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

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

\textsuperscript{1} INESC-ID and Instituto Superior Técnico
\textsuperscript{2} Institute of Software Research, Carnegie Mellon University
Contents

- Motivation
- TrimTuner
- Evaluation
- Ongoing Work
- Final Remarks
Training large models

- In recent years:
  - Machine Learning (ML) is ubiquitous
  - ML models are increasing in complexity
  - Data-set grow larger and larger

- Training process involves enormous amount of computational resources
Training large models on the cloud

- **Cloud** provides access to:
  - “unlimited” on-demand resources
  - a large spectrum of Virtual Machines (VMs) with different types, sizes, and prices

...that can be used to train ML models

(it represents the standard approach nowadays)
## Cloud Cost & Ecological Footprint

<table>
<thead>
<tr>
<th>Model Description</th>
<th>Energy Consumption (kWh)</th>
<th>CO2 Emissions (Kg of CO2e)</th>
<th>Cloud Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (65M parameters) (Jun, 17)</td>
<td>27</td>
<td>12</td>
<td>$41-$140</td>
</tr>
<tr>
<td>Transformer (213M parameters) (Jun, 17)</td>
<td>201</td>
<td>87</td>
<td>$289-$981</td>
</tr>
<tr>
<td>ELMo (Feb, 18)</td>
<td>275</td>
<td>119</td>
<td>$433-$1,472</td>
</tr>
<tr>
<td>BERT (110M parameters) (Oct, 18)</td>
<td>1507</td>
<td>652</td>
<td>$3,751-$12,571</td>
</tr>
<tr>
<td>Transformer (213M parameters) (NAS) (Jan, 19)</td>
<td>656347</td>
<td>284019</td>
<td>$942,973-$3,201,722</td>
</tr>
<tr>
<td>GPT-2 (Feb, 19)</td>
<td>-</td>
<td>-</td>
<td>$12,902-$43,008</td>
</tr>
</tbody>
</table>

Which cloud & model configuration to use?

- Large spectrum of VMs

- Several hyper-parameters defined over large domains

- Testing even a single configuration can be very expensive with current large data-sets
Joint optimization of cloud & model parameters

- **Disjoint optimization** is unable to identify optimal configurations:
  - up to 3.7x more expensive configurations with Neural Network [ICDCS20]

- The wrong configuration selection can significantly **reduce model’s accuracy** and **amplify operational costs**

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

Yielding vast search spaces
TrimTuner

System to optimize the training of ML jobs in the cloud

Exploits sub-sampling techniques to enhance the efficiency of the optimization process

Jointly optimizes the cloud configuration and hyper-parameters of ML models
TrimTuner - Optimization Problem

\[
\begin{align*}
\text{maximize} \quad & \quad & A(x, s = 1) \\
\text{subject to} \quad & \quad & q_1(x, s = 1) \geq 0, \ldots, q_m(x, s = 1) \geq 0
\end{align*}
\]

\[x: \text{configuration} \quad \quad \quad s: \text{data-set size } [\%] \quad X: \text{set of configurations to test} \]

\[A(x,s): \text{accuracy of the ML model} \]

\[q_i(x,s): \text{Quality of Service (QoS) constraints (e.g., cost or execution time)} \]
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
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 $f$
BO: Algorithm

1. Test a set of initial configurations
2. Build a model of $f$
3. $x_i = \arg \max_{x \in \mathcal{X}} \{a(x)\}$
   - $\mathcal{X}$: untested set
   - $a$: acquisition function
4. Test the configuration $x_i$
5. Update the model of $f$
Typically, Gaussian Processes are used in BO:

- Provides not only information about a configuration, but also its uncertainty (the prediction follows a normal distribution)

- Analytical tractability and flexibility (the covariance function can be chosen)

- Continuous and discrete

As we are going to see next, GPs have a high computational cost
Alternatively, it is possible to use an ensemble of learners:

- Predicted value - determines the average of the prediction of each learner
- Uncertainty - calculate the standard deviation using the predictions of each learner

BO: Model

TrimTuner uses Ensemble of Decision Trees (DTs)
BO: Acquisition Function

Exploits the model’s knowledge and uncertainty to determine which configuration to evaluate next by balancing:

- **Exploitation** - recommend configs that the model deems to be optimal

- **Exploration** - recommend configs whose knowledge can reduce the model’s uncertainty and enhance its accuracy
BO: Acquisition Function

- **Expected Improvement:**
  - Estimates the expected amount by which evaluating $f$ at $x$ can improve over the current best value (or incumbent)
  
  $$
  \alpha_{EI}(x) = \int \max(0, f(x) - \eta)p(f(x)|S)df(x)
  $$

- **Entropy Search:**
  - Estimates the Information that can be gained about the optimum by evaluating $f$ at $x$
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 configs $x$ in all the sub-sampled data-set sizes

2. Build a model of $f$

3. $(x_i, s_i) = \arg \max_{x \in \mathbb{X}, s \in S} \{a(x, s)\}$
   - $\mathbb{X}$: untested set
   - $S$: sub-sampled data-set sizes
   - $a$: acquisition function

4. Test the config $(x_i, s_i)$

5. Update the model of $f$
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 ($x,s$), where $s<1$.
- ...normalized by the cost of testing ($x,s$)

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

Normalize by the cost of testing ($x,s$)

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
   - Smaller search spaces & globally sub-optimal recommendations

3. **The acquisition function is very expensive to compute**
   - Problem exacerbated with **large search spaces** (up to 13 minutes to recommend a configuration (Search space: 1440 configs))
TrimTuner

Novel acquisition function to keep into account QoS constraints

Optimizes both the hyper-parameters and cloud resources

Lightweight models based on Decision Trees (vs GPs)

