Bayesian Generational Population-Based Training

Xingchen Wan, Cong Lu, Jack Parker-Holder, Philip J. Ball, Vu Nguyen, Binxin Ru and Michael A. Osborne

Presented by
Ridwan Muheeb
rm2084
Historical Context

• It has been proven that success of a neural network depends upon the joint tuning of the model structure, its data and the details of how the model is optimized.

• Each of these components of a learning framework is controlled by a number of parameters i.e. hyperparameters (HP) which influence the learning process and must be properly tuned to fully unlock the network performance.

• There are two approaches for doing this:
  • Parallel Search
  • Sequential Search
1. **Sequential Search:**
   - Run few optimizations in parallel but many times sequentially with outputs of previous epochs guiding later epochs in order to find the best case.
   - E.g. hand tuning or Bayesian optimization.
   - Works best but time consuming due long optimization processes.

   ![Sequential Optimisation Diagram](source: Jaderberg, et al, 2017)

   Start a simple rate and decrease by a fixed factor in each epoch e.g. start 0.005 decrease by factor of 10 for each 100 epochs
   0.005 → 0.0005 → 0.00005 → 0.000005
Historical Context

2. Parallel Search:
   - Run multiple optimizations in parallel in bid to find one best output.
   - E.g. grid or random search.
   - Time and computationally expensive.

Source: Jaderberg, et al, 2017
3. **Population Based Training:**

- parallel + sequential optimization methods.
- Start like parallel search, randomly sampling HP and weight initializations.
- Underperforming population model replaces self with a better performing model and explore new HPs by modifying the better model’s HPs before training is continued.
- Allow it to focus on weight space that has best potential to produce good results.
- Proven to be effective in Generative Adversarial Networks (GANs) and Machine Learning Translation.

*Source: Jaderberg, et al, 2017*
Motivation

- Fragility of reinforcement learning to key hyperparameters and choice of network architecture.
- Expensive RL parameters tuning.
- Possibility of obtaining algorithmic optimality at different training points due to changing data distribution.
- Evolving training and data and increased agent complexity.
- Existing Population Based Training styles are not scalable to higher dimensional data.

Solution -> Bayesian Generational Population Based Training
Key Ideas

• Capable of tweaking a large proportion of agents configurations.

• On-the-fly and automatic finetuning of HPs and architectures during training epochs.

• Achieve these using two techniques:
  • Model based HPs architecture exploration steps built on local Bayesian optimization
  • Generational learning which combines PBT and network distillation.

• Experimented for Proximal Policy Optimization (PPO) on Brax, a less computing intensive differentiable physics engine simulation environments.
Algorithm 1 BQ-PIT: distillation and NAS steps marked in magenta (§3.2)

1: Input: pop size $B$, $t_{\text{ready}}$, max steps $T$, $q$ (% agents replaced per iteration)
2: Initialize $B$ agents with weights $\{\theta_0^{(i)}\}_{i=1}^B$, random hyperparameters $\{z_0^{(i)}\}_{i=1}^B$, and architectures $\{y_0^{(i)}\}_{i=1}^B$.
3: for $i = 1, \ldots, T$ (in parallel for all $B$ agents) do
4: Train models & record data for all agents
5: if $t \mod t_{\text{ready}} = 0$ then
6: Replace the weights & architectures of the bottom $q$% agents with those of the top $q$% agents.
7: Update the surrogate with new observations & returns and adjust/restart the trust regions.
8: Check whether to start a new generation (see §3.2)
9: if start a new generation then
10: Clear the gr training data.
11: Create $B$ agents with archs. from BO/random.
12: Distill from a top-$q$% performing agent of the existing generation to new agents.
13: else
14: Select new hyperparameters $z$ for the agents whose weights have been just replaced with randomly sampled configs (if $D = \emptyset$) OR using the suggestions from the no agent described conditioned on $y$ (otherwise).

Source: Wan, et al (2021)

- **Consists of three parts.**
- Use a Bayesian optimization approach to select new HP configurations $z$ for agents.
- Extend the search space to accommodate architecture search to allowing agents to choose their own networks.
- Use on-policy distillation to transfer between different architectures.
Key Ideas
Within a generation

- Consist of three stages.
- **Initialization**: Random HP and weights of different architectures are used for training.
- **Exploitation**: Underperforming agents copies weight and architectures of the best-performing agent.
- **Exploration**: HPs suggestions by time-varying, high-dimensional BO agent.

Source: Wan, et al (2021)
Key Ideas
Across generations

- Consist of two stages.
  
  **Initialization:** Generate 1 random architecture.
  
  Subsequent generations: BO agent performance of the previously generated is used to suggest new architectures.

- **Transfer Knowledge:** (On-policy distillation): Best agents from previous generation guides subsequent ones.

Source: Wan, et al (2021)
Performance – Comparative Evaluation

Source: Wan, et al (2021)

- Experiments conducted on 7 Brax environments.
- Outperforms Random Search, Population Based Training (Jaderberg et al, 2017), PB2 (Parker-Holder et al, 2020) in all the 7 environments.
Performance on Discovered Hyperparameter and Architecture Schedules

Source: Wan, et al (2021)

- Increasing HP size over time during training to model complex behaviors.
- Start with few hyperparameter sizes and increase accordingly to model complex behaviors.
- BG-PBT achieved declining learning rate and batch size increment over time without any pre-defined schedule.
- Result consistent with common practices in deep and reinforcement learning.
Pros

• On-the-fly hyperparameters finetuning to achieve optimal results with less computing resources.

• Results consistent with trends in deep and reinforcement learning domain (declining learning rate and increasing batch size).

• Outperforms existing architectures of PBT based solutions in the simulation environments.
Limitations

- Although the researchers were able to automate Reinforcement Learning hyperparameters using BG-PBT, they recognized the need to automate PBT parameters themselves, e.g., the number of iterations/epochs needed to achieve optimal results.

- Environmental complexity, network architecture sensitivity, and poor selection of architectures can affect the system performance.
Suggestion for Future Research

• Applicability of BG-PBT to other domains outside of reinforcement learning such as GANs, Machine Learning Translation/NLP.
Bibliography

• Jaderberg, Max, et al. "Population based training of neural networks." arXiv preprint arXiv:1711.09846 (2017).

• Wan, Xingchen, et al. "Bayesian Generational Population-Based Training." First Conference on Automated Machine Learning (Main Track). 2022.