Welcome

“Reproducing 150 Research Papers and Testing Them in the Real World: Challenges and Solutions”

Grigori Fursin

Twitter Hashtag: #ACMLearning

Tweet questions & comments to: @ACMeducation

Post-Talk Discourse: https://on.acm.org

Additional Info:

- Talk begins at the top of the hour and lasts 60 minutes
- For volume control, use your master volume controls and try headphones if it’s too low
- Submit questions at any time using the Q&A window (type your question and click “Submit”)”
- On the bottom panel you’ll find a number of widgets, including Twitter and sharing apps
- If you are experiencing any issues, try refreshing your browser or relaunching your session
- At the end of the presentation, please help us improve our talks by taking the experience survey that opens in a new window (you can also click on it at any time in the Resources window)
- This session is being recorded and will be archived for on-demand viewing. You’ll receive an email when it’s available.
Reproducing 150 Research Papers and Testing Them in the Real World: Challenges and Solutions

Speaker: Grigori Fursin

Moderator: Peter Mattson
ACM.org Highlights

For Scientists, Programmers, Designers, and Managers:

- **Learning Center** - [https://learning.acm.org](https://learning.acm.org)
  - View past TechTalks with top inventors, innovators, entrepreneurs, & award winners
  - Access to O’Reilly Learning Platform – technical books, video courses, & learning paths
  - Access to Skillsoft Training & ScienceDirect – vendor certification prep, technical books & courses
  - Listen to our ByteCast interviews with top computing luminaries and visionary researchers & entrepreneurs
  - Check out the new ACM Selects – themed shortlists of learning resources curated by subject matter experts

- **Ethical Responsibility** – [https://ethics.acm.org](https://ethics.acm.org)

**By the Numbers**

- 2,200,000+ content readers
- 1,800,000+ DL research citations
- $1,000,000 Turing Award prize
- 100,000+ global members
- 1200+ Fellows
- 700+ chapters globally
- 170+ yearly conferences globally
- 100+ yearly awards
- 70+ Turing Award Laureates

**Popular Publications & Research Papers**

- Communications of the ACM - [http://cacm.acm.org](http://cacm.acm.org)
- Queue Magazine - [http://queue.acm.org](http://queue.acm.org)
- Digital Library - [http://dl.acm.org](http://dl.acm.org)

**Major Conferences, Events, & Recognition**

- [https://www.acm.org/conferences](https://www.acm.org/conferences)
- [https://www.acm.org/chapters](https://www.acm.org/chapters)
- [https://awards.acm.org](https://awards.acm.org)
Welcome

“Reproducing 150 Research Papers and Testing Them in the Real World: Challenges and Solutions”

Grigori Fursin

Twitter Hashtag: #ACMLearning
Tweet questions & comments to: @ACMeducation
Post-Talk Discourse: https://on.acm.org

Additional Info:

- Talk begins at the top of the hour and lasts 60 minutes
- For volume control, use your master volume controls and try headphones if it’s too low
- Submit questions at any time using the Q&A window (type your question and click “Submit”)
- On the bottom panel you’ll find a number of widgets, including Twitter and sharing apps
- If you are experiencing any issues, try refreshing your browser or relaunching your session
- At the end of the presentation, please help us improve our talks by taking the experience survey that opens in a new window (you can also click on it at any time in the Resources window)
- This session is being recorded and will be archived for on-demand viewing. You’ll receive an email when it’s available.
Reproducing 150 Research Papers and Testing Them in the Real World

Grigori Fursin
Founder @ cKnowledge.io and cTuning.org
cKnowledge.io/@gfursin
Outline

• Personal motivation
• MILEPOST project: using ML to improve the system efficiency and reduce costs
• Reproducibility efforts and ACM
• Reproducing 150 research papers at CGO, PPoPP, ASPLOS, PACT and MLSys
• Bridging the growing gap between academic research and industry with DevOps and FAIR principles
• Validating research papers via open hackathons and tournaments
• Validating research papers in the real world
• Conclusions
1996: my first R&D project to test Hopefield Network in the real world

Research paper

Implement

Test in production

What could possibly go wrong?
1996: my first R&D project to test Hopefield Network in the real world

- **Simulate electronic circuits**
- **Implement model in C from scratch**
- **Implement training in C and bash**
- **Implement validation in C and bash**
- **Training too slow: parallelize**
- **Too many crashes: add fault-tolerance**
- **Keep track of experiments**
- **Get first results: not matching results from the paper**
- **Debug**
- **Model is finally working but too slow and inaccurate for production on real data**
- **Collect more realistic datasets**
- **Optimize matrix multiply in assembler**
- **Implement semiconductor NN**
- **Develop automation and visualization tools**
- **Test in production**

