.
Wastes computation on easily identifiable bad hyperparameters.
Still limits to fixed hyperparameters for all of training.

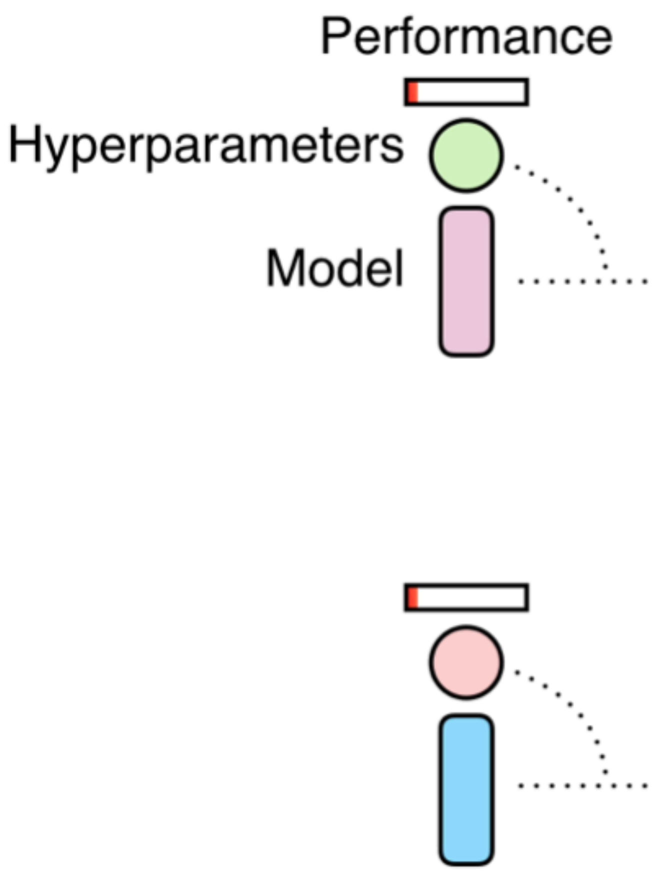
POPULATION BASED TRAINING (PBT)



Start with random search.
Allow workers to share information.
Workers can **exploit** for model selection, and **explore** new hyperparameters.

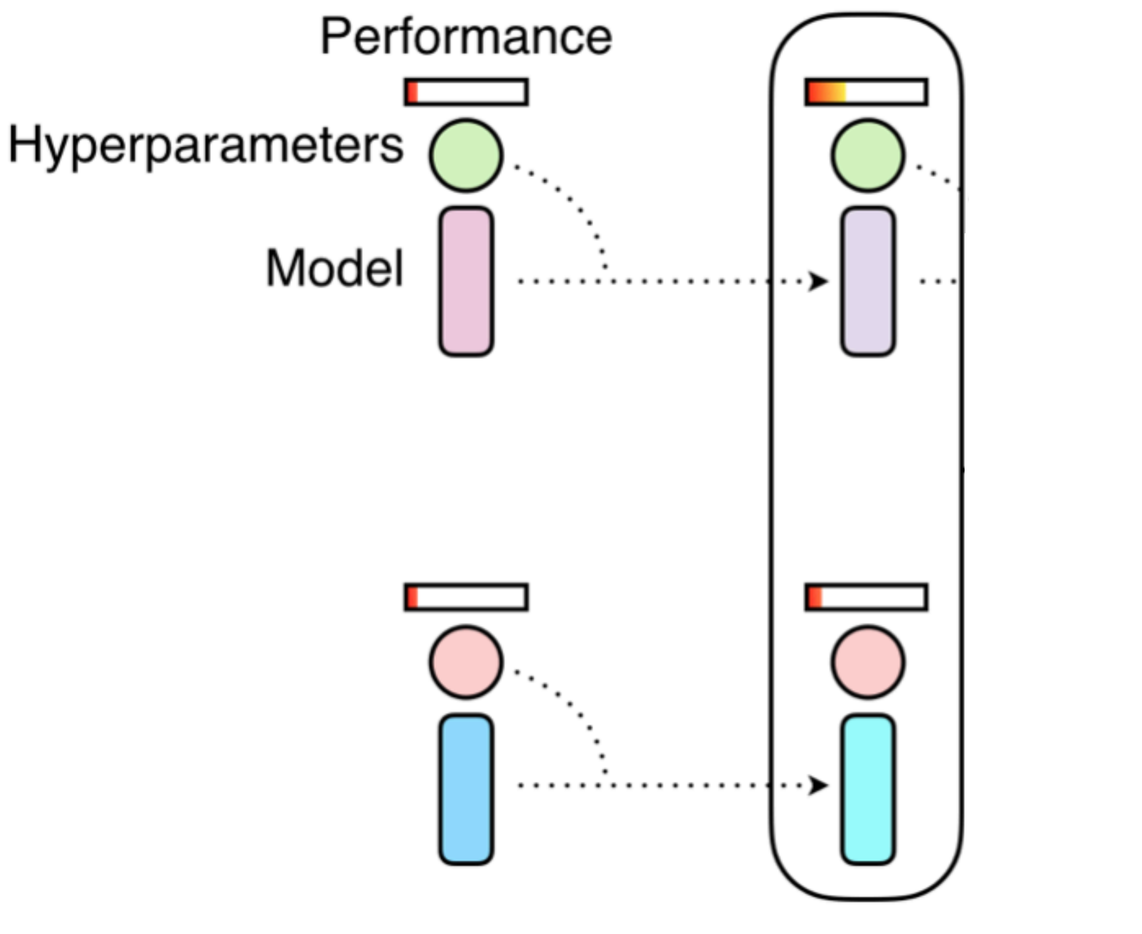
Genetic algorithm acting on a **timescale which allows gradient based learning.**

Population Based Training (PBT)



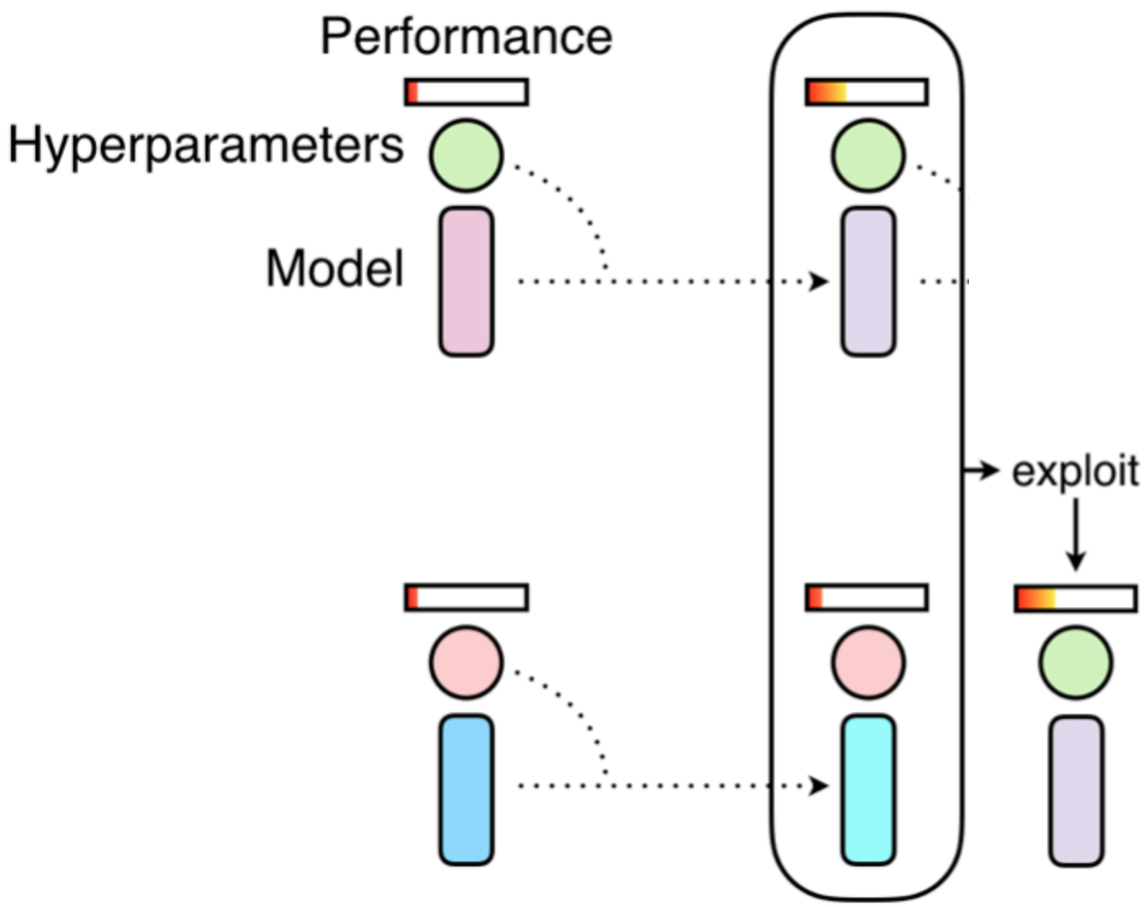
Start with random search.
Randomly initialise model weights.
Randomly initialise hyperparameters from a prior distribution

Population Based Training (PBT)



Allow training for enough steps for learning to occur.