Novel filtering heuristic to reduce the number of configurations to compute the acquisition function
TrimTuner: acquisition function

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

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

Need to simulate the models using the predicted values.
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
Introduces the **Constrained Expected Accuracy (CEA)** as the filtering heuristic

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

CEA can be seen as a rough, but cheap, approximation of the acquisition function

TrimTuner: filtering out non-promising configurations
TrimTuner: filtering heuristic

- CEA can be efficiently evaluated for all the untested configurations
- TrimTuner ranks the untested configurations via the CEA
TrimTuner: Models

Allows both modelling techniques:
- Gaussian Processes
- Ensemble of Decision Trees (DTs)
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

Our data-set is publicly available: github.com/pedrogbmendes/TrimTuner

<table>
<thead>
<tr>
<th>VM Type</th>
<th>VM Characteristics</th>
<th>#VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2.small</td>
<td>1 vCPU, 2GB RAM</td>
<td>8, 16, 32, 48, 64, 80</td>
</tr>
<tr>
<td>t2.medium</td>
<td>2 vCPU, 4GB RAM</td>
<td>4, 8, 16, 24, 32, 40</td>
</tr>
<tr>
<td>t2.xlarge</td>
<td>4 vCPU, 16GB RAM</td>
<td>2, 4, 8, 12, 16, 20</td>
</tr>
<tr>
<td>t2.2xlarge</td>
<td>8 vCPU, 32GB RAM</td>
<td>1, 2, 4, 6, 8, 10</td>
</tr>
</tbody>
</table>

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
## Constraints

Only \( \approx 10\% \) of configurations are feasible and have a high accuracy.

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Feasible Configurations</th>
<th>Feasible configurations with high accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>178 (61.8%)</td>
<td>28 (9.72%)</td>
</tr>
<tr>
<td>MLP</td>
<td>161 (55.8%)</td>
<td>29 (10.07%)</td>
</tr>
<tr>
<td>CNN</td>
<td>111 (38.5%)</td>
<td>39 (13.54%)</td>
</tr>
</tbody>
</table>
Evaluation Metrics

Constrained Accuracy:

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

We report the average value of 10 runs.
Optimization Process - RNN

TrimTuner can recommend **better configurations** while reducing the **cost**.

**Initial sampling:** dashed lines
TrimTuner Gains

To recommend configs close to optimum (within 5%)

Cost reduced by up to:
- $\textbf{50x}$ w.r.t. Elc
- $\approx \textbf{10x}$ w.r.t Elc/USD

Time accelerated by up to:
- $\textbf{65x}$ w.r.t. Elc
- $\textbf{15x}$ w.r.t Elc/USD
## Recommendation Times

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Avg. time to recommend a configuration [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrimTuner (GPs)</td>
<td>18.65</td>
</tr>
<tr>
<td>TrimTuner (DTs)</td>
<td>1.36</td>
</tr>
<tr>
<td>Fabolas</td>
<td>13.96</td>
</tr>
<tr>
<td>Elc (Elc/USD)</td>
<td>1.17</td>
</tr>
</tbody>
</table>

Recommendation time reduced by **13.7x**
Efficiency of the CEA

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.
## Efficiency of the CEA - RNN

<table>
<thead>
<tr>
<th>Filtering Heuristic</th>
<th>Rec. Time TT DTs [min]</th>
<th>Rec. Time TT GPs [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEA</td>
<td>1.72</td>
<td>16.85</td>
</tr>
<tr>
<td>Direct</td>
<td>2.63</td>
<td>36.18</td>
</tr>
<tr>
<td>CMAES</td>
<td>2.26</td>
<td>30.87</td>
</tr>
<tr>
<td>Random</td>
<td>1.62</td>
<td>16.53</td>
</tr>
</tbody>
</table>
Computation of acquisition functions based on ES is **very expensive** (particularly, when GPs are used)

New acquisition function (TrimTuner 2.0)
TrimTuner 2.0: acquisition function

Determines the probability that the new optimum is feasible
Estimates the expected accuracy of the new optimum
Normalizes by the cost of evaluating \((x, s)\)

\[
\alpha T_2(x, s) = \frac{1}{C(x, s)} \cdot \mathbb{E}_{p(q, a \mid x, s, S)} \left[ A(x^*, s = 1 \mid S \cup \{x, s, q, a\}) \right] \cdot \mathbb{E}_{p(q, a \mid x, s, S)} \left[ \prod_{q_i \in Q} p(q_i(x^*, s = 1) \geq 0 \mid S \cup \{x, s, q, a\}) \right]
\]

Expected accuracy of the new predicted inc.

Probability that the new predicted inc. meets the constraints
Ongoing Work

New filtering heuristic

- The filtering heuristic can be seen as a rough, but cheap approximation of the acquisition function.
- The acquisition function is normalized by the cost (in order to sample cheaper configs).

Constrained Expected Accuracy per unit cost
CEA/USD

$$CEA/USD(x, s) = \frac{A(x, s)}{C(x, s)} \cdot \prod_{q_i \in Q} p(q_i(x, s) \geq 0 | S)$$

Advantages: selects cheaper configs using smaller data-set and thus reduces the optimization cost.
New Acquisition Function & Filtering Heuristic - RNN

The joint use of the new acq. func. and new filtering heuristic reduces the optimization cost by up to $5.6x$. 

![Graph showing accuracy vs. optimization cost for TrimTuner2.0, TrimTuner1.0, CEA, and CEA/USD]
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 EIc);

It accelerates the recommendation process by up to 117x via

- Novel filtering heuristic: CEA
- Lightweight predictive models based on DTs
Code and data-sets are available online

github.com/pedrogbmendes/TrimTuner

Check our paper for more details
Questions?