

# TrimTuner: Efficient Optimization of Machine Learning Jobs in the Cloud via Sub-Sampling

**Pedro Mendes**<sup>1</sup>, Maria Casimiro<sup>1,2</sup>, Paolo Romano<sup>1</sup>, David Garlan<sup>2</sup>

<sup>1</sup> INESC-ID and Instituto Superior Técnico

<sup>2</sup> Institute of Software Research, Carnegie Mellon University



# Contents

1. Motivation
2. TrimTuner
3. Evaluation
4. Final Remarks

# Motivation: Training large models on the cloud

- In recent years:
  - Machine Learning (ML) **models** are increasing in **complexity**
  - **Data-set grow larger and larger**
- Training process involves **enormous** amount of **computational resources**
- **Cloud** provides access to “unlimited” **on-demand resources** to train ML models (and it represents the standard approach nowadays)

# Motivation: which cloud and model config. to use?

- Cloud providers offer a **large spectrum of Virtual Machines** (VMs) with different types, sizes, and prices
- **Cloud resources** and models **hyper-parameters** need to be **tuned jointly**, yielding a **vast search space**
- Furthermore, **testing** even a single **configuration** can be very **expensive** with current **large data-sets**

# Motivation: joint optimization of cloud&model parameters

- The wrong configuration selection can significantly **reduce model's accuracy** and **amplify operational costs**
- **Disjoint optimization** is unable to identify optimal configurations:
  - up to 3.7x more expensive configurations with Neural Network models [ICDCS20]

Fundamental to **joint optimize**  
cloud and application parameter

# TrimTuner

- System to optimize the training of **ML jobs** in the **cloud**
- Exploits data **sub-sampling** techniques in order to enhance the efficiency of the optimization process
- **Jointly optimizes** the cloud configuration and hyper-parameters of ML models

# TrimTuner - Optimization Problem

$$\begin{aligned} & \underset{\mathbf{x} \in \mathbb{X}}{\text{maximize}} && A(\mathbf{x}, s = 1) \\ & \text{subject to} && q_1(\mathbf{x}, s = 1) \geq 0, \dots, q_m(\mathbf{x}, s = 1) \geq 0 \end{aligned}$$

$\mathbf{x}$ : configuration                       $s$ : data-set size [%]                       $\mathbb{X}$ : set of configurations to test  
 $A(\mathbf{x}, s)$ : accuracy of the ML model  
 $q_i(\mathbf{x}, s)$ : Quality of Service (QoS) constraints (e.g., cost or execution time)

# TrimTuner

- Deploy the job using **sub-sampled data-sets to reduce the cost** of testing individual configurations
- **Build predictive models** that keep into account how shifts of the data-set size affect the accuracy and cost of the target job
- Use **Gaussian Processes** (GPs) and Ensemble of **Decision Trees** (DTs) as modelling techniques



# TrimTuner

- Leverages **Bayesian Optimization** (BO) techniques to update the models and select the next configuration to evaluate
- Does not assume **any *a priori* knowledge** about the target job and the training platform

# Bayesian Optimization: base idea

- Aims at identifying the optimum of an unknown function  $f$
- Does not assume any *a priori* knowledge about the target job
- Builds black-box model of  $f$ :

1. Test a set of initial configurations

2. Build a model of  $f$

3.  $\mathbf{x}_i = \arg \max_{\mathbf{x} \in \mathbb{X}} \{\alpha(\mathbf{x})\}$       $\mathbb{X}$ : untested set;  $\alpha$ : acquisition function;

4. Test the configuration  $\mathbf{x}_i$

5. Update the model of  $f$

**Problem**: testing even a single configuration can be **very costly** with large data sets

# BO using sub-sampling - FABOLAS [AISTATS17]

- Aims to maximize ML model accuracy by optimizing its hyper-parameters
- Leverages BO to solve the optimization problem
- Trains the model with sub-sampled data-sets to reduce the cost of testing configurations

# BO + sub-sampling: base idea

- Additional dimension in the search space that corresponds to the data-set size

1. Test a set of initial configurations  $x$  in all the sub-sampled data-set sizes

2. Build a model of  $f$

3.  $(\mathbf{x}_i, s_i) = \arg \max_{\mathbf{x} \in \mathbb{X}, s \in S} \{\alpha(\mathbf{x}, s)\}$

$\mathbb{X}$ : untested set;  $S$ : sub-sampled data-set sizes;  $\alpha$ : acquisition function;

4. Test the configuration  $(\mathbf{x}_i, s_i)$

5. Update the model of  $f$



# BO + sub-sampling: FABOLAS acquisition function

- **Information gain per unit cost**


- **Entropy Search** [JMLR12]: estimates the information on the optimum using the full data-set ( $s=1$ ) that can be gained by testing a sub-sampled configuration  $(\mathbf{x}, s)$ , where  $s < 1$ .
- ...normalized by the cost of testing  $(\mathbf{x}, s)$

$$\alpha_F(\mathbf{x}, s) = \frac{1}{C(\mathbf{x}, s)} \mathbb{E}_{p(y|\mathbf{x}, s, \mathcal{S})} \left[ \int p_{opt}^{s=1}(\mathbf{x}' | \mathcal{S} \cup \{\mathbf{x}, s, y\}) \log \frac{p_{opt}^{s=1}(\mathbf{x}' | \mathcal{S} \cup \{\mathbf{x}, s, y\})}{u(\mathbf{x}')} d\mathbf{x}' \right]$$

Normalize by the  
cost of testing  $(\mathbf{x}, s)$

Entropy Search

# FABOLAS: Limitations

1. Does not keep into account any **QoS related constraints**
    - e.g. on model's training cost or latency.
  2. Only optimizes the hyper-parameters of the ML model:
    - Underlooks the problem of **jointly optimizing** cloud parameters  
⇒ Much smaller search spaces & globally sub-optimal recommendations
  3. The acquisition function is **very expensive** to compute:
    - Problem exacerbated with **large search** spaces:
      - i. up to 13 minutes to recommend a configuration (Search space: 1440)
- 

# TrimTuner

1. Novel **acquisition function** to keep into account **QoS constraints**
2. **Optimizes** both the **hyper-parameters** and **cloud resources**
3. Lightweight models based on **Decision Trees** (vs Gaussian Processes)  
**Novel filtering heuristic** to reduce the number of configurations to compute the acquisition function for.

# TrimTuner: acquisition function

- Determines the probability that the new optimum is feasible
- Estimates the information gain of evaluating  $(\mathbf{x}, s)$  about the optimum
- Normalizes by the cost of evaluating  $(\mathbf{x}, s)$ .

Need to **simulate the models** using the predicted values

$$\alpha_T(\mathbf{x}, s) = \mathbb{E}_{p(\mathbf{q}, a | \mathbf{x}, s, \mathcal{S})} \left[ \prod_{q_i \in \mathbf{Q}} p(q_i(\mathbf{x}^*, s = 1) \geq 0 | \mathcal{S} \cup \{\mathbf{x}, s, \mathbf{q}, a\}) \right] \alpha_F(\mathbf{x}, s)$$

Probability that the new predicted incumbent meets the constraints

FABOLAS acquisition function



# Speeding up the recommendation process

- The computation of FABOLAS acquisition function is very **expensive**
- TrimTuner tackles this issue by:
  - **Decreasing the number of configurations** for which the acquisition function is evaluated:
    - key idea: filtering non-promising configuration via a fast heuristic
  - Relying on **ensembles of Decision Trees**
    - Unlike FABOLAS, which exploits Gaussian Processes

# TrimTuner: filtering out non-promising configurations

- Introduces the Constrained Expected Accuracy (CEA) as the filtering heuristic

$$CEA(\mathbf{x}, s) = A(\mathbf{x}, s) \cdot \prod_{q_i \in \mathbf{Q}} p(q_i(\mathbf{x}, s) \geq 0 | \mathcal{S})$$

- CEA can be seen as a rough, but cheap, approximation of the acquisition function
- It is computed for all the untested configurations and selects configuration with largest CEA

# Evaluation

- Considered baselines:
  - FABOLAS
  - BO using Constrained Expected Improvement (Elc)
  - BO using Elc/USD
  - Random search
- Search space composed of **1440 configurations**;
- Target jobs: Distributed training of **3 Neural Networks** (NNs) deployed in AWS EC2

# Configurations

## Cloud parameters

VM Type	VM Characteristics	#VMs
t2.small	1 vCPU, 2GB RAM	8, 16, 32, 48, 64, 80
t2.medium	2 vCPU, 4GB RAM	4, 8, 16, 24, 32, 40
t2.xlarge	4 vCPU, 16GB RAM	2, 4, 8, 12, 16, 20
t2.2xlarge	8 vCPU, 32GB RAM	1, 2, 4, 6, 8, 10

Parameter	Values
Learning Rate	$10^{-3}$ , $10^{-4}$ , $10^{-5}$
Batch size	16, 256
Training mode	synchronous, asynchronous
Data-set size [%]	1/60, 1/10, 1/4, 1/2, 1

## Hyper-parameters

**Our data-set is publicly available:**  
<https://github.com/pedrogbmendes/TrimTuner>

# Constraints

Only  $\approx 10\%$  of configurations are feasible and have a high accuracy

Neural Network	Feasible Configurations	Feasible configurations with high accuracy
RNN	178 (61.8%)	28 (9.72%)
MLP	161 (55.8%)	29 (10.07%)
CNN	111(38.5%)	39 (13.54%)

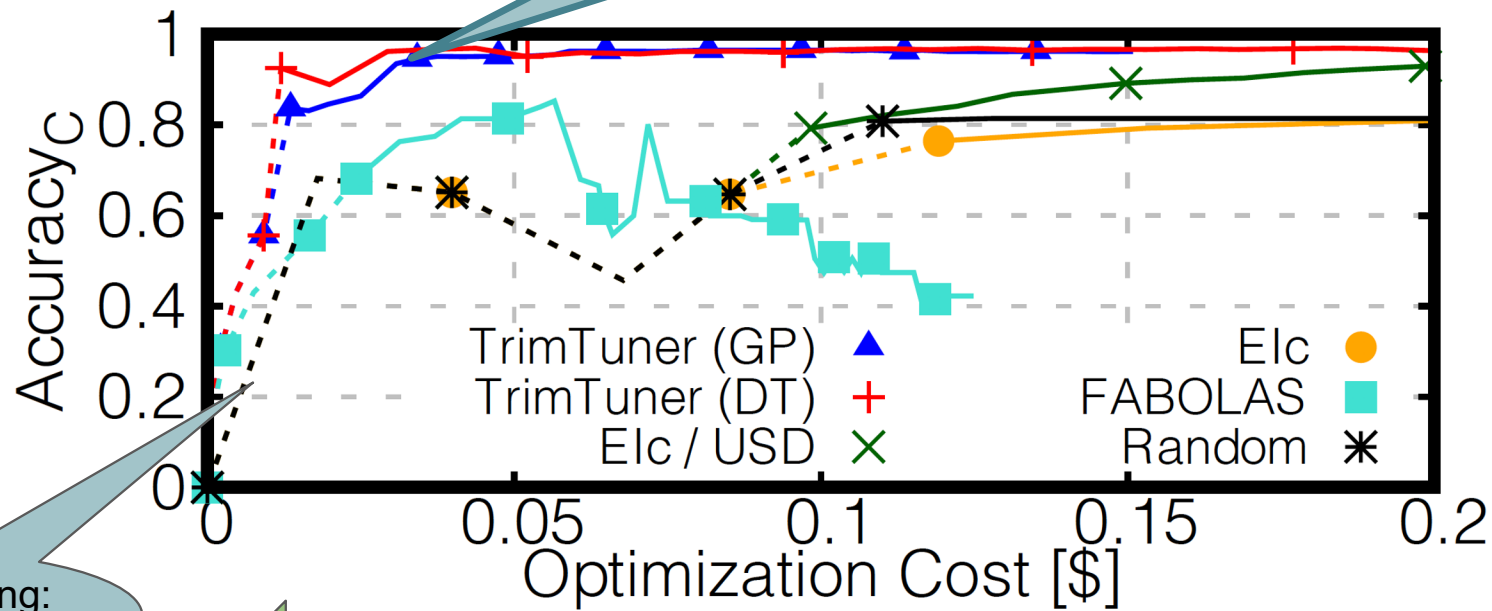
# Evaluation Metrics

- Constrained Accuracy:

$$Accuracy_C = \begin{cases} A(x, s), & \text{if } (x, s) \text{ is feasible} \\ A(x, s) \cdot \frac{C_{max}}{C(x, s)}, & \text{otherwise} \end{cases}$$

- We report the average value of 10 runs.

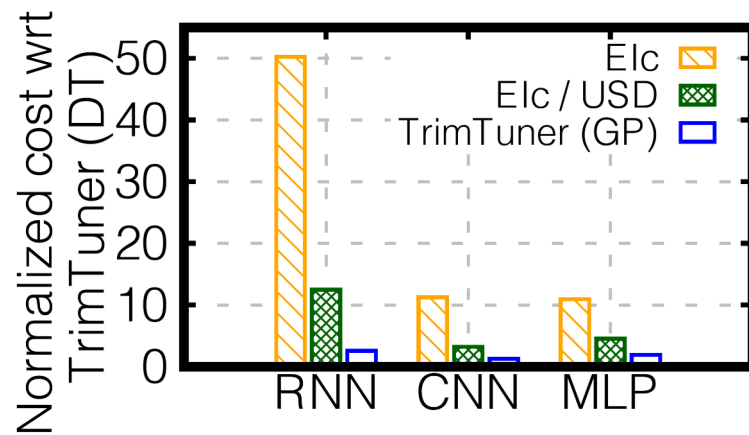
# RNN



TrimTuner recommends configurations close to optimum (within 5%) reducing the optimization cost by a factor of **50x** w.r.t. Elc

