Population Based Training of Neural Networks

M. Jaderberg, V. Dalibard, S. Osindero, W.M. Czarnecki

November 14, 2018

DeepMind, London, United Kingdom

Presented by: Devin Taylor

Problem Statement

Problem statement

Neural networks suffer from sensitivity to empirical choices of hyperparameters

Solution

Asynchronous optimisation algorithm that jointly optimises a population of models

Key Idea

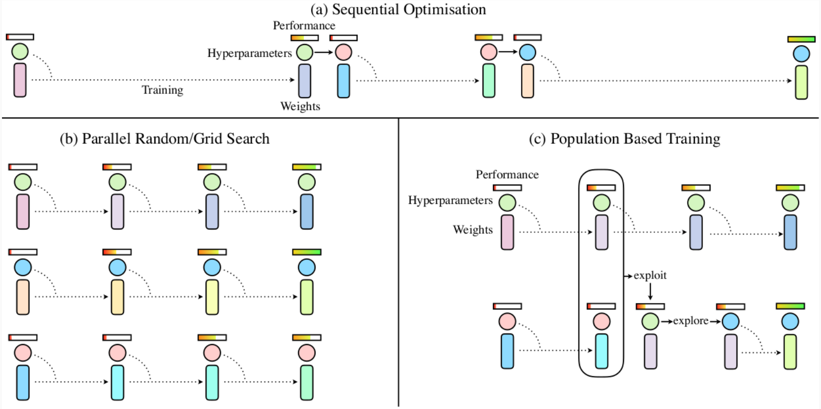

Figure 1: Overview of proposed approach

Population Based Training - Algorithm

- `step` - weight update
- `eval` - performance evaluation
- `ready` - current path limit
- `exploit` - compare to population
- `explore` - adjust hyperparameters

Algorithm 1 Population Based Training (PBT)

```
procedure TRAIN(P)
    for (θ, h, p, t) ∈ P (asynchronously in parallel) do
        while not end of training do
            θ ← step(θ|h)                                    ▷
            p ← eval(θ)
            if ready(p, t, P) then
                h′, θ′ ← exploit(h, θ, p, P)
                if θ ≠ θ′ then
                    h, θ ← explore(h′, θ′, P)
                    p ← eval(θ)
                end if
            end if
            update P with new (θ, h, p, t + 1)
        end while
    end for
    return θ with the highest p in P
end procedure
```

Figure 2: PBT algorithm

Population Base Training - Core

- `exploit`
  - Replace weights and/or hyperparameters
  - T-test selection, truncation selection, binary tournament
- `explore`
  - Adjust hyperparameters
  - Perturb, resample

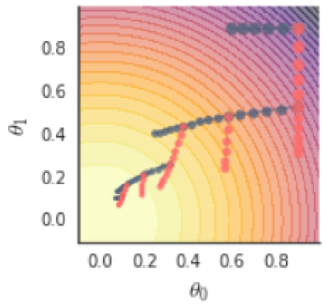

Figure 3: PBT dummy example

Implementation Notes

- Asynchronous
- No centralised orchestrator
- Only current performance information, weights, hyperparameters published
- No synchronisation of population

Experiments

Experiments conducted in three areas:

- **Deep reinforcement learning** - Find policy to maximise expected episodic return
- **Neural machine translation** - Convert sequence of words from one language to another
- **Generative adversarial networks** - Generative models with competing components, *generator* and *descriminator*