- **Prepare datasets**
- **Research paper**

- **Implement**

- **...**
1996-1999: Took a bit longer than expected 😊

- **Research paper**
- **Prepare datasets**
- **Simulate electronic circuits**
- **Implement model in C from scratch**
- **Implement training in C and bash**
- **Implement validation in C and bash**
- **Training too slow: parallelize**
- **Too many crashes: add fault-tolerance**
- **Develop automation and visualization tools**
- **Keep track of experiments**
- **Get first results: not matching results from the paper**
- **Debug**
- **Model is finally working but too slow and inaccurate for production on real data**
- **Optimize matrix multiply in assembler**
- **Collect more realistic datasets**
- **Hardware implementation: too expensive and inflexible**
- **Software implementation: too slow**
- **Need more automation, realistic data sets and further optimization**
- **Test in production**
- **Implement semiconductor NN**
From research to production: many challenges and tradeoffs

How to accelerate innovation?

• Need reference implementations (code)
• Need realistic benchmarks and data sets
• Need public SOTA results for validation
• Need a user-friendly access to HPC resources (cloud)
• Need better and simpler automation tools
• Need performance portability
• Need to improve system efficiency (math libraries, software and hardware)
• Need to balance speed, accuracy, size and all costs in the real world
The methodology to design computer systems has hardly changed in decades

**Traditional computer systems research and engineering**

- **Hardware development**
  - A limited number of benchmarks and data sets
  - Verification, validation and testing
  - Years

- **Compiler development**
  - Semi-manual tuning of optimization heuristics
  - Months, years

- **Production (the real world)**
  - Real workloads
  - Performance, energy, accuracy and costs rarely match official numbers.
  - Further tedious optimization required.
  - Years

- **Research**
  - A few benchmarks and data sets
  - Mostly focus on papers.
  - No time to test in the real world.
  - Lack of automation.
  - Years

**Challenges:**

- Engineers and researchers spend too much time on repetitive and ad-hoc tasks rather than innovation.
- Simply no time to validate new techniques on realistic benchmarks, data sets, compiler/OS versions, hardware.
- A growing gap between research and production; increasing time to market for new ideas.
- Waste of expensive resources
Too many design and optimization choices

GCC compiler (similar trends in LLVM)
How to accelerate design and optimization space exploration?

1 program, 1 data set, 1 platform, 1 compiler, 1000 random combinations of optimizations

[Diagram showing relationships between Algorithm, Program, Compiler, Binary and libraries, System state, Data set, Run-time, Software, Platform, and Results, with measurements (speed, size, accuracy, costs).]
What about using ML to improve system efficiency?

Similar programs, data sets and platforms may benefit from similar optimizations.
What about using ML to improve system efficiency?

Train model on N benchmarks

Algorithm
  Program
  Compiler
  Binary and libraries
System state
  Data set
Run-time
  Software
  Platform
Results

Measurements (speed, size, accuracy, costs)

Feature vector

Hardware counters (dynamic features)

Program embeddings (semantic features)

Similar programs, data sets and platforms

Minimize cost function

$$\sqrt{\frac{\sum_{i=1}^{N}(\text{Speedup}(\text{opt}(i)) - \text{Speedup}(\text{opt}'(i)))^2}{N}}$$

KNN, decision trees, SVM, DNN ...

Train model on N benchmarks

may benefit from similar optimizations
Predict optimizations based on semantic and dynamic features

- Algorithm
- Program
- Compiler
- Binary and libraries
- System state
- Data set
- Run-time
- Software
- Platform

Measurements (speed, size, accuracy, costs)

Results

Previously unseen program

Feature vector

Hardware counters (dynamic features)

Program embeddings (semantic features)

Accelerate auto-tuning and DSE (probabilistic focused search)

Suggest top-N optimizations

cKnowledge.io/rpi-ml-crowd-tuning
Predict optimizations based on semantic and dynamic features

Feature vector
- Hardware counters (dynamic features)
- Program embeddings (semantic features)

Previously unseen program

Test phase:
- Validate output
- Validate optimizations
- Engineer features
- Retrain model(s) or select a better one

Suggest top-N optimizations

cKnowledge.io/rpi-ml-crowd-tuning
MILEPOST project (2006-2009): test ML in a production compiler

