Population Based Training of Neural Networks (PBT)

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

(DeepMind, London, UK)
2017

Alexander Frost for R244

Presentation Structure

- Overview of PBT
- Problem context
- Solution proposed
- Comparison with existing work
- Conclusion and discussion

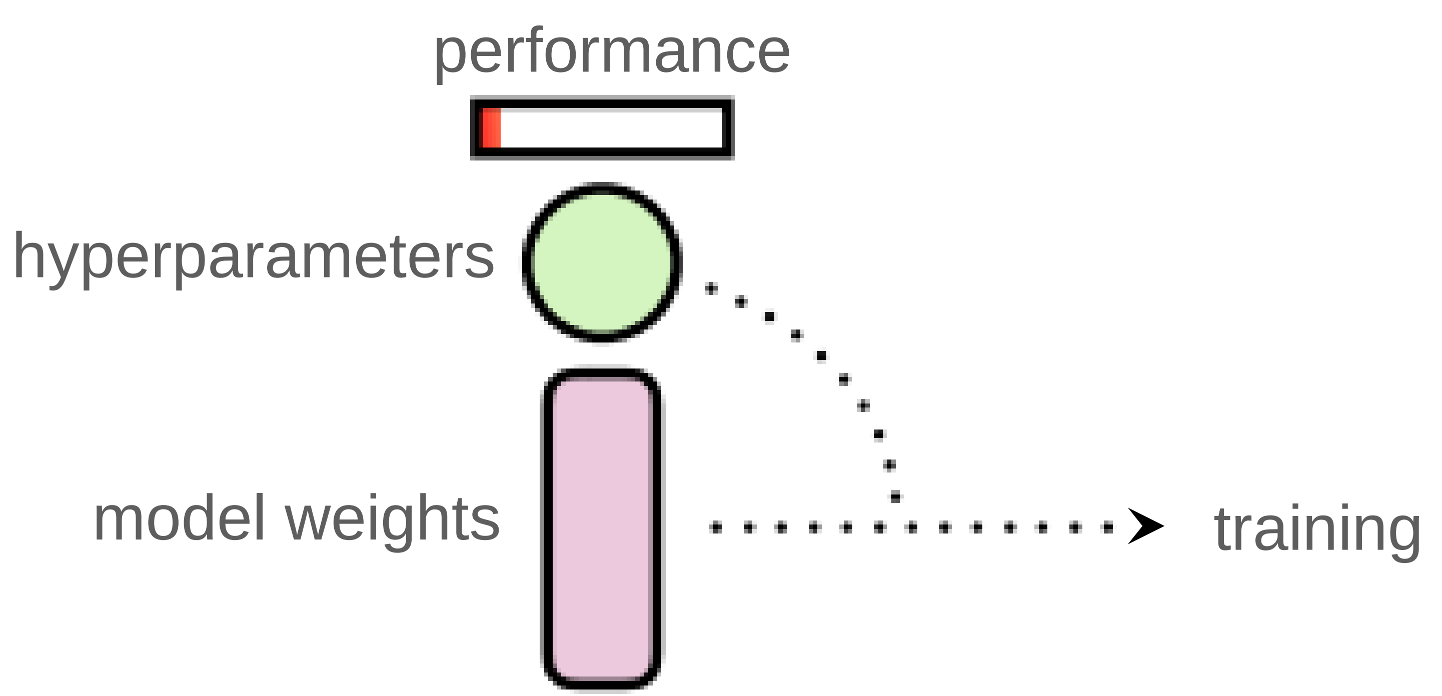
Overview of PBT

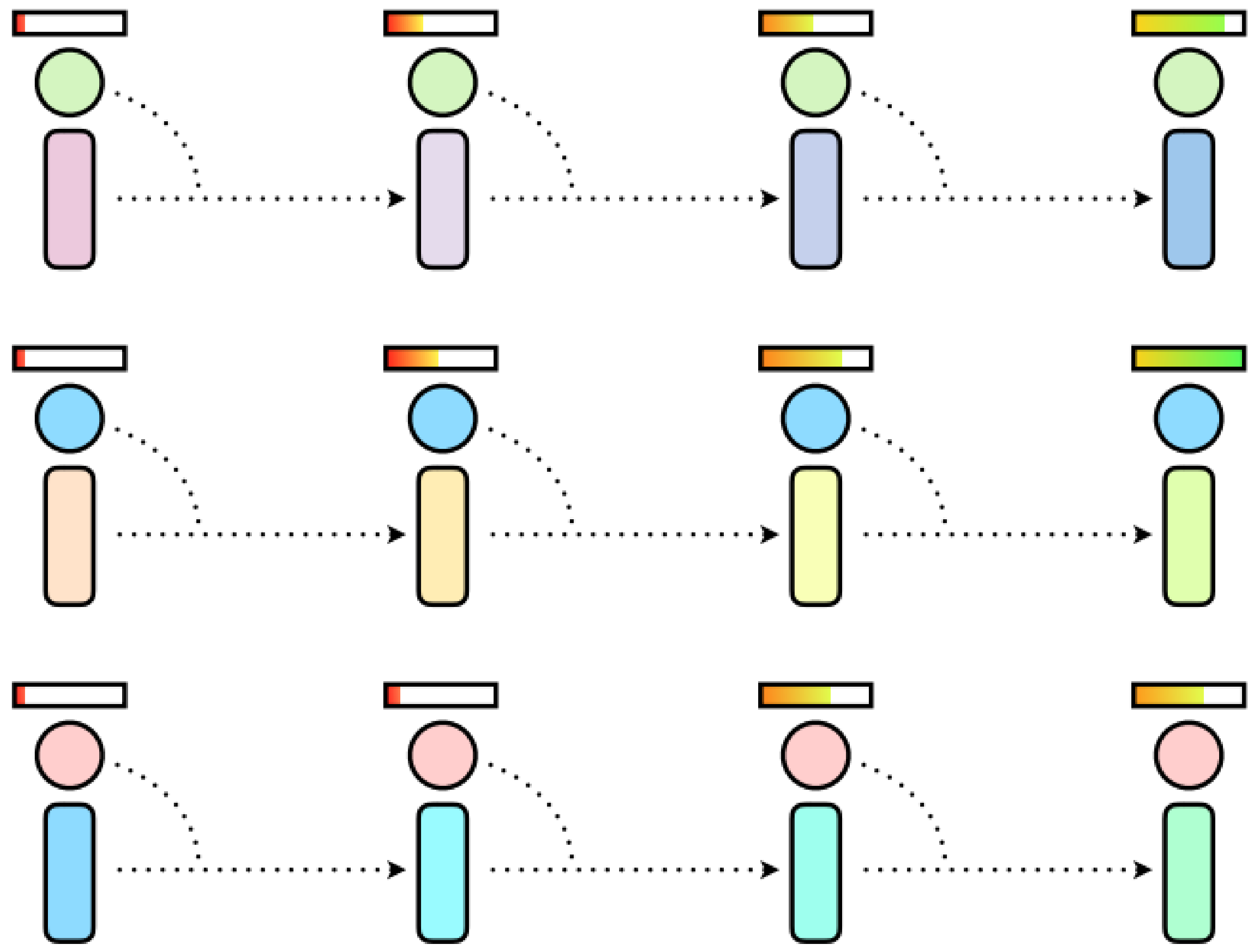
- Algorithm for optimisation of neural network hyperparameters
- Naturally inspired approach based on metal annealing – note PBT is <u>not</u> a genetic algorithm!
- Dynamically assigns computational resources to most promising solutions
- Uses exploration + exploitation mechanism
- Hybridisation of parallel and sequential methods

Problem Context

Parallel optimisation

- E.g. grid search. Allows multiple models to be evaluated with little manual intervention
- Can cover the full solution space reasonably effectively
- Scope for significant speedup IF we assume hyperparameters independent from one another – not always wise…
- Assumes uniform prior over hyperparameters
- Often time exploring poor areas