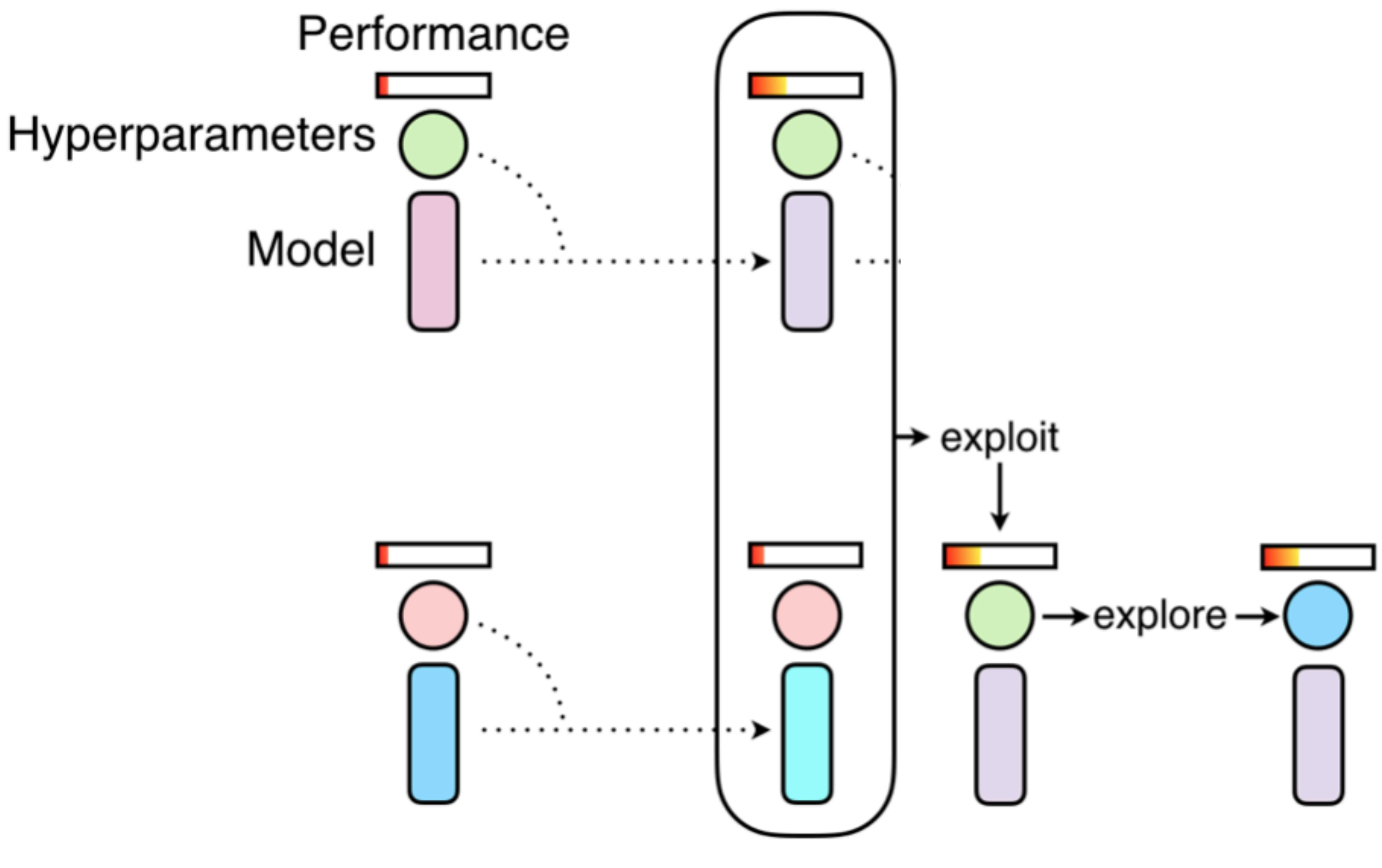
POPULATION BASED TRAINING (PBT)



Exploit: each worker compares its performance to the population. If bad, then inherit the partial solution from a better worker (e.g. copy the model and hyperparameters).

- Binary tournament — random opponent, better model wins.
- Truncation selection — if in bottom 20% inherit from top 20%.

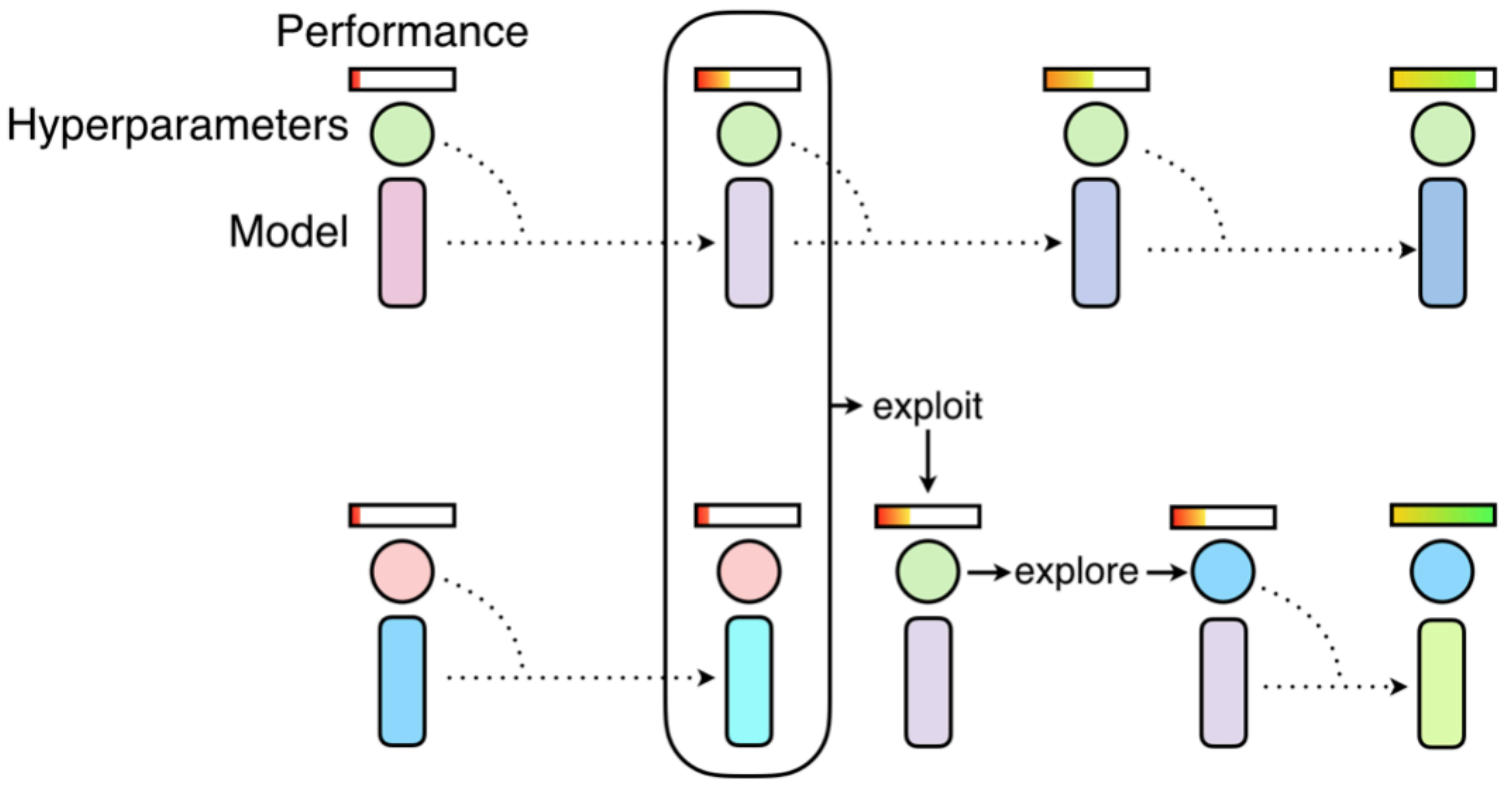
POPULATION BASED TRAINING (PBT)



Explore: mutate the hyperparameters that were inherited to explore potentially better hyperparameters at this point in training. Mutate each hyperparameter independently.

- Perturb current value randomly by factor of e.g. 20%.
- Resample from the initial prior distribution defined.

POPULATION BASED TRAINING (PBT)