- Algorithm
- Program
- Compiler
- Binary and libraries
- System state
- Data set
- Run-time
- Software
- Platform

Results

Measurements (speed, size, accuracy, costs)

Previous unseen program

Feature vector

Hardware counters (dynamic features)

Program embeddings (semantic features)

Trained model(s)

Test phase:
- Validate output
- Validate optimizations
- Engineer features
- Retrain model(s) or select a better one

Suggest top-N optimizations

cKnowledge.io/rpi-ml-crowd-tuning
80% of time spent on development, benchmarking and feature engineering

- Assembled cBench with miDataSets, KDataSets and cDataSets suitable for ML-based optimization
- Developed GCC plugin framework with semantic feature extractors (IBM+INRIA)
- Implemented multiple models
- Developed autotuning framework
- Developed experiment automation framework with hardware counter collection
- Developed training and validation framework
- Developed database for collaborative experiments

<table>
<thead>
<tr>
<th>ft1</th>
<th>Number of basic blocks in the method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ft19</td>
<td>Number of direct calls in the method</td>
</tr>
<tr>
<td>ft20</td>
<td>Number of conditional branches in the method</td>
</tr>
<tr>
<td>ft21</td>
<td>Number of assignment instructions in the method</td>
</tr>
<tr>
<td>ft22</td>
<td>Number of binary integer operations in the method</td>
</tr>
<tr>
<td>ft23</td>
<td>Number of binary floating point operations in the method</td>
</tr>
<tr>
<td>ft24</td>
<td>Number of instructions in the method</td>
</tr>
<tr>
<td>ft25</td>
<td>Average of number of instructions in basic blocks</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>ft29</td>
<td>Number of basic blocks with phi nodes in the interval [0, 3]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>ft54</td>
<td>Number of local variables that are pointers in the method</td>
</tr>
<tr>
<td>ft55</td>
<td>Number of static/extern variables that are pointers in the method</td>
</tr>
</tbody>
</table>

**Dynamic features**

cTuning.org/wiki/index.php/CTools:MilepostGCC:StaticFeatures:MILEPOST_V2.1

CGO’06, CGO’07, IJPP’10
Promising research results on 22 benchmarks and 2 platforms

“MILEPOST GCC: machine learning enabled self-tuning compiler”, GCC Summit’09 and IJPP’11

“Evaluating iterative optimization across 1000 data sets”, PLDI’10

“Rapidly Selecting Good Compiler Optimizations using Performance Counters”, CGO 2007
(ACM CGO’17 test of time award)

How to test it in the real world?

Top-3 optimization prediction accuracy: ~87%
cTuning.org (2009): collaborative platform to train ML compiler

An experimental toolset and repository to perform ML-based optimizations with the help of the community

Released all code and data (open-source)

Benchmarks and data sets

Feature extractor plugin for GCC

Collective tuning framework

ML training/validation framework

Communication with cTuning.org

What could possibly go wrong?

cTuning.org/wiki

github.com/ctuning/reproduce-milepost-project

SOTA dashboards

Top optimizations

Top models

Mediawiki

Web API
cTuning.org (2009): collaborative platform to train ML compiler

An experimental toolset and repository to perform ML-based optimizations with the help of the community

- Released all code and data (open-source)
- Benchmarks and data sets
- Feature extractor plugin for GCC
- Collective tuning framework
- ML training/validation framework
- Communication with cTuning.org

- Difficult to adapt to continuously evolving software, hardware, data and models (lack of portability)
- Difficult to reproduce empirical results
- Difficult to add new benchmarks, data sets, compilers and tools (lack of customization)
- Need more representative benchmarks, data sets and features (lack of diverse data sets) - optimization prediction accuracy dropped on new programs, data sets, platforms and compilers
- Published results are quickly outdated
- Difficult to reproduce and compare other papers (no shared code and data; lack of a common methodology; lack of apple-to-apple comparison)

Reproducibility crisis with combined issues from two domains: ML and Systems
2010-2014: Reproducibility studies and initiatives

**reproducibility.cs.arizona.edu**
A comprehensive study of ~600 papers to examine if related code was shared and can be built (weak reproducibility).

**evaluate.inf.usi.ch/artifacts** and **artifact-eval.org**
The original and successful introduction of the artifact evaluation process at ACM conferences (strong reproducibility).
Artifacts are evaluated after papers are accepted and before the camera-ready deadline.
Paper receive the reproducibility badge only if the related artifact is consistent, complete, well documented and easy to reuse.