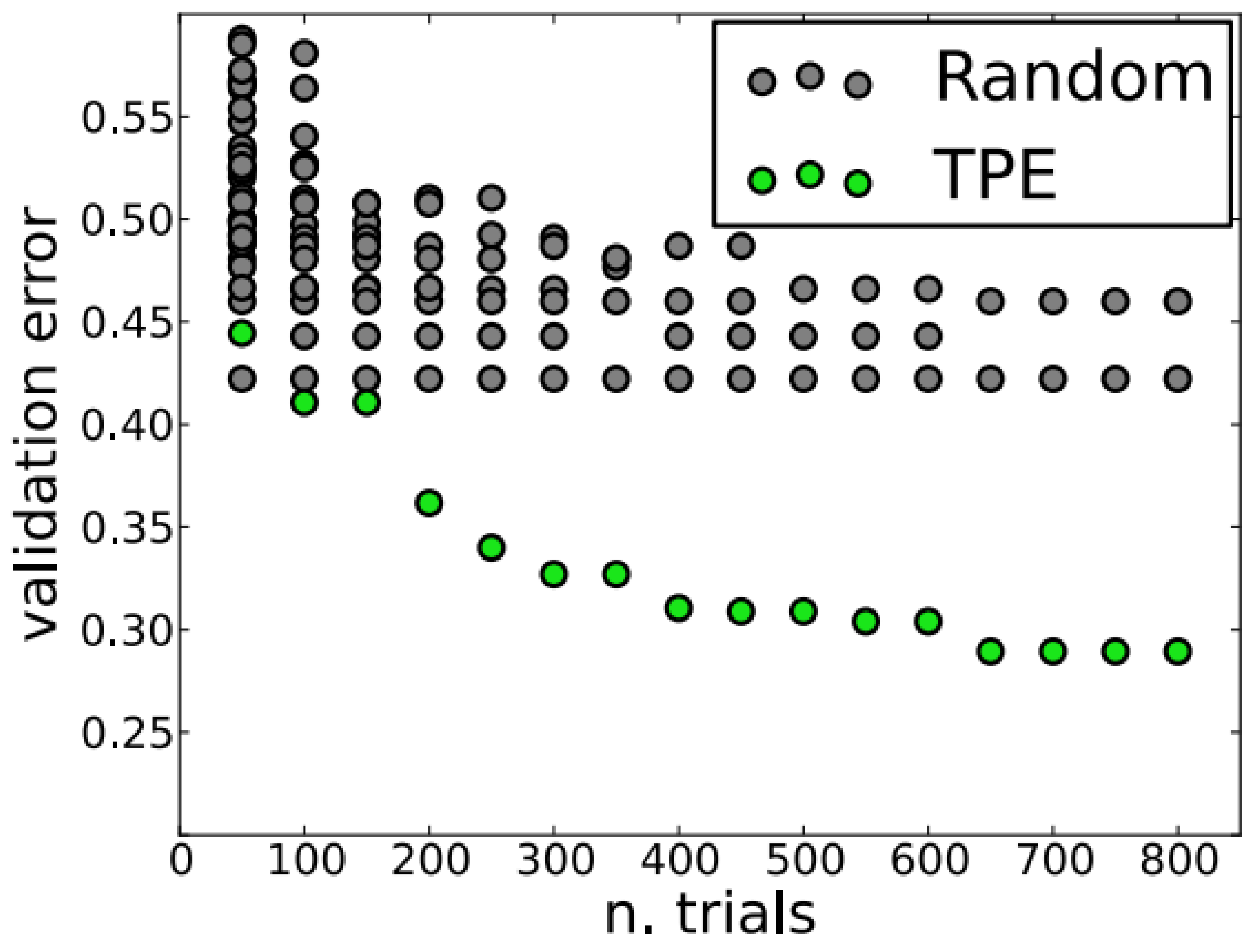
Problem context

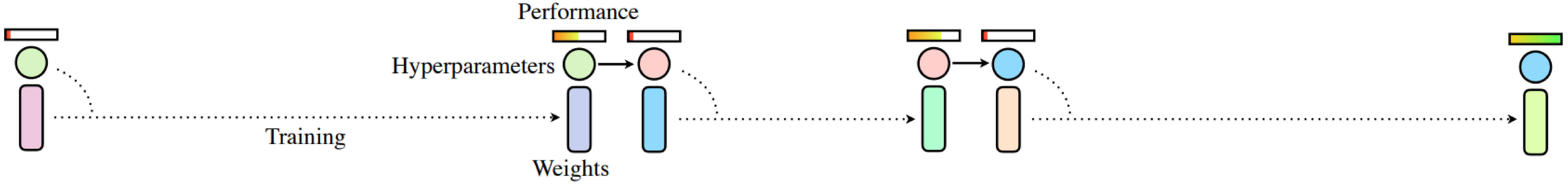
Sequential optimisation

$$p(h|y) = \frac{p(h) \times p(y|h)}{p(y)}$$

- Example: hand-tuning after each run
- Example: Bayesian optimisation
- We choose (sample) hyperparameters based on a priori assumptions, as well as what we learn from running the network each time – minimises evaluations