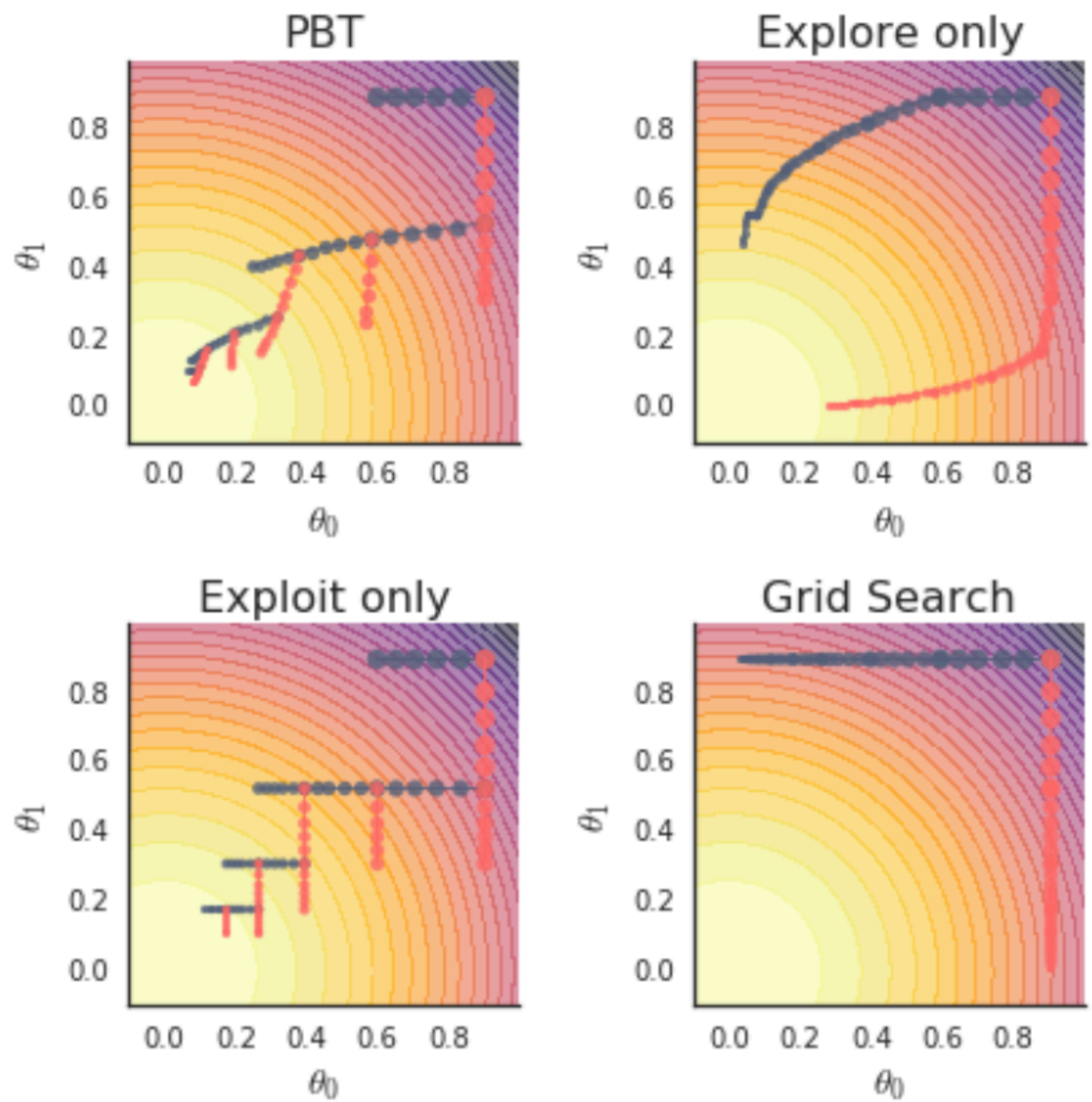
Step: perform steps of regular gradient-based training.

Exploit: if worker is bad, then inherit better partial model.

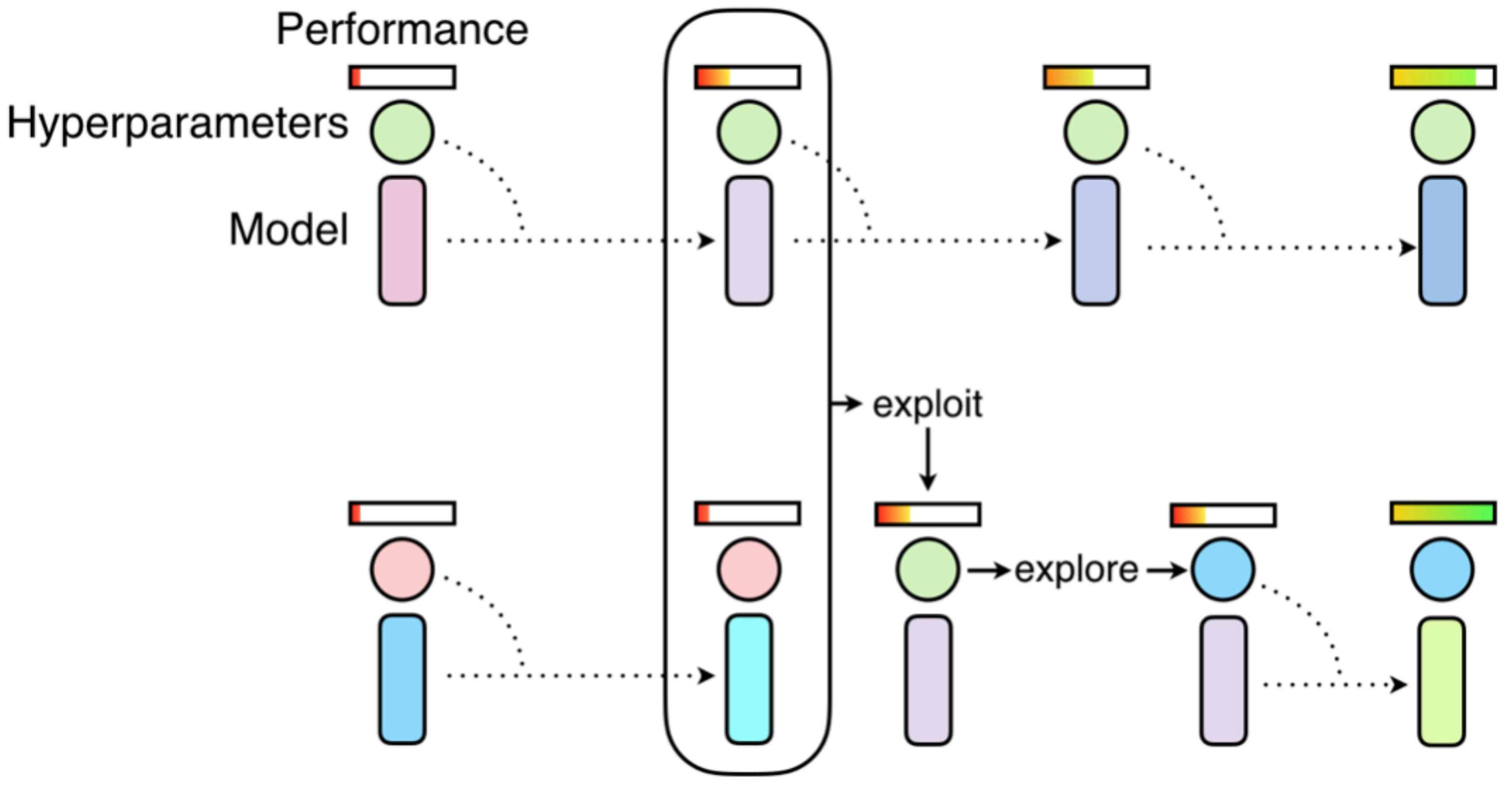
Explore: mutate the hyperparameters that were inherited.

Repeat.

Toy Example


PBT
Explore only
Exploit only
Grid Search
θ_0
θ_1
0.0
0.2
0.4
0.6
0.8

Population Based Training (PBT)



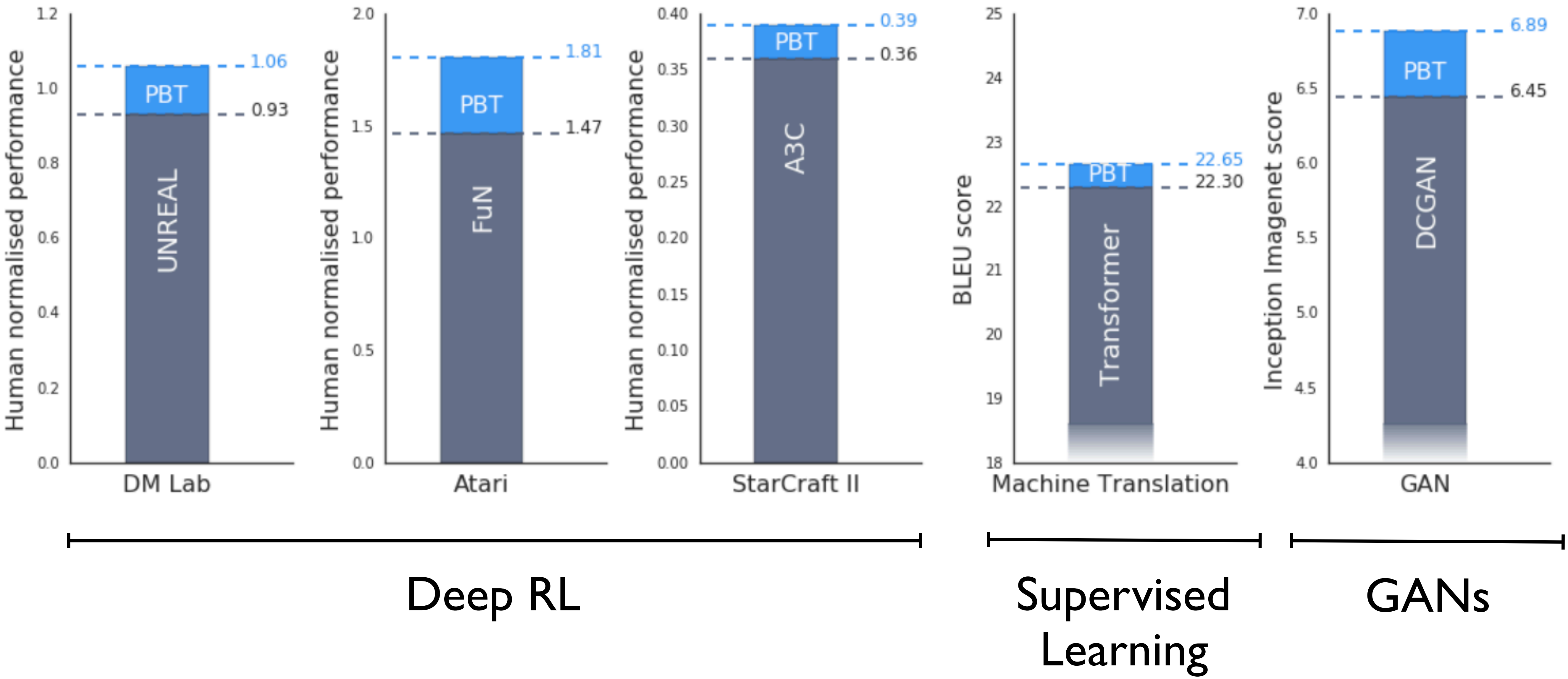
Combines local optimisation with gradients with model selection and hyperparameter refinement. Two-timescale learning system.