**cTuning.org/ae**
Cooperative process between authors and evaluators to help pass artifact evaluation.
Learn how to unify and automate this process particularly for very complex artifacts.
Learn how to make it easier to test research techniques in the real world with the latest software, hardware and data.
Encourage code and data sharing and test for artifact functionality, reproducibility and reusability separately.

Bruce R. Childers, Grigori Fursin, Shriram Krishnamurthi, Andreas Zeller:
2015-now: ACM and NeurIPS/ICML initiatives


- Artifacts and reproducibility badges in the ACM Digital Library [dl.acm.org/doi/proceedings/10.1145/3229762](dl.acm.org/doi/proceedings/10.1145/3229762) [dl.acm.org/search/advanced](dl.acm.org/search/advanced)


- ACM Emerging Interest Group on Reproducibility [reproducibility.acm.org](reproducibility.acm.org)

- Reproducibility initiative at NeurIPS’19 [nips.cc/Conferences/2019/CallForPapers](nips.cc/Conferences/2019/CallForPapers)

- PapersWithCode tips for publishing research code [github.com/paperswithcode/releasing-research-code](github.com/paperswithcode/releasing-research-code)

- NISO artifact badges [www.niso.org/publications/rp-31-2021-badging](www.niso.org/publications/rp-31-2021-badging)
2015: introduced unified appendix and reproducibility checklist

Up to 2 pages in all papers passing artifact evaluation

- Algorithm
- Program
- Compiler
- Binary and libraries
- System state
- Data set
- Run-time
- Software
- Platform
- Results

1. Abstract
2. Check-list
3. How to obtain?
4. Prepare software
5. Prepare hardware
6. Prepare data sets
7. Proprietary code and data
8. Installation
9. Experiment workflow
10. Evaluation and expected result
11. Notes

**Keywords**
- Algorithm
- Program
- Compilation
- Transformations
- Binary
- Data set
- Run-time environment
- Hardware
- Run-time state
- Execution
- Output
- Experiment workflow
- Publicly available?

My goal is to learn how to automate artifact evaluation

cTuning.org/ae/checklist.html
cKnowledge.io/reproduced-papers
dl.acm.org
Déjà vu: main challenges during Artifact Evaluation at ML and Systems conferences

- Artifacts and workflows are very complex and diverse
- No common formats and APIs for shared artifacts and workflows: reviewers spend most of their time understanding the structure of the project from ReadMe files, fixing hardwired paths, building and running code on their platforms, checking correctness, etc.
- Sharing code, data and Jupyter notebooks is not enough to reproduce results and test them in the real world: need to adapt to rapidly evolving systems and plug in other artifacts (programs, data sets, models, software and hardware).
- Containers become quickly outdated: they hide the dependency hell rather than solving it.
- Difficult to have fair “apple-to-apple” comparison of different research techniques: researchers tend to write their software and tools from scratch to collect, process and report results.
- Difficult to reuse shared code in production without common APIs, meta descriptions and portability: many research project die when PhD students graduate or key developers leave.
Started noticing some patterns across different projects

1. Abstract
2. Check-list
3. How to obtain?
4. Prepare software
5. Prepare hardware
6. Prepare data sets
7. Proprietary code and data
8. Installation
9. Experiment workflow
10. Evaluation and expected result
11. Notes

---

**Could be reused across projects**

**Common (reusable) actions**

**Common objects**

**Common meta**

---

![Diagram](image-corner-detection)_Ad-hoc scripts to compile and run a program or a benchmark

- image corner detection
- matmul OpenCL
- data compression
- object detection CUDA

- GCC V9.3
- LLVM V11.1
- Intel Compilers 2021

- image-jpeg-0001
- bzip2-0006
- txt-0012
- video-raw-1280x1024

- cvs speedups
- txt hardware counters
- xls table with graphs

- Have some common info: which datasets can use, how to compile, CMD, ...
- Have some common info: configuration, compilation, linking and optimization flags
- Have some common info: filename, size, width, height, colors, ...
- Have some common info: features, characteristics, optimizations
Can convert ad-hoc scripts into micro-services with a unified API