Results - Spoiler

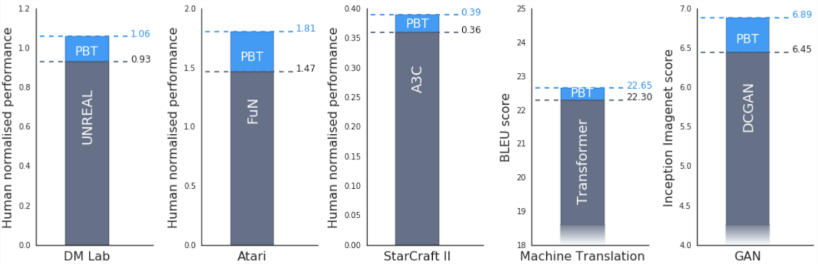

Figure 4: PBT result summary

Results - Deep reinforcement learning

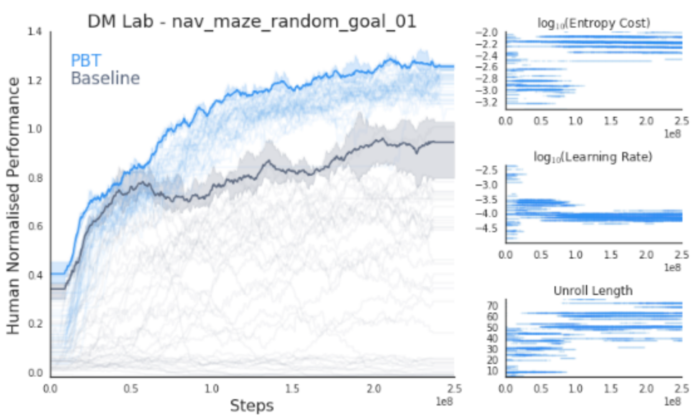

Figure 5: PBT deep reinforcement learning result - DM Lab

Results - Machine translation

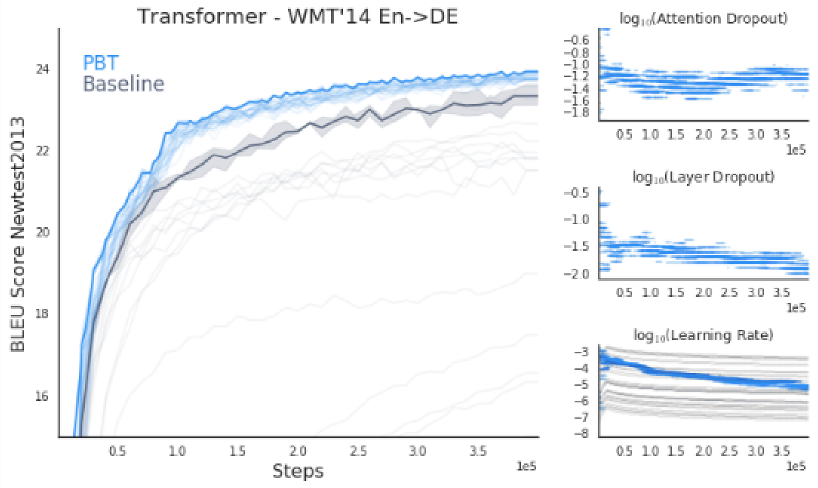

Figure 6: PBT machine translation results

Results - Generative Adversarial Networks

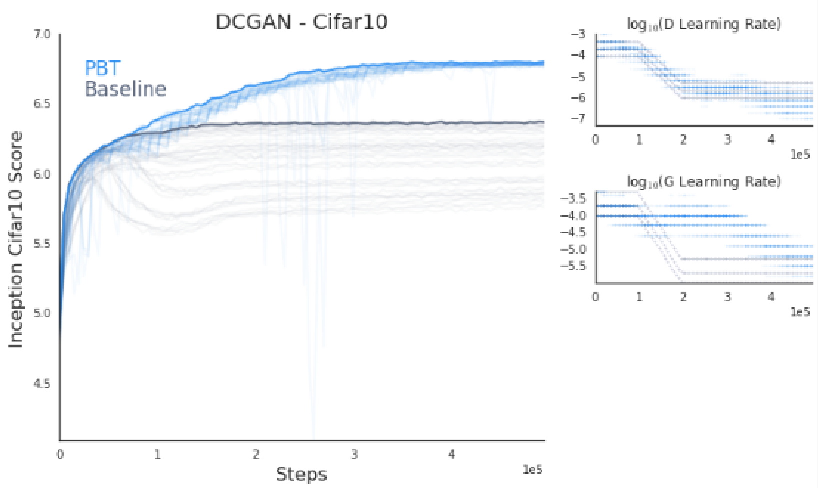

Figure 7: PBT GAN results

Analysis

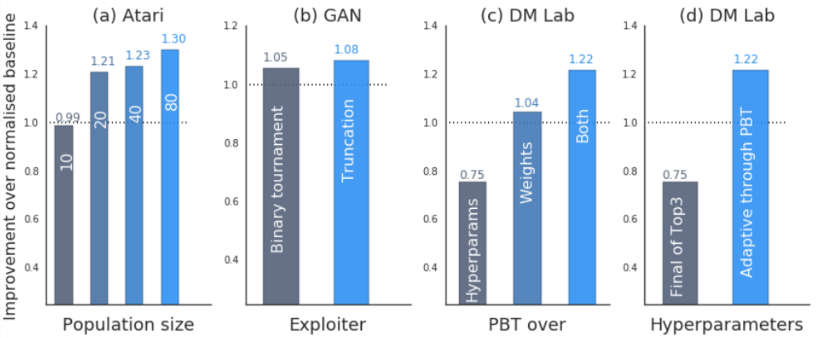

Figure 8: PBT design space analysis

Analysis

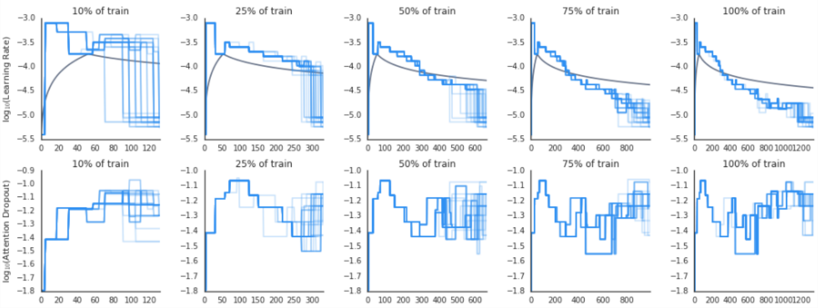

Figure 9: PBT lineage analysis

Analysis

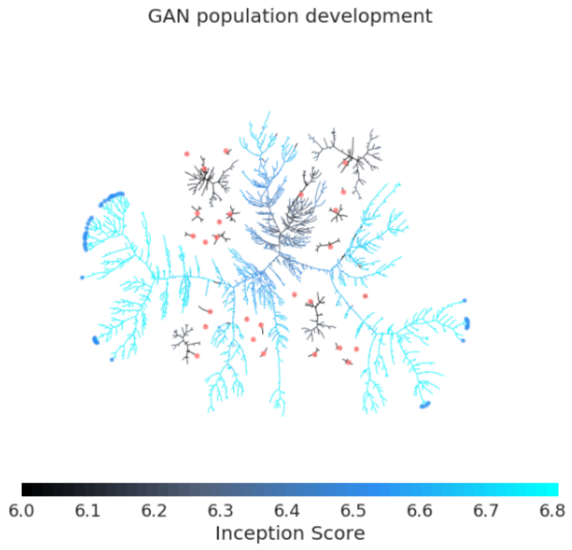

Figure 10: PBT development as phylogenetic tree

Critique

Positives

- Well written
- Detailed analysis - although some questions left unanswered
- Result improvements without sacrificing on time
- Approximate complex paths for hyperparameter tuning
- Improved training stability

Negatives

- No results showing evidence of reduced time
- Added in additional hyperparameters (`ready` steps, perturb, etc)
- Is susceptible to local minima
- Minimum computational requirements (10 workers) quite large

Related Work

- Unique genetic algorithm approach to implementation - parallel and sequential
- Author: Max Jaderberg
  - Mix&Match: Agent Curricula for Reinforcement Learning - boostrapping off simpler agents

Conclusion

- Presented algorithm that asynchronously and jointly optimises a population of models
- Obtained improved results on a range of different algorithms
- Still certain questions unanswered but still a good contribution