Exploit can **optimise for non-differentiable & expensive metrics**. Allows **online adaptation** of hyperparameters.

Asynchronous and very easy to integrate with existing pipelines.

EXPERIMENTS

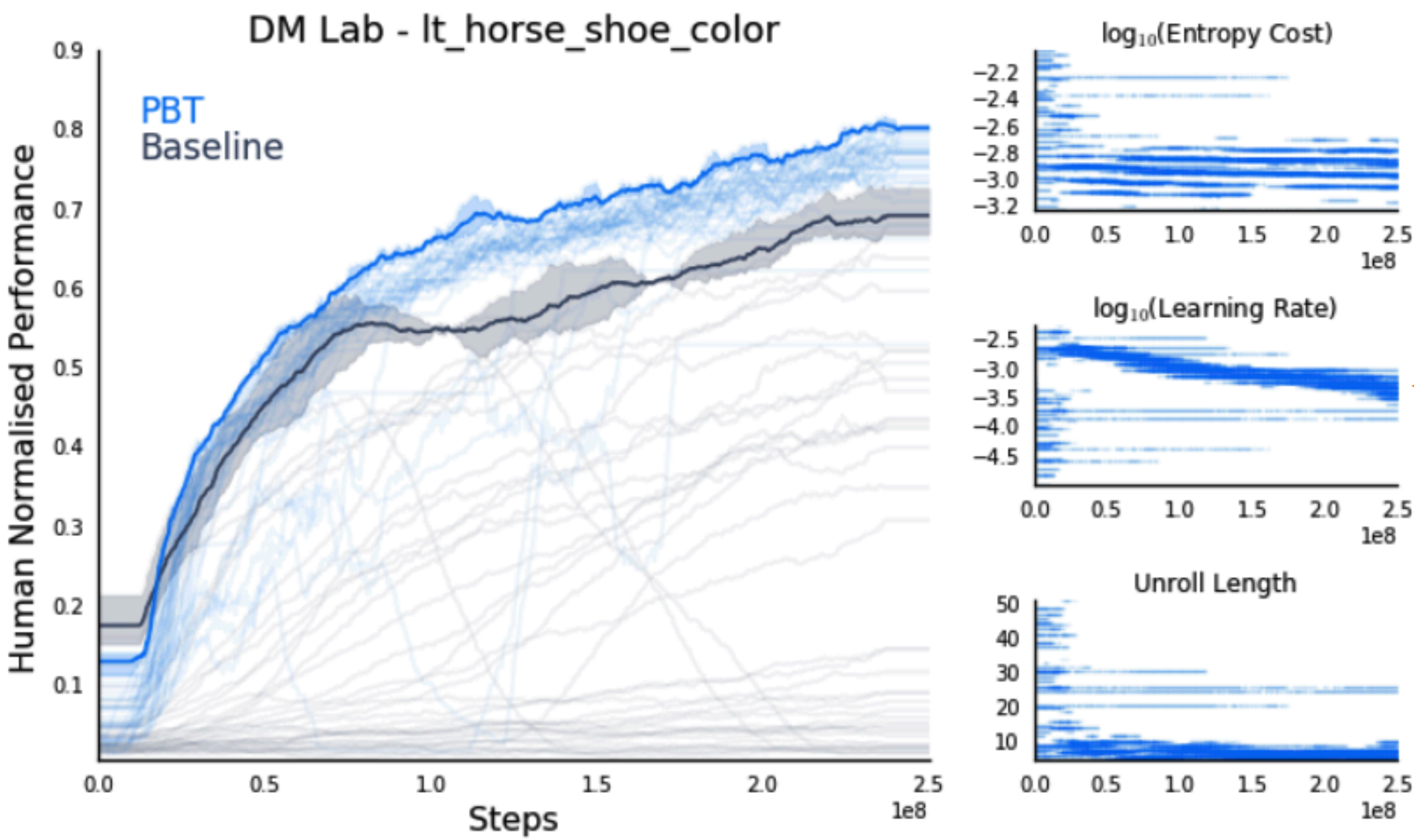
Demonstrate on a range of domains.



Speeds up learning, can use less computational resources, better final performance.

UNREAL ON DM LAB

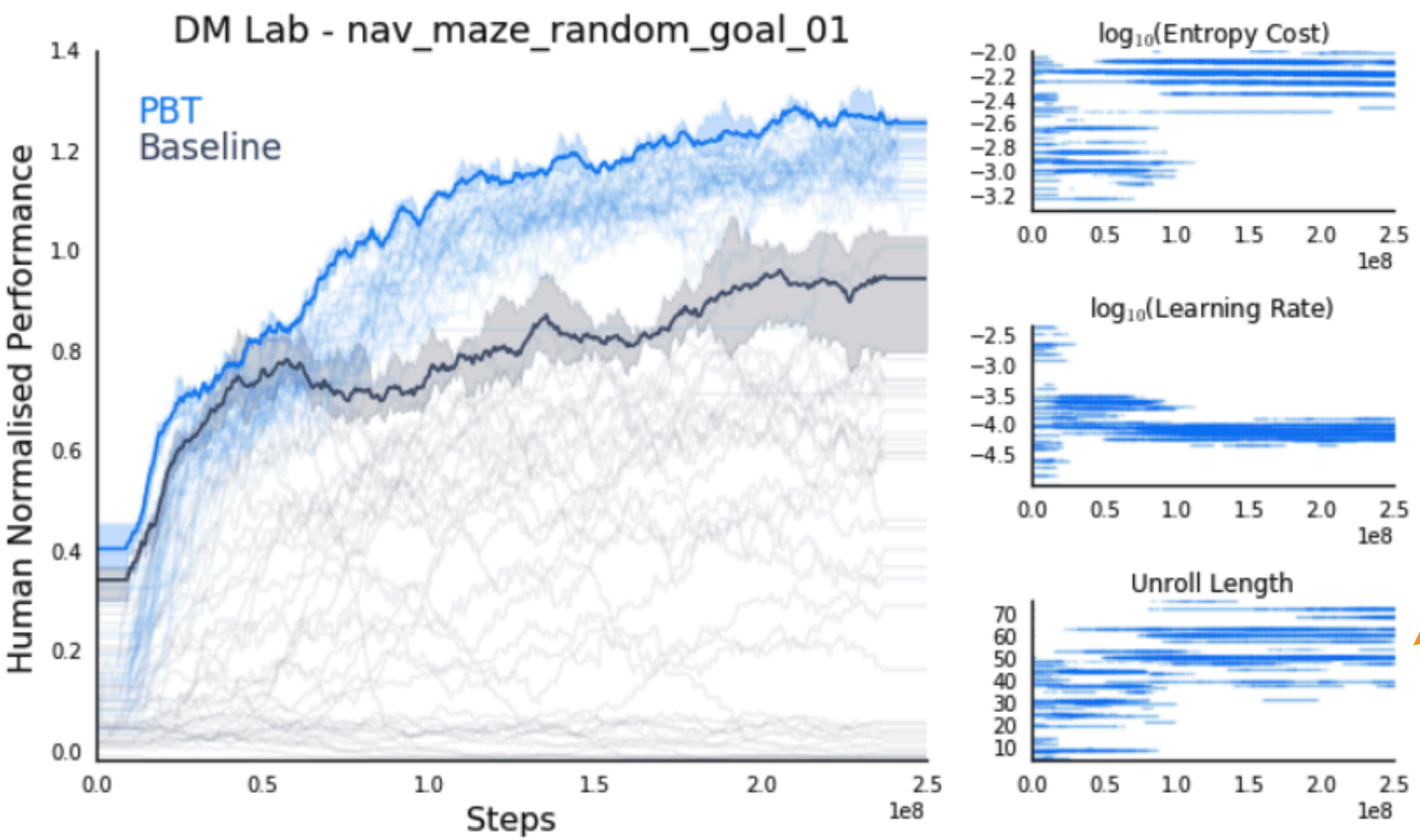
UNREAL [Jaderberg '16] on DeepMind Lab 3D environments.