Common actions as micro-services with a unified API, CLI and I/O

Python module “program” with functions:
- compile and run

Python module “soft” with function:
- detect

Python module “package” with function:
- download

Python module “experiment” with function:
- add, plot, replay

Common objects

- image corner detection
- matmul OpenCL
- data compression
- object detection CUDA
- GCC V9.3
- LLVM V11.1
- Intel Compilers 2021
- image-jpeg-0001
- bzip2-0006
- txt-0012
- video-raw-1280x1024
- cvs speedups
- txt hardware counters
- xls table with graphs

Common meta

Have some common info:
- configuration, compilation, linking and optimization flags
- filename, size, width, height, colors, ...
- features, characteristics, optimizations

Common actions

- JSON input
- CLI
- Web service

- Ad-hoc scripts to compile and run a program or a benchmark
- Ad-hoc scripts to install packages or set up environment for code and data deps on a given platform
- Ad-hoc dirs for data sets with some ad-hoc scripts to find them, extract features, etc
- Ad-hoc dirs and scripts to record and analyze experiments

Have some common info:
- which datasets can use, how to compile, CMD, ...
- configuration, compilation, linking and optimization flags
- features, characteristics, optimizations
Can add UID, meta description and provenance info to all objects

Common actions as micro-services with a unified API, CLI and I/O

Findable and reusable plug&play objects

Unified JSON/YAML
Collective Knowledge Framework:

github.com/ctuning/ck

cKnowledge.org

A simple Python framework with a unified CLI/API and minimal dependencies to manage research projects as a database of reusable components.

Collective Knowledge Framework:

github.com/ctuning/ck
cKnowledge.org

A simple Python framework with a unified CLI/API and minimal dependencies to manage research projects as a database of reusable components.

Can share research projects as a database of reusable components

pip install ck
ck pull repo --url=https://github.com/ctuning/ai
ck ls program:*automotive*
ck search dataset --tags=image,jpeg
ck find package:imagenet-2012-train
ck find 1dc07ee0f4742028:b4f26f2ca41539d9
/home/gfursin/CK/ai/package/imagenet-2012-train
ck search package --tags=pytorch
ck add dataset:test
ck rm experiment:*
Started automating and sharing the most common actions from reproduced papers

```python
# Simple Python API with dict/JSON/YAML input/output
import ck.kernel as ck
input = {'action': 'detect', 'module_uoa': 'platform'}
output = ck.access(input)
if output['return'] > 0: ck.err(output)
print(json.dumps(output, indent=2))
```

1) Describe different operating systems
   - `ck pull repo:ck-env`
   - `ck ls os`
   - `ck load os:linux-64 --min`

2) Detect and unify information about platforms
   - `ck detect platform --help`
   - `ck detect platform --out=json`
   - `ck load os:android29-arm64 --min`

3) Detect installed software (code, data, models, scripts)
   - `ck search soft --tags=compiler`
   - `ck detect soft:compiler.llvm`
   - `ck show env --tags=compiler`

4) Install missing packages (code, datasets, models, scripts)
   - `ck search package --tags=dataset,imagenet`
   - `ck install package --tags=dataset,imagenet,2012,min`
   - `ck virtual env --tags=dataset,imagenet`

Record experiments; perform stat. analysis; plot results; validate outputs; generate papers, etc ...

cKnowledge.io/modules  cKnowlege.io/browse
Enabled portable program workflows that can adapt to continuously changing SW/HW

**ck compile** program:conv-armcl-opencl-uint8

**ck run** program:conv-armcl-opencl-uint8 --env. CK_SEED=123

**CK module program** (Python + meta JSON/YAML to implement portable workflows)
- **Query CK DB** to find conv-armcl-opencl-uint8
- **Load** conv-armcl-opencl-uint8 meta.json
- Resolve dependencies and prepare env
  - **Query CK DB** soft recipes by tags
  - Detect soft (compilers, frameworks, libraries, data sets, models)
  - **Query CK DB** package to install missing packages
- Compile program
- Run pro-processing
- Run program
- Run post-processing and unify metrics for apple to apple comparison
- **Query CK DB** program.output to validate output for correctness
- **Record** experiment with all provenance data to CK DB experiment

**CK program entry conv-armcl-opencl-uint8**
- **File conv.cpp** with algorithm
- **File meta.json** describing SW/HW deps, configs and compile/run info

```json
deps: {
    "compiler": {"name": "C++ compiler","tags": "compiler,lang-cpp"},
    "lib-nntest": {"name": "NNTest library","tags": "lib,nntest"},
    "library": {"name": "ARM Compute Library (OpenCL, uint8 )", "or_tags": "armcl;vdefault;vconv-uint8", