A Generalized Framework for Population Based Training

Andrew Tan | CS 294 | Feb 20, 2019

Outline

Background	3
Hyperparameter Optimization	4
Google Vizier	5
Population Based Training	9
Black-box PBT Framework	14
Key Innovations	17
Key Results	19
Conclusion & Future	22

Background

Hyperparameter Optimization

Google Vizier

Population Based Training (PBT)

Hyperparameter Optimization

Success of neural networks often depends on choice of hyperparameters

Variety of algorithms for automatic hyperparameter tuning:

- Grid search
- Random search
- Bandits (Hyperband)
- Evolutionary algorithms

Goals:

1. Find better search algorithms
2. Create a framework to reduce overhead

Google Vizier

Vizier: Problem & Solution

A service for black box optimization

Easy to use

Scalable

State of the Art

Available

Flexible

Vizier: Key Innovations

Easy to use

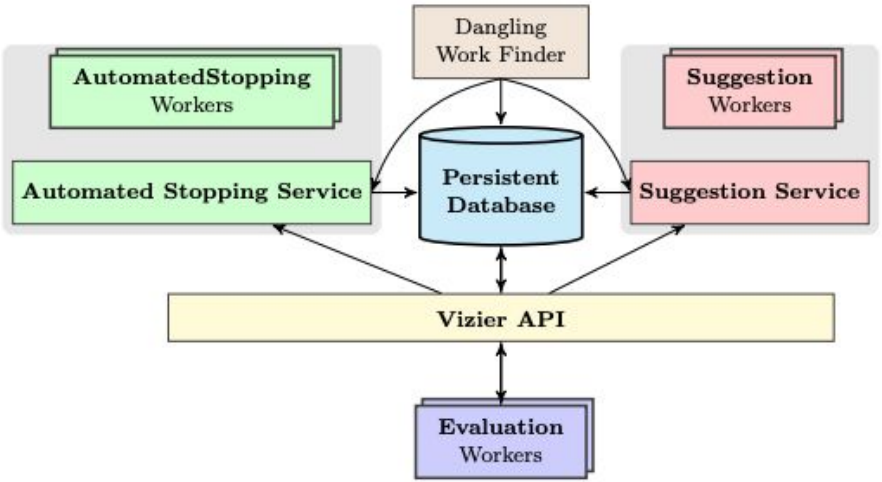
- Minimal configurations
- Simple client workflow

Scalable

- Thousands of parallel evaluations / study
- Millions of trials / study

State of the Art

- Suggestion algorithms are modular



Vizier: Takeaways & Limitations

- Reduce the effort required for setting up a hyperparameter tuning experiment
- High flexibility in the setup of the training procedure in the client side
- Performance is limited by the algorithm used

Population Based Training

PBT: Problem & Solution

Hyperparameter tuning prior to PBT:

- Experience
- Random search
- Computationally intensive search processes

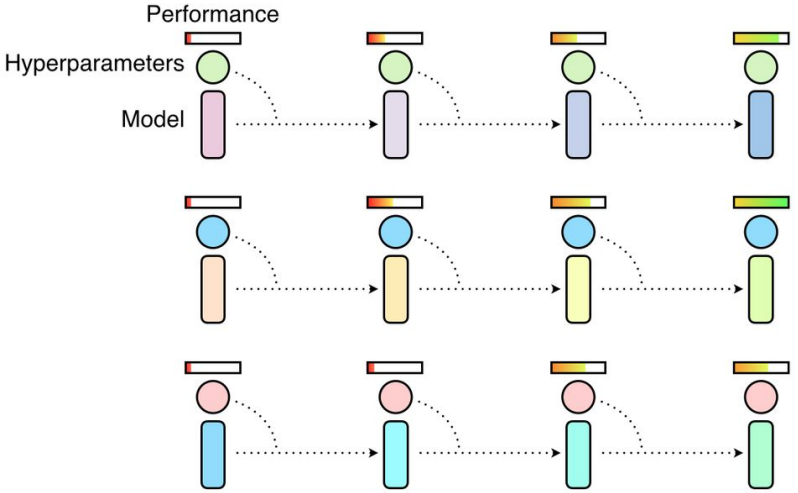
Solution: method that trains and optimizes a series of networks with low overhead