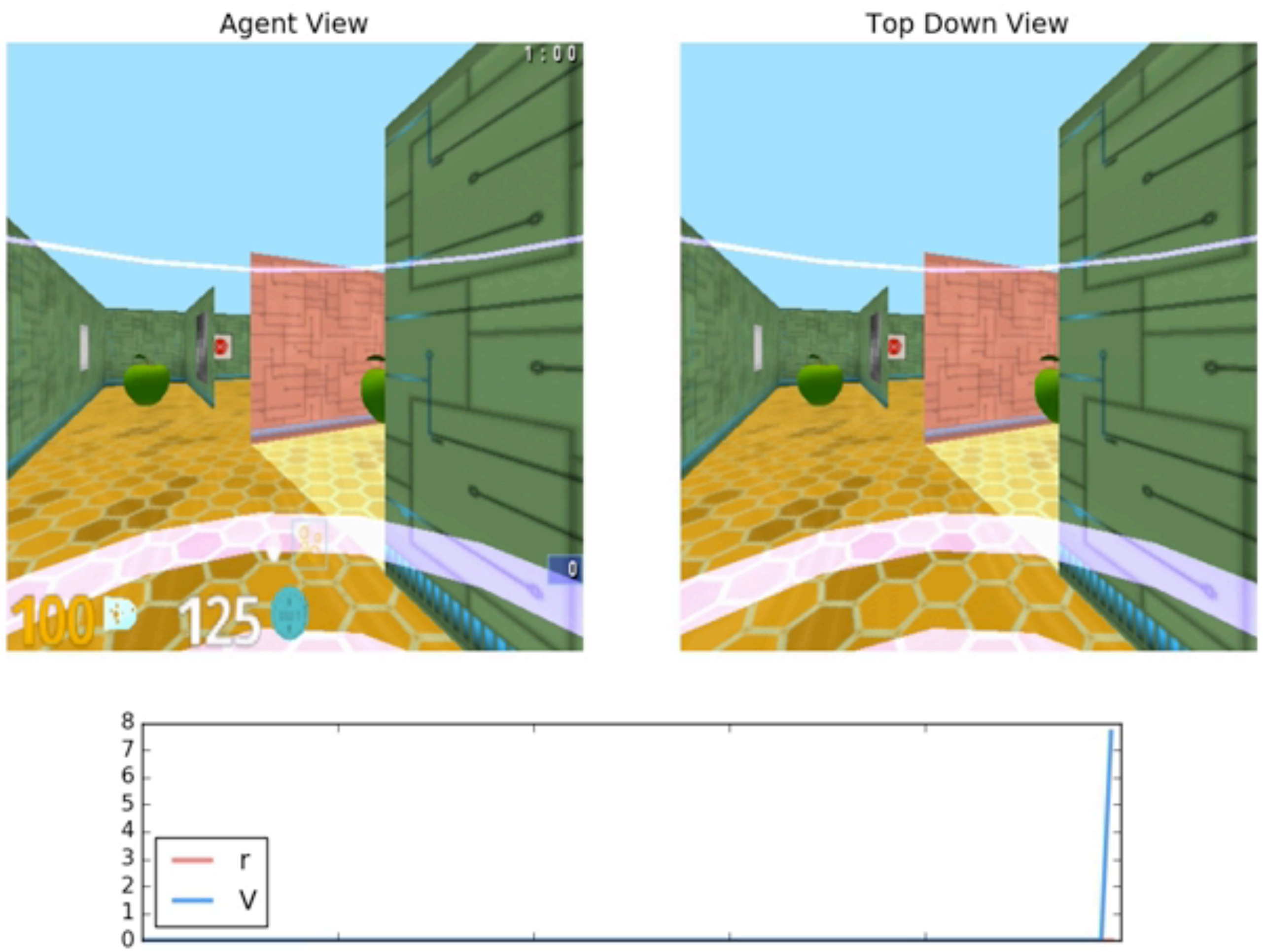
Automatic learning rate decay

UNREAL ON DM LAB

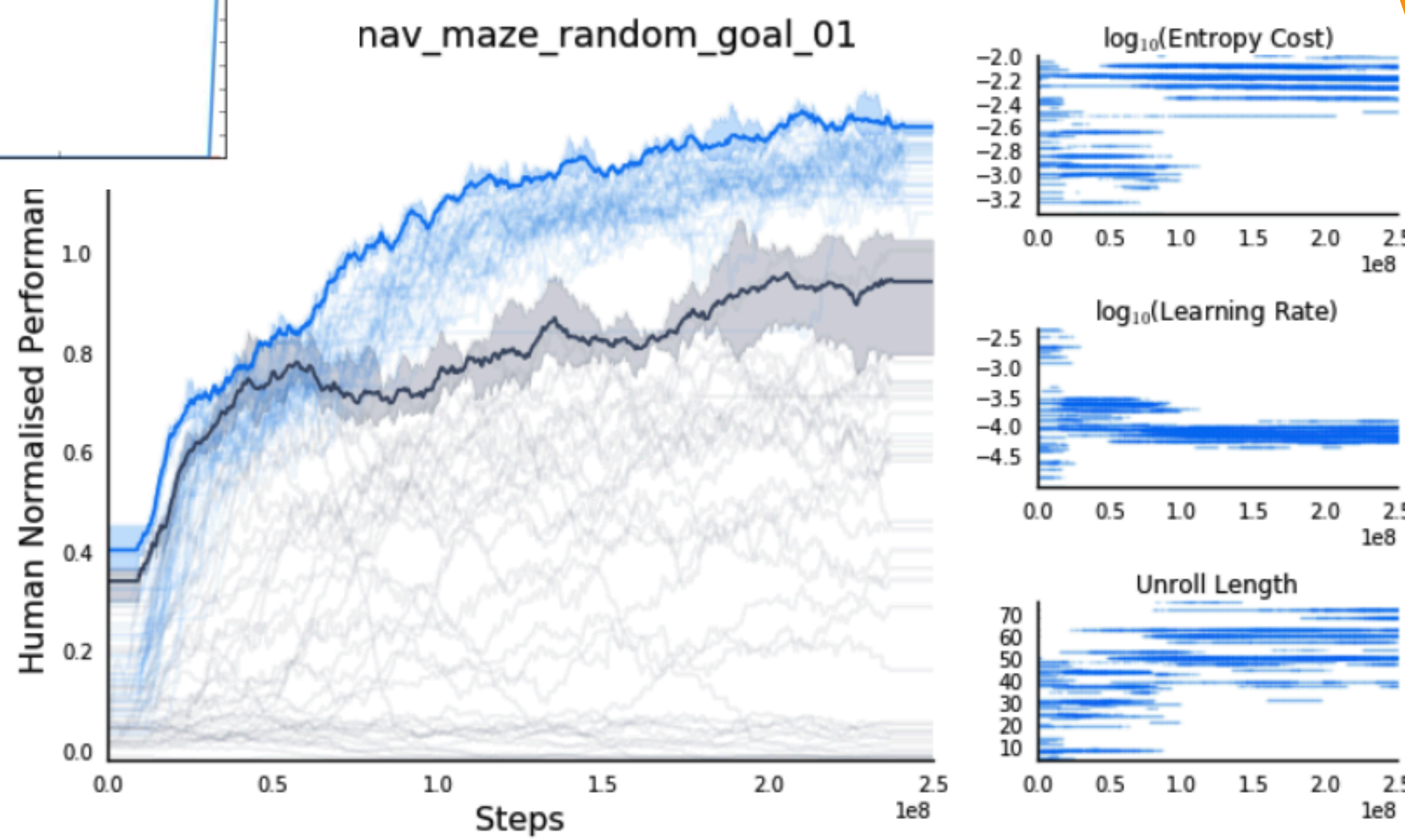
Discovers unroll length outside initial distribution



UNREAL ON DM LAB



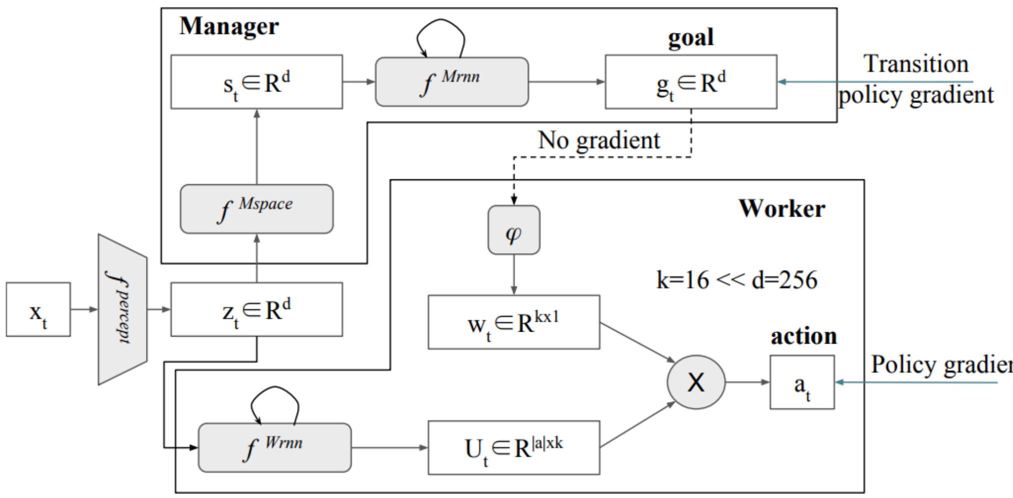
Discovers unroll length outside initial distribution