Initial sampling: dashed lines

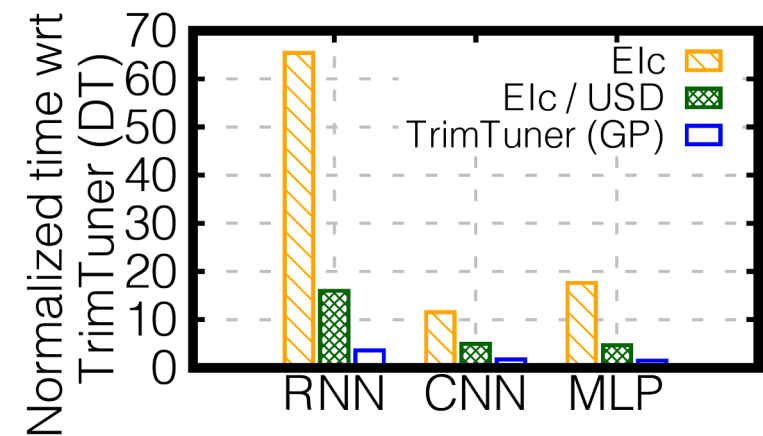
# TrimTuner Gains



Cost savings

Cost reduced by up to:

- **50x** w.r.t. Elc
- **≈10x** w.r.t Elc/USD



Time speed ups

Time accelerated by up to:

- **65x** w.r.t. Elc
- **15x** w.r.t Elc/USD



# Recommendation Times

Optimizer	Avg. time to recommend a configuration [min]
TrimTuner (GPs)	18.65
<b>TrimTuner (DTs)</b>	<b>1.36</b>
Fabolas	13.96
Elc (Elc/USD)	1.17

Recommendation time  
reduced by **13.7x**

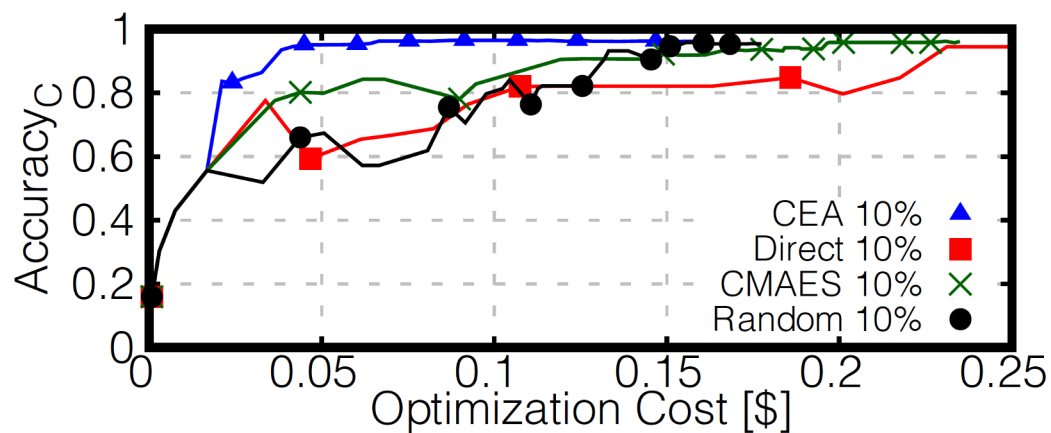
# Efficiency of the CEA

## 1. CEA compared with other filtering heuristics:

- Direct [JGO01]
  - CMAES [EC03]
  - Random approach
- } Two state-of-the-art black-box optimizers

# Efficiency of the CEA

RNN



CEA reduces the recommendation cost by up to **7x/3.6x** w.r.t. Direct/CMAES

Filtering Heuristic	Rec. Time TT DTs [min]	Rec. Time TT GPs [min]
CEA	<b>1.72</b>	16.85
Direct	2.63	36.18
CMAES	2.26	30.87
Random	1.62	16.53

# Final Remarks

- TrimTuner: novel system to **optimize ML model training in the cloud**
  - Maximizes model accuracy subject to QoS constraint
  - Exploits sub-sampling data to reduce the costs
- TrimTuner relies on subsampling techniques to reduce the:
  - Training cost by up to **50x**,
  - Latency of the exploration process by up to **65x** (compared with BO using Elc);
- It accelerates the recommendation process by up to **117x** via
  - Novel filtering heuristic: CEA;
  - Lightweight predictive models based on DTs.

Check our paper  
for more details

Code and data-sets  
are available online

<https://github.com/pedrogbmendes/TrimTuner>