HyperJump: Accelerating HyperBand via Risk Modelling

Pedro Mendes\textsuperscript{1,2}, Maria Casimiro\textsuperscript{1,2}, Paolo Romano\textsuperscript{1}, David Garlan\textsuperscript{2}

\textsuperscript{1} INESC-ID and Instituto Superior Técnico, Universidade de Lisboa
\textsuperscript{2} Software and Societal Systems Department, Carnegie Mellon University
Performing hyper-parameter tuning of complex models using large data sets is very costly.

Idea: Exploit cheap low-fidelity models to extrapolate the quality of fully trained ones.

Approach exploited by a number of recent papers: HyperBand, ASHA, BOHB, Freeze-Thaw BO, Fabolas...
HyperBand

Successive Halving: halves the number of tested configurations while doubling the fidelity in each stage

Ci - Configuration
Bi - Budget (fidelity)
\( \eta \) - Reduction factor
(in this example \( \eta = 2 \))

HyperBand runs SH starting at different budgets
HyperJump: Base idea (1/2)

Can we accelerate HB by skipping the testing of some configurations in the current stage?

Skip the testing of C5, C6, C7, and C8
HyperJump: Base idea (2/2)

… or even completely skipping several stages of HB?

Skip the testing of C5, C6, C7, C8, and the entire stage using budget B2.
To jump or not to jump?

**Key problem**: how to quantify the risk of accelerating HB?

**Risk**: expected quality reduction caused by discarding “wrong” configurations due to a jump
RoadMap

1. How to evaluate the risk of a jump?

1.a How to favor jumps?

1.b Which configurations to include in the target stage?

1.c Quantify the risk given a set of configs to jump

2. Generalization multi-stage jumps

Single stage jumps

Quantify the risk given a set of configs to jump
How to evaluate the risk of jumping?

Consider jumping from stage $s$ to $s+1$, while discarding the configs in $D$.

Risk of discarding the best quality configuration in stage $s$.

Mathematically formalized as:

$$\max_{c \in D} A(c,s) > \max_{c \in S} A(c,s)$$
How to evaluate the risk of jumping?

Consider jumping from stage $s$ to $s+1$, while discarding the configs in $D$

GPs are used to predict the quality of untested configurations

Mathematically formalized as:
\[
\max_{c \in D} A(c,s) > \max_{c \in S} A(c,s)
\]

Easily computed in closed form!
Expected Accuracy Reduction (EAR)

Expected value of the difference of the accuracy of the best configurations in D and S

\[ \text{EAR}(D,S) = \int_{0}^{\infty} x \ P[\max_{c \in D} A(c,s) - \max_{c \in S} A(c,s) = x] \, dx \]

Given that we discard the best configuration when jumping

Computed via numerical methods
Relative Expected Accuracy Reduction

relative Expected Accuracy Reduction:

\[ r\text{EAR}(D,S) = \frac{\text{EAR}(D,S)}{l^*} \]

Rationale: adjust risk quantification to current incumbent quality

HyperJump decides to jump if \( r\text{EAR}(D,S) < \lambda \)

**Default value:** 10% ⇒ robust across a number of benchmarks!
How to favor jumps?

Which configurations to include in the target stage?

Single stage jumps

1. How to evaluate the risk of a jump?

2. How to favor jumps?

Generalization multi-stage jumps

1.a

1.b

1.c

Quantify the risk given a set of configs to jump
Which configurations to include in the target stage?

**Goal:** identify the sets of discarded/selected configs to minimize the risk of jumping

**Problem:** Infeasible to evaluate risk for all possible subsets of configs in the current stage

**Solution:** Use heuristics that leverage the model’s prediction and uncertainty and also account for mispredictions to identify most likely candidates
RoadMap

1. How to favor jumps?
   1.a Which configurations to include in the target stage?
   1.b Single stage jumps
   1.c Generalization multi-stage jumps

2. How to evaluate the risk of a jump?
   Quantify the risk given a set of configs to jump
Generalizing to multi-stage jumps

Accumulate the rEAR of different stages until it reaches the maximum risk allowed ($\lambda$) or the last stage.

\[
\text{Risk}(B_1 \rightarrow B_3) = \text{Risk}(B_1 \rightarrow B_2) + \text{Risk}(B_2 \rightarrow B_3)
\]

Simulate all intermediate stages till target.
RoadMap

How to favor jumps?

1. How to evaluate the risk of a jump?

1.a Single stage jumps
   Quantify the risk given a set of configs to jump

1.b Generalization multi-stage jumps

1.c Which configurations to include in the target stage?

2.
Prioritize testing configurations that are expected to reduce risk the most

**Goal:** favor early jumps

**How:** for each untested configuration $c$, use the model to predict how risk will drop if $c$ is tested

More details in the paper
**Evaluation**

Considered baselines:

- HyperBand
- BOHB
- ASHA
- Fabolas
- BO using EI
- Random search

Sequential and Parallel deployment

Benchmarks:

- NATS-Bench (ImageNet, **Cifar100**, Cifar10) (no. of epochs as budget)
- SVM (Covertype) (data set size as budget)
- **UNET** (time as budget)
- Distributed training of 3 Neural Networks (CNN, MLP, RNN, +NAS) (MNIST) (training time and data set size as budget)
HJ reduces the optimization time around 20x to recommend near-optimal solutions.
Ablation Study

The “HJ-no-Jump” variant includes all HJ’s optimizations except jumping.

Jumping is clearly the mechanism that most contributes to HJ’s efficiency.

Cifar100 (NATS-Bench)
Recommendation Overhead

Average recommendation time 1.08s*, which includes:

- model training
- risk computation (i.e. determine whether to jump)
- select the next configuration to test

(*) using VMs equipped with 16 vCPUs and 16GB of RAM – underlying CPU: AMD EPYC 7501 CPUs
Additional aspects discussed in the paper

1. Additional details on the mathematical modeling of EAR
2. How to preserve HB’s theoretical guarantees
3. How to parallelize HJ
4. Computational complexity of HJ
5. Impact of setting the risk threshold (λ)
6. Evaluating the predictions for the risk jumping
7. Additional optimizations:
   a. Bracket warm-starting
   b. Opportunistic evaluation
   c. Pause-resume training

See the paper
Final Remarks

**HyperJump**: novel system to optimize ML model training that builds on HyperBand's robust search strategy

**Key idea**: skipping the evaluation of configurations that are likely to be anyway discarded by HyperBand.

**EAR**: novel technique that exploits model uncertainty to predict risk of discarding high quality configurations due to jumps.

**Peak gains of up to 20x w.r.t. HB and recent model-based multi-fidelity approaches.**
Thank You!

Link to the paper

Code and dataset available