DeepMind

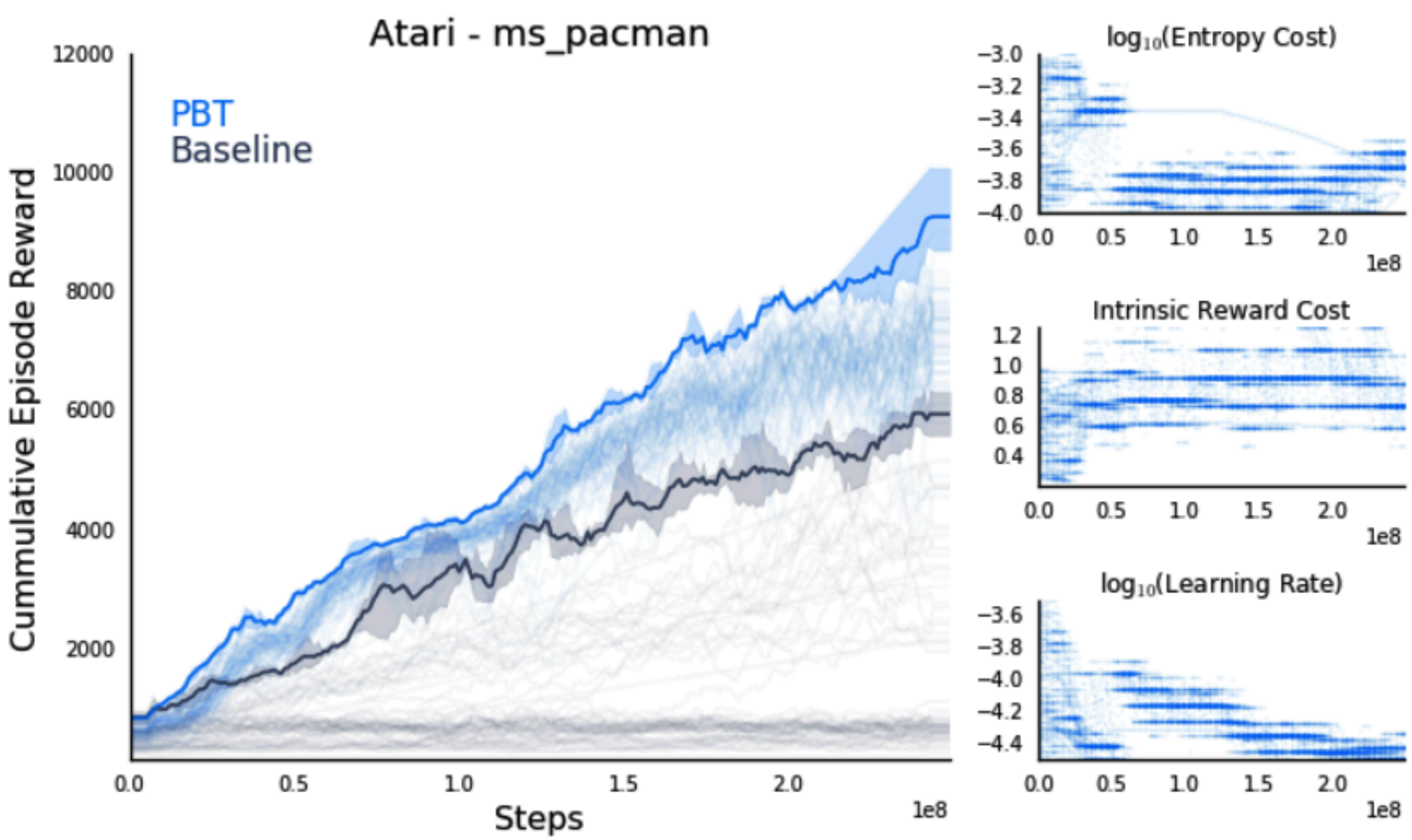
FuN on Atari

Feudal Networks (FuN) [Vezhnevets '16] on Atari environments.



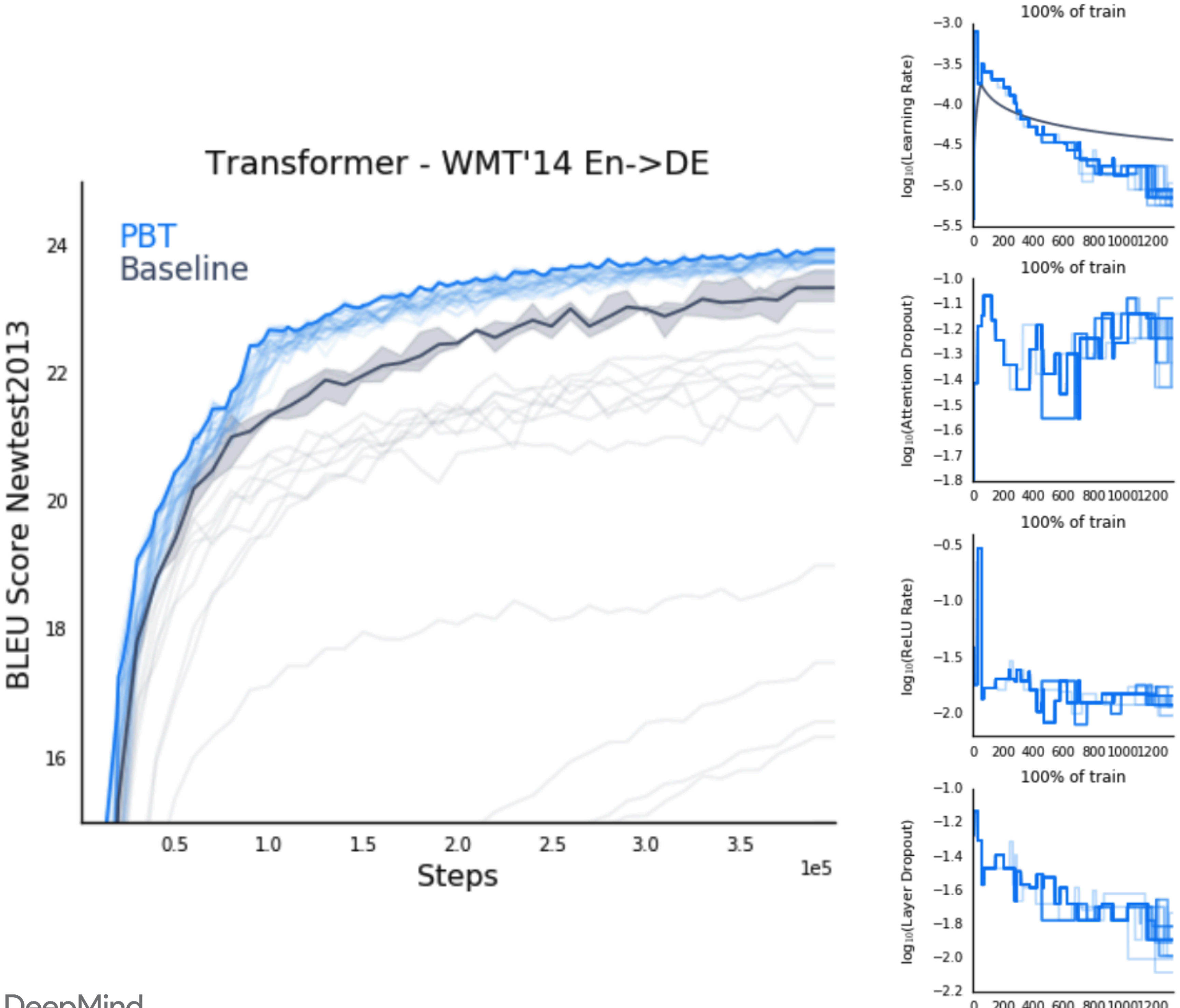
$$R_t + \alpha R_t^I$$

Auto optimise the intrinsic reward



Machine Translation

Transformer Networks [Vaswani '17] for WMT English-German.
Optimise for BLEU score directly.