random vs tree parzen estimators





Problem context

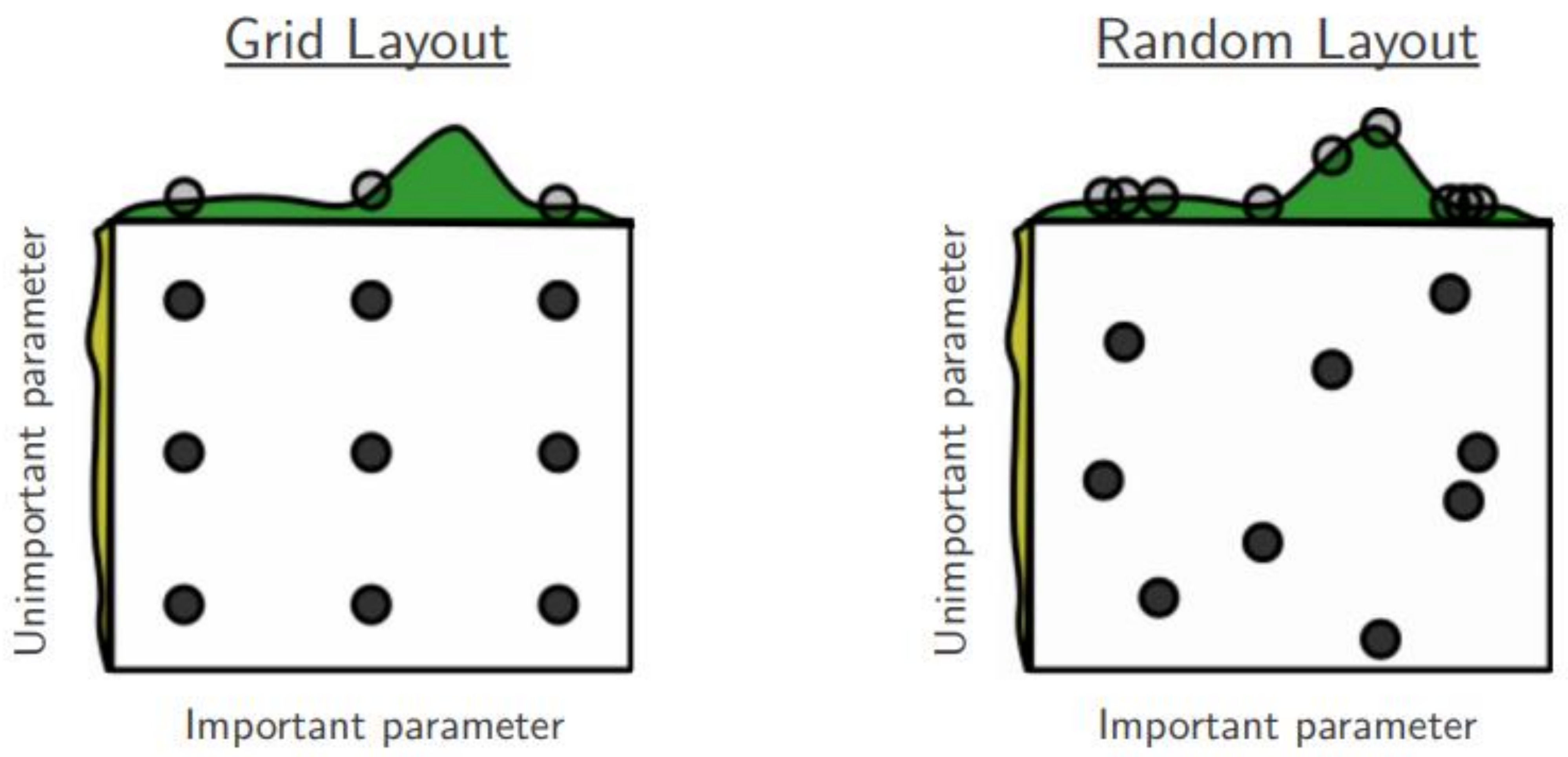
The “unreasonable success” of random search

- Random search generally much faster than grid search (more likely to modify important parameters – most hyperparameters have little effect on outcome)

- BUT still sampling from a uniform prior

- No ‘honing in’ on optimal solutions

- How to combine computational benefits of parallelisation, but still leverage knowledge gained from each (expensive) solution evaluation?