- Hybrid of random search and hand-tuning
- Shares inspiration from evolutionary methods

Random Search & Hand Tuning

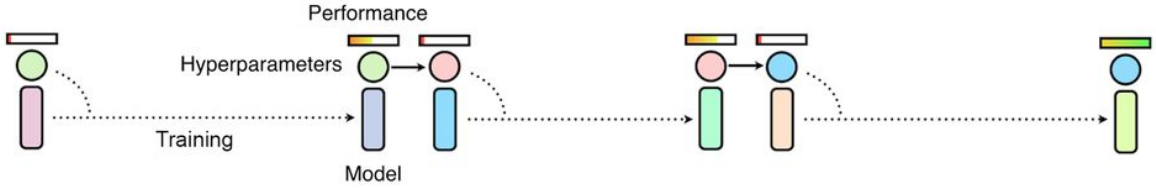
Random Search

- Trained independently in parallel
- Highest performing model selected after convergence
- Wastes lots of resources



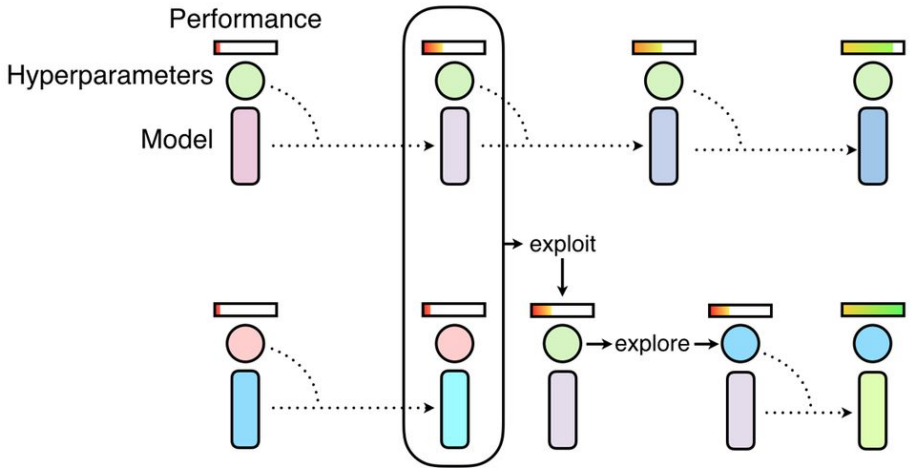
Hand Tuning

- Repeatedly select params, train, and evaluate
- Serial process, time consuming



PBT: Key Innovations

- Start with many networks trained in parallel
- Subsequent trials use information from rest of population
  - Refine hyperparameters
  - Direct computational resources
- Continuously explore and exploit
- Adaptive model, automatic learning
- Warm starts instead of waiting for convergence