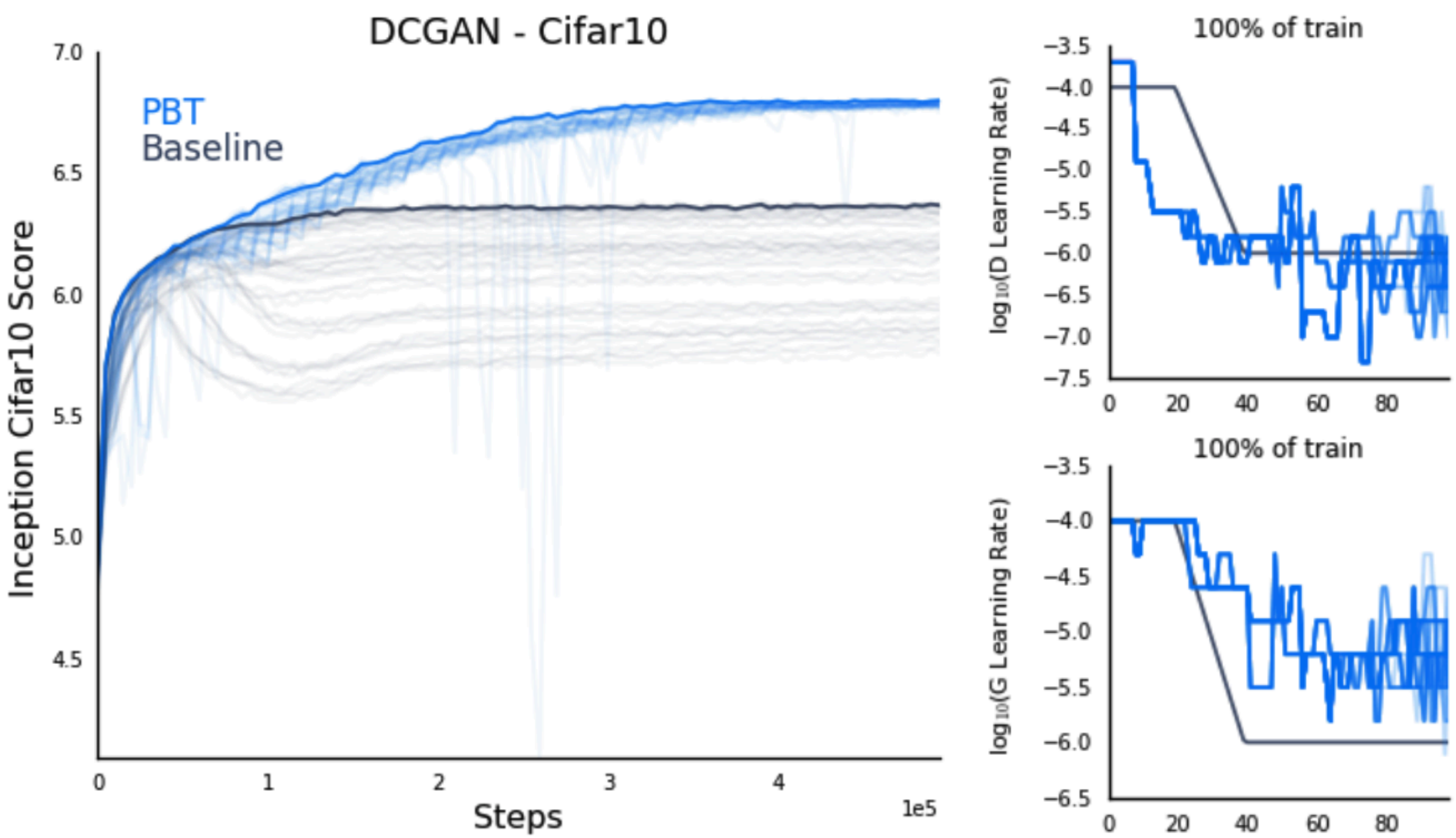


Discovers hand designed learning rate schedule

Generative Adversarial Networks

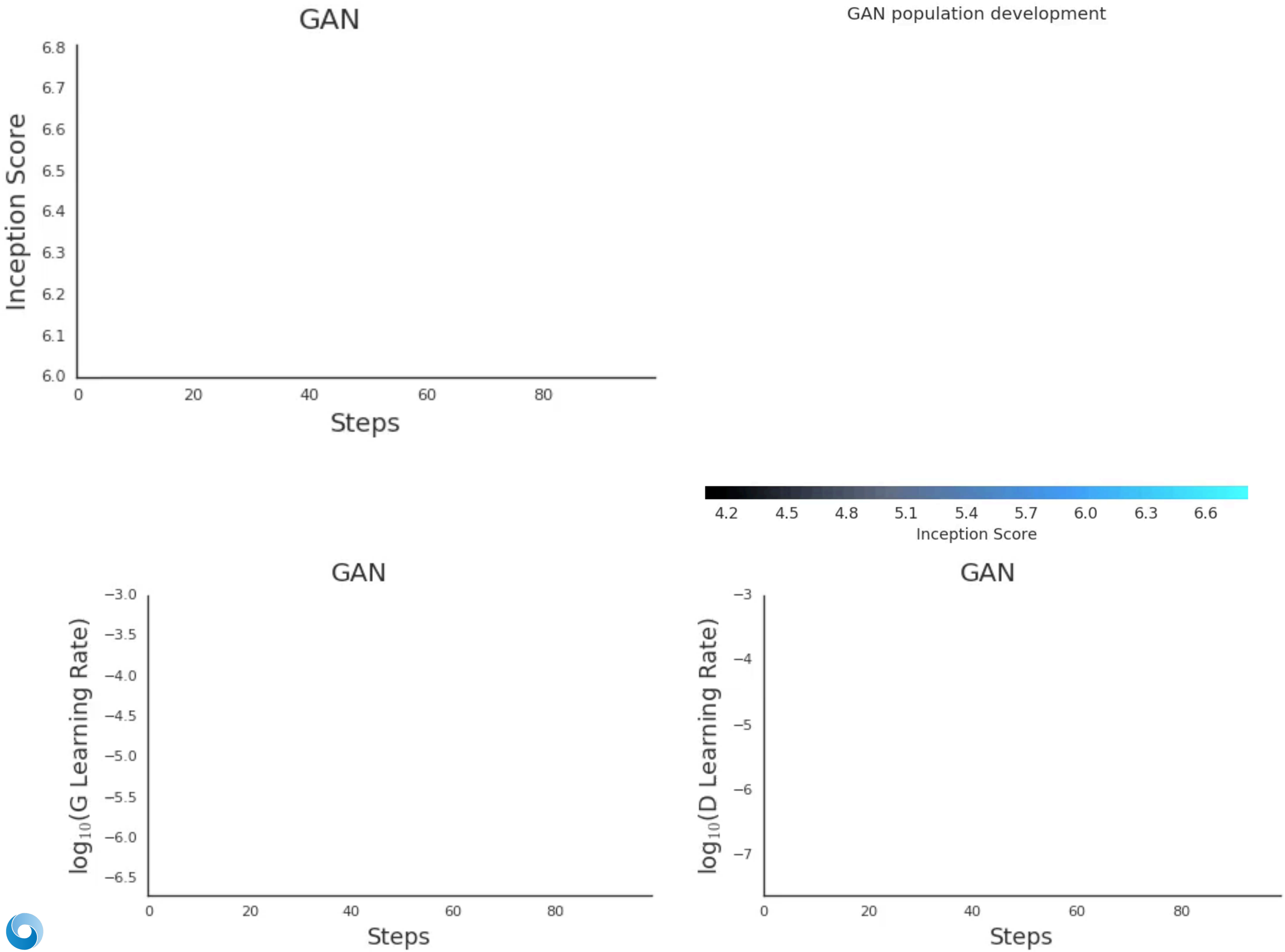
DCGAN architecture [Radford '16] on CIFAR-10.
Optimise for Inception score directly.

Discriminator LR annealed aggressively

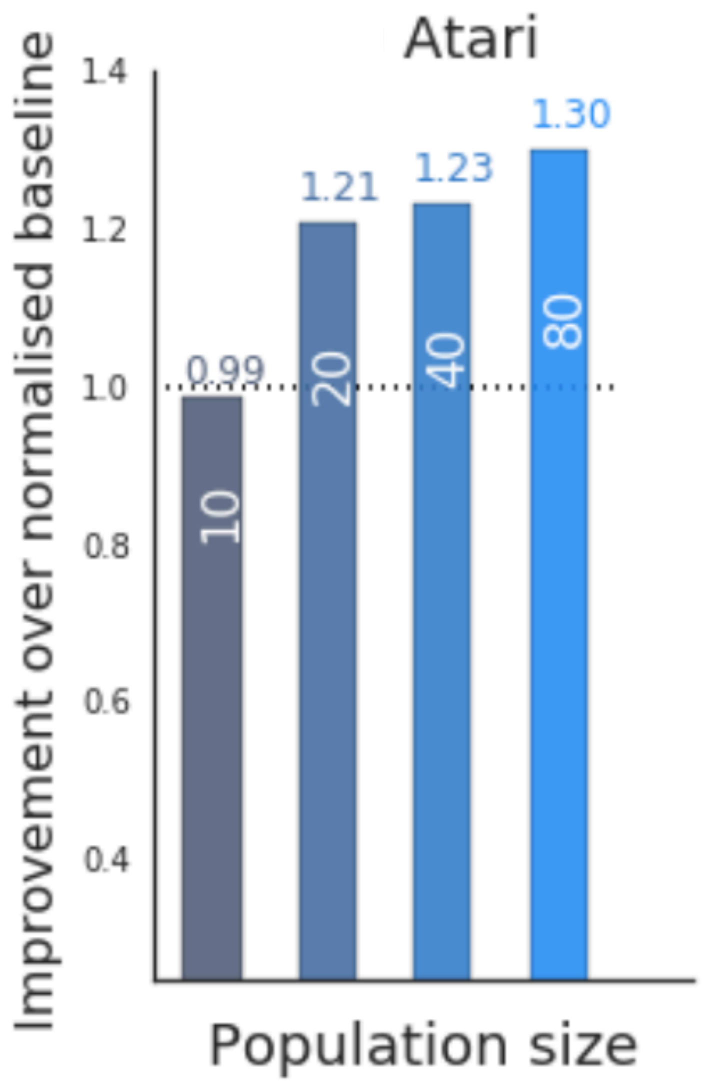


Generator LR annealed slower

GAN population development

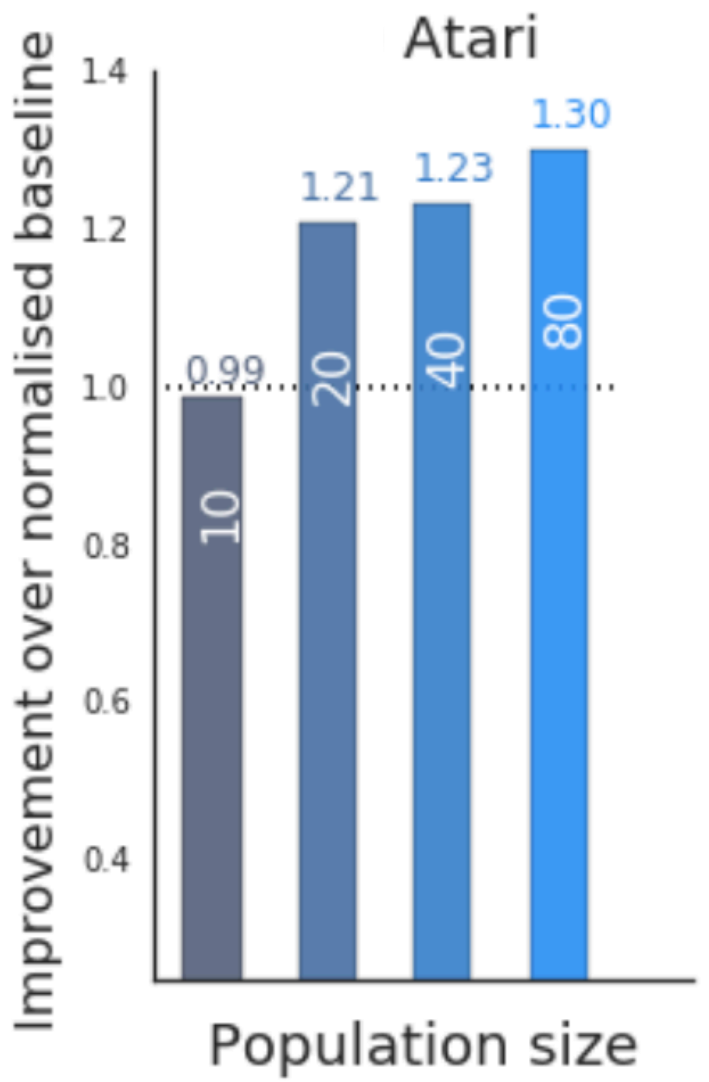

GAN
Inception Score
6.8
6.7
6.6
6.5
6.4
6.3
6.2
6.1
6.0
0
20
40
60
80
Steps
GAN population development
4.2
4.5
4.8
5.1
5.4
5.7
6.0
6.3
6.6
Inception Score
GAN
log10(G Learning Rate)
−3.0
−3.5
−4.0
−4.5
−5.0
−5.5
−6.0
−6.5
0
20
40
60
80
Steps
GAN
log10(D Learning Rate)
−3
−4
−5
−6
−7
0
20
40
60
80
Steps