Solution proposed

Population based training

- Randomly initialise population of candidate solutions
- Evaluate solutions asynchronously for *a while* (# evaluations? Threshold?)
- When a solution is ‘ready’, use knowledge from population to decide whether to persist, or tack off and try a more promising alternative
- Important: no need for global synchronisation. Just copy more promising solution (+ some noise) and start from there

Solution proposed

Proposed 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**
      $\theta \leftarrow \mathtt{step}(\theta | h)$ ▷ one step of optimisation using hyperparameters h
      $p \leftarrow \mathtt{eval}(\theta)$ ▷ current model evaluation
      if $\mathtt{ready}(p, t, \mathcal{P})$ **then**
        $h', \theta' \leftarrow \mathtt{exploit}(h, \theta, p, \mathcal{P})$ ▷ use the rest of population to find better solution
        if $\theta \neq \theta'$ **then**
          $h, \theta \leftarrow \mathtt{explore}(h', \theta', \mathcal{P})$ ▷ produce new hyperparameters h
          $p \leftarrow \mathtt{eval}(\theta)$ ▷ 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

Solution proposed

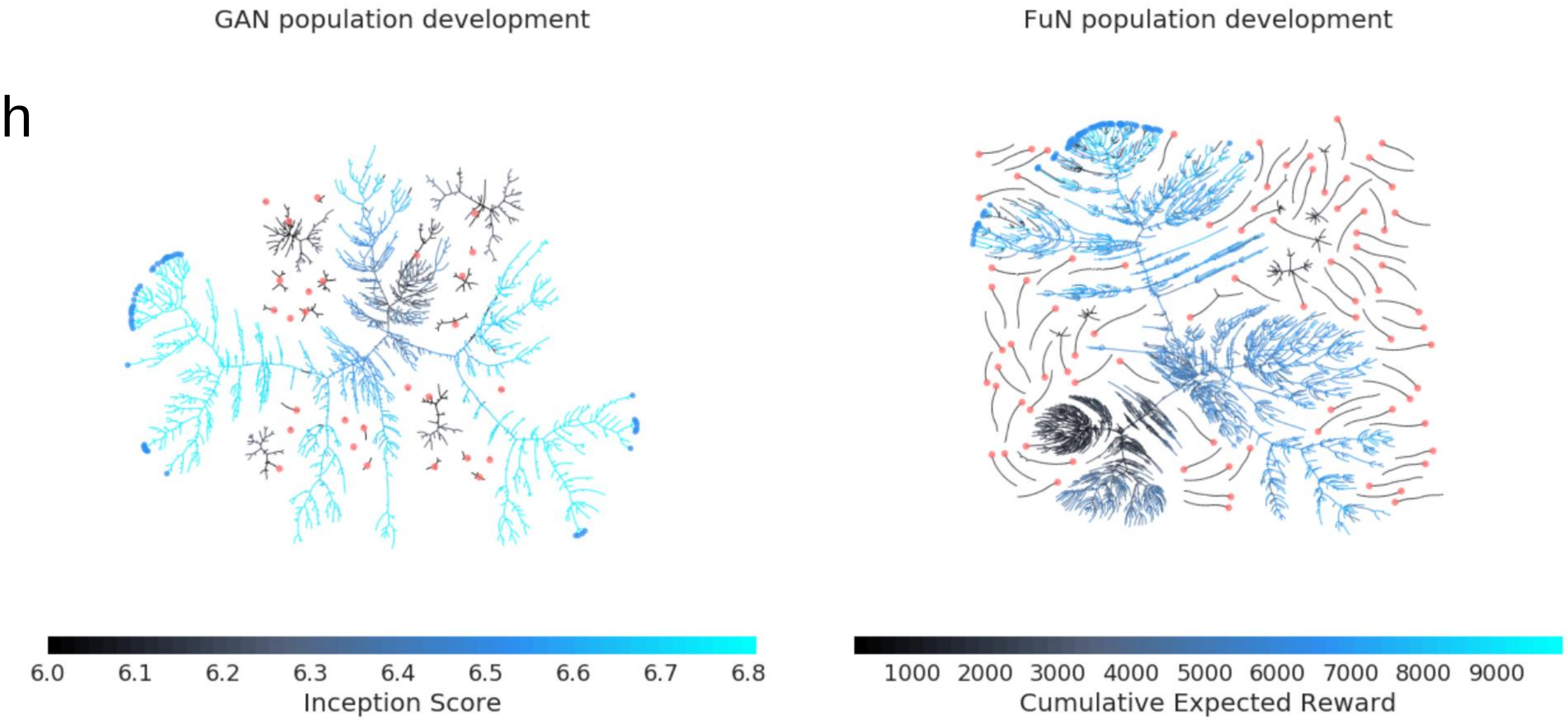
Exploit and explore

- Similar to cloning and mutation genetic operators – but note no recombination
- Typical exploitation: tournament selection, truncation, elitist. May copy entire alternative, or just hyperparameters, omitting model weights
- Explore: can be gradient-based, re-sampling from original prior, or adding random noise/perturbations
- Actual implementation simply applies a multiplier of either 1.2 or 0.8 to hyperparameters (mild perturbations) or 2.0/0.5 (aggressive)

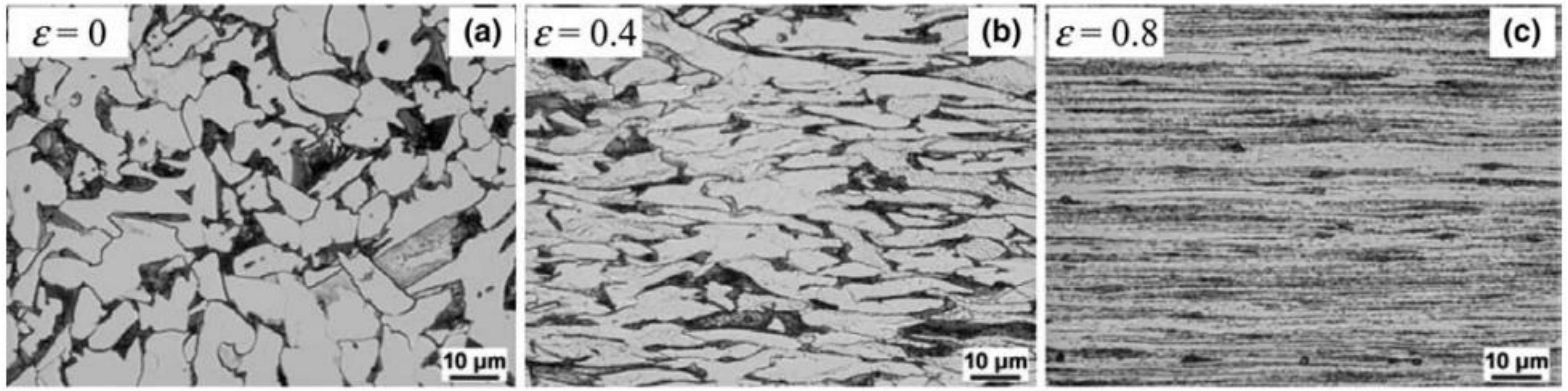
Solution proposed

Output interpretation

- Solutions are not retrained from scratch – model weights are copied over
- So output is not a fixed set of optimal hyperparameters, but actually an adaptive schedule



changing optical microstructure of steel during deformation (work hardening)