PBT: Limitations

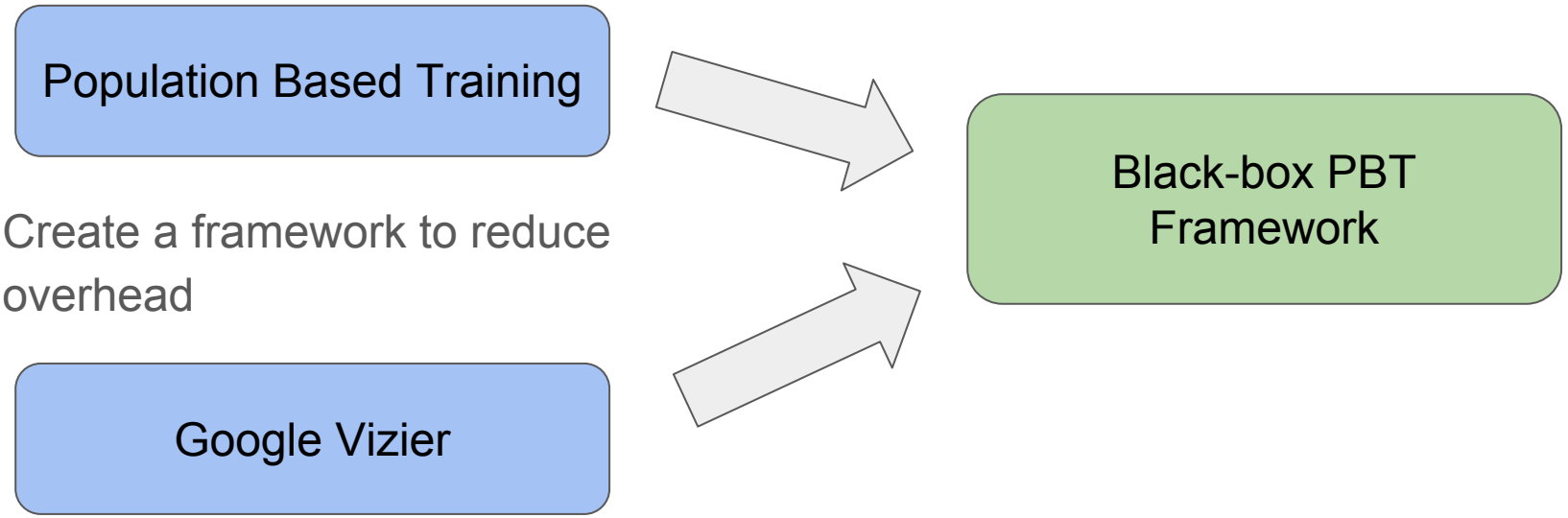
- Changes made to the computation graph can be complicated
- Gracefully handling the case of a worker job being preempted by another worker job
- Not extendable to advanced evolution or mutation decisions

Black-box PBT Framework

Problem

1. Find better search algorithms

2. Create a framework to reduce overhead



Metrics of Success

Train a state-of-the-art WaveNet generative model for human voice synthesis and compare:

- Accuracy
- Sensitivity
- Convergence time

The same outline can be applied to any deep learning application

- Neural machine translation, GANs, reinforcement learning

Key Innovations

- Stateless service
- Black-box, jointly optimize model weights and hyperparameters
- Decision making done by central controller, each trial is small number of steps
- Main Advantages:
  - No need to define hyperparameters in computation graph
  - Allows both differentiable and non-differentiable objectives
  - Allows hyperparameters to be dynamic over time
  - Sufficient scalability and flexibility for low priority workers
  - Flexible: works with most ML model training frameworks

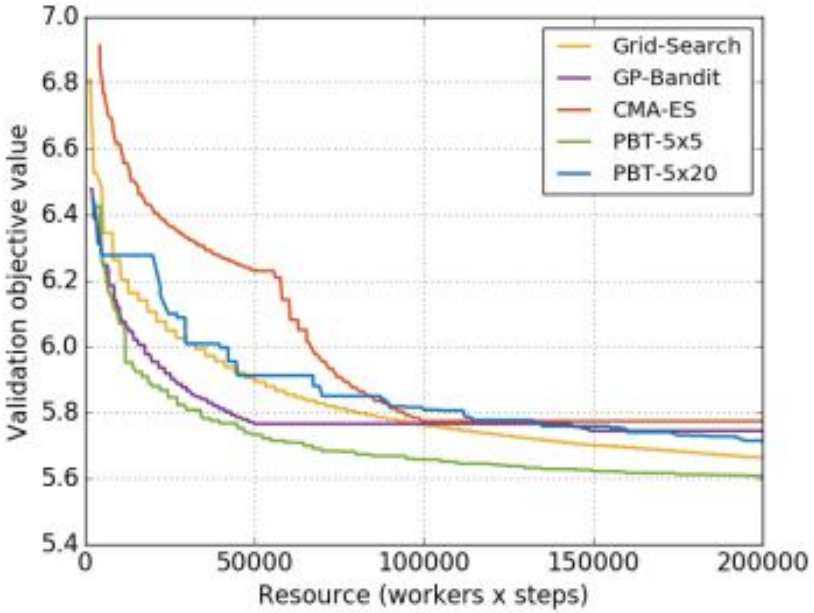
PBT Service Framework

- Trial: continuous training session, configuration defined using protobuf
- Parameters: supports integer, floats, discrete, and categorical values
- Controller: population controller similar to Vizier
  - `GetNewSuggestion(trials, k)`: return list of k new trials given existing trials
  - `GetEarlyStoppingTrials(trials)`: return list of trials that need to be stopped early given existing trials
- Initiator Based Evolution: simple explore/exploit framework, can be extended
  - Fitness representation, reproduction strategy, opponent selection, parent
- Worker: entire training process composed of a trainer and evaluator
  - Parent checkpoint, warm start, continuous evaluation of checkpoints
- Training Replay: large population size, many snapshots
- Training Recovery: stateless, recovery of paused or faulty procedures

Key Results

WaveNet Case Study

- Application of PBT on speech synthesis using WaveNet
- Check for accuracy and performance of PBT system



Key Results

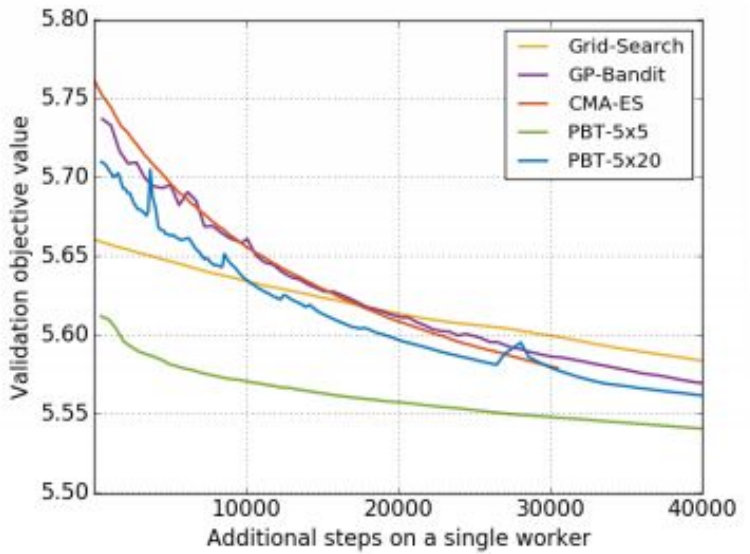


Figure 5: Continue training on a single worker after 200000 resources exhausted, starting with the best checkpoint and its corresponding hyperparameters. Lower objective values are better.

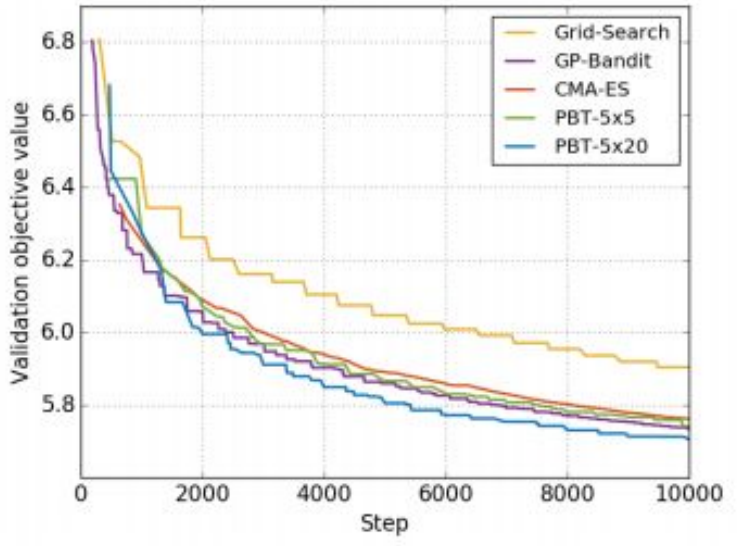


Figure 6: Objective Value vs. Training Step: PBT with 20 population size outperforms all other methods. PBT with 5 population size performs in the second place, which shows that bigger population benefits the model accuracy. Lower values of the objective are better.