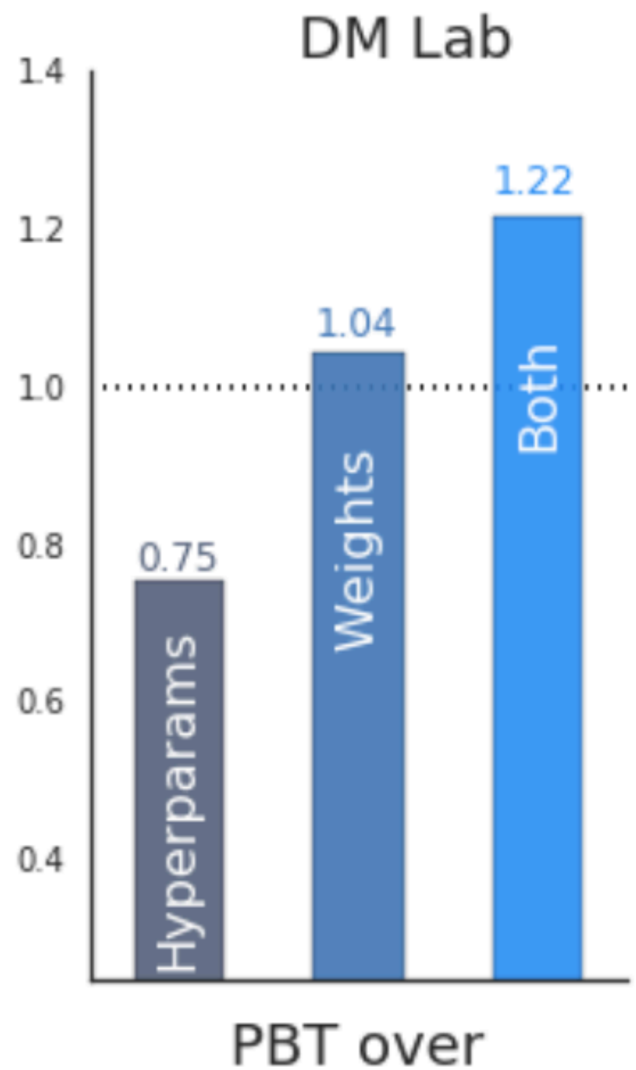
Algorithm Analysis



Smaller population size means higher variance results due to greedy algorithm.

ALGORITHM ANALYSIS

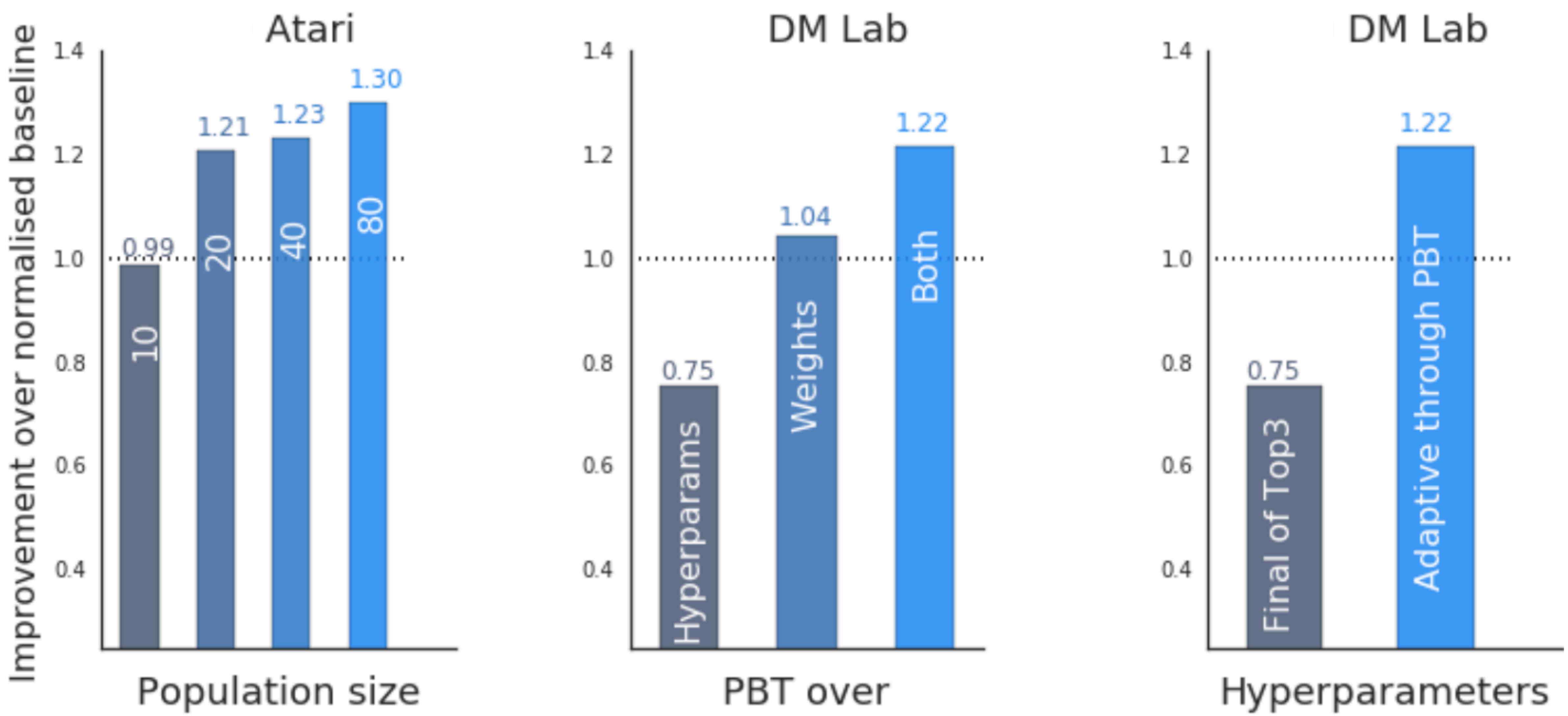




Smaller population size means higher variance results due to greedy algorithm.

PBT on weights and hyperparameters are crucial to best performance.

ALGORITHM ANALYSIS

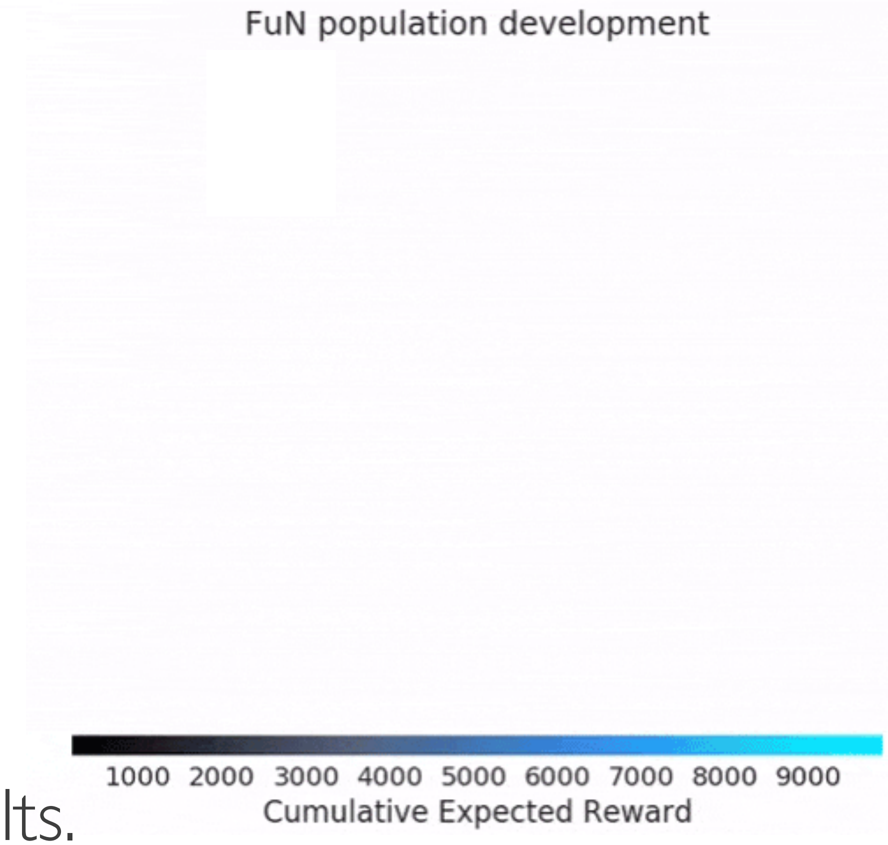


Smaller population size means higher variance results due to greedy algorithm.

PBT on weights and hyperparameters are crucial to best performance.

Adaptation of hyperparameters better than using best found hyperparameters

CONCLUSIONS



Algorithm for joint optimisation of model and hyperparameters

- Online adaptation of hyperparameters.
- Model selection by weight inheritance.
- Easy to integrate with existing training code.

Enhances training across many domains

- Improves performance of final models found.
- Does not change the wall clock time for final results.
- Can reduce the computational resources required.
- Good for new unfamiliar models.
- Adapts to non-stationary training problems.
- Optimise indirect performance metrics.

Future work with PBT

- Better exploit the population in non-greedy way.
- Better explore in hyperparameter space, e.g. online modelling, crossover.

Questions?