Solution proposed

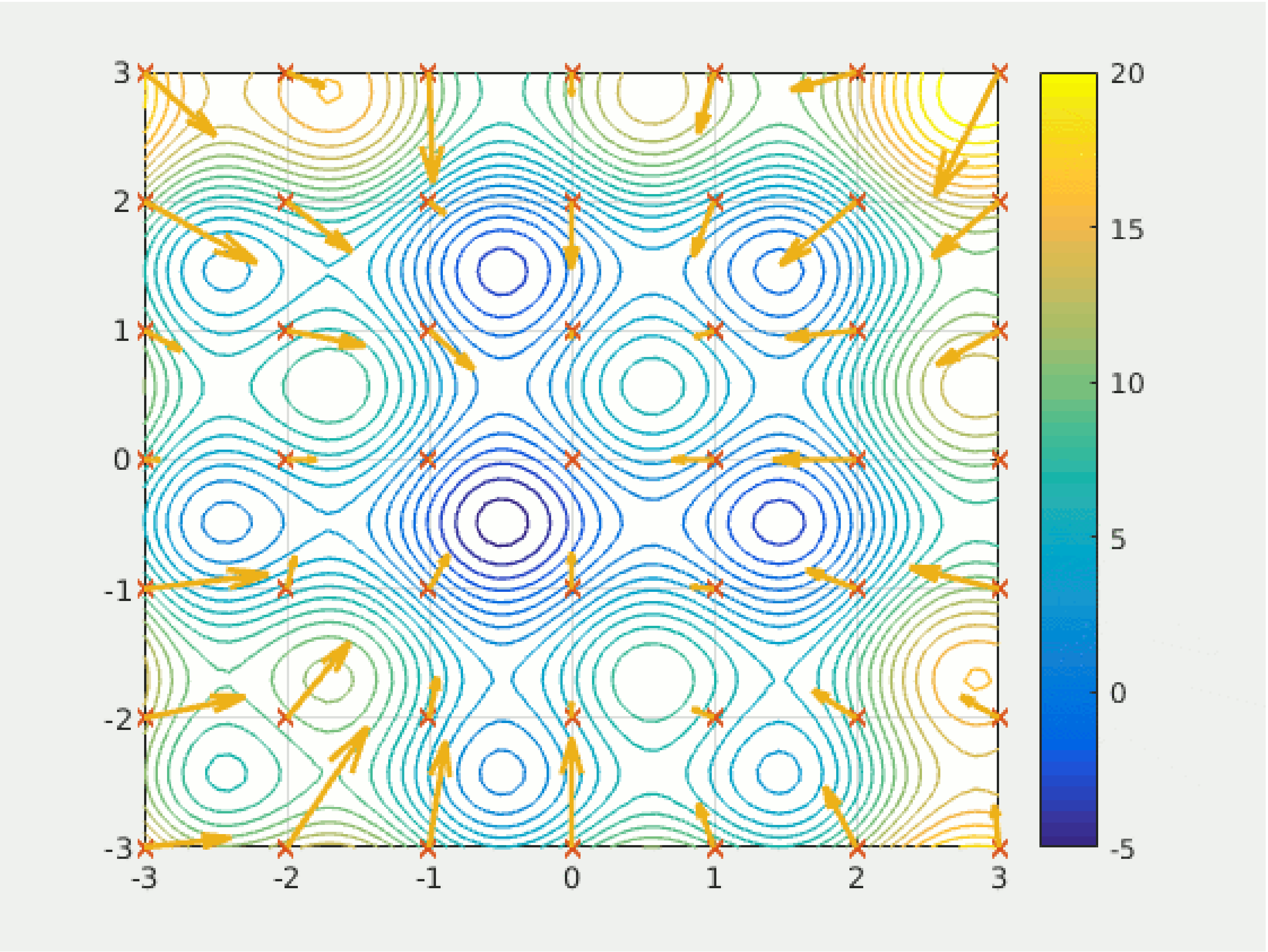
Annealing analogy

- PBT more akin to metal annealing than genetic algorithms
- When working metal, grains becomes brittle and needle-like, dislocations (abrupt changes in structure) introduced in stressed positions
- Heating to gentle glow breaks atomic bonds, relaxes molecular structure, dislocations fall away to stress-free positions
- Slow cooling results in gradual recrystallisation, atoms set in place but remain softer, malleable. <u>Slower cooling = better grain growth</u>
- Simulated annealing: aims to replicate this with hyperparameters (or weights…); over time, become less tolerant of poor solutions

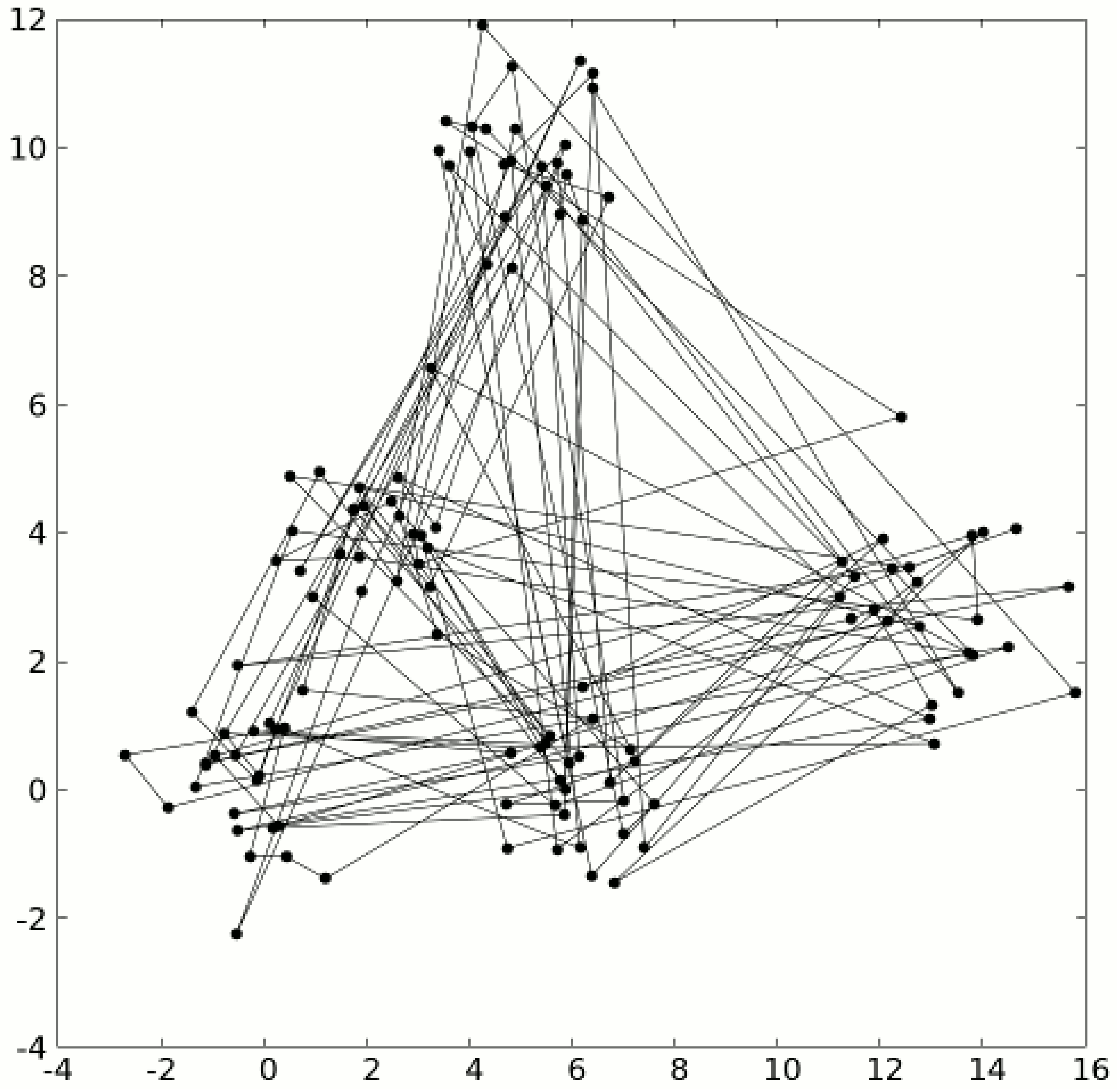
Existing work

- Particle swarm optimisation uses knowledge from population, but keeps individuals separate, i.e. no branching
- Simulated annealing, is itself its own optimisation technique
- Obvious parallels with REGAL, but more different than at first glance
- Practical Bayesian optimisation of ML algorithms (Snoek, Lerochelle, Adams, 2012) approach from other direction, starting with a Gaussian Process, then parallelising it

particle swarm optimisation



simulated annealing



Concluding thoughts

- No constant set of hyperparameters output, so really more ‘model optimisation’ than hyperparameter optimisation. Hyperparameters optimised just like weights, only with lower frequency updates
- Not wholly convinced by the decision to perturb parameters by 1.2/0.8 multiplier rather than adding random noise. (May be to keep better track of annealing schedule, though questionable benefit. As likely for sake of simplicity)

References

- Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, Fernando C. Population based training of neural networks. arXiv preprint arXiv:1711.09846. 2017 Nov 27.

- DNA icon made from Icon Fonts (http://www.onlinewebfonts.com/icon) is licensed by CC BY 3.0

- Random search vs Bayesian optimisation comparison: J. Bergstra, D. Yamins, and D. D. Cox. 2013. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13)

- James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, null (3/1/2012), 281–305.

- Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 2951–2959.

- Particle swarm optimisation and simulated annealing gifs obtains under CC0 1.0 https://en.wikipedia.org/wiki/Simulated_annealing https://en.wikipedia.org/wiki/Particle_swarm_optimization