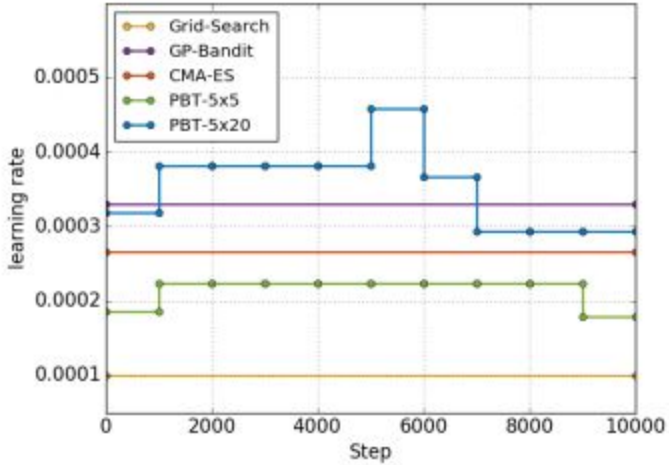


Figure 7: Learning rate schedules found by different approaches.

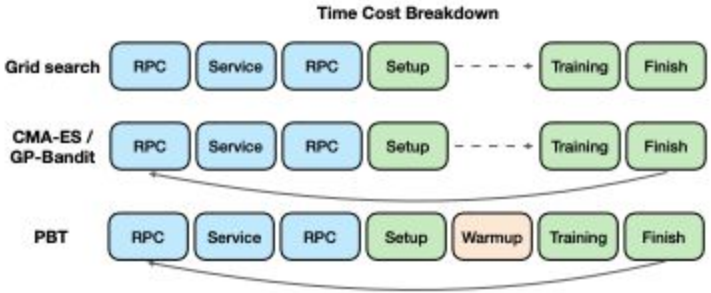


Figure 8: Time cost breakdown for different methods.

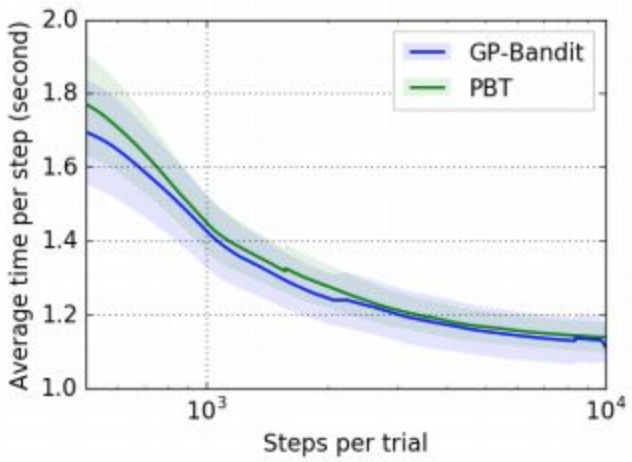


Figure 9: The average time (seconds) per step varies when the number of steps per trial increases. PBT is slightly more expensive than GP-Bandit at the same number of steps (+0.023*s* @ 1K and +0.028*s* @10K), probably due to the extra warm-starting. The shaded area represents the 95% confidence interval.

Conclusion

- General, black-box PBT framework
- Minimal infrastructure and overhead
- No assumptions about architectures or training
- Central controller coordinates asynchronous trials across workers
- Supports dynamic hyperparameter schedules
- Feasible for large scale deep learning
- Scalable and extendable

Future Implications & Research Areas

More than just hyperparameter tuning

Future Research Areas

- Connection with neural architecture search regarding evolutionary methods
- Applying the idea of warm start in other domains
- AutoML - making ML available for non-experts

Discussion

- What other domains can this framework be extended to?
- With the increase in AutoML, what new research problems (opportunities) will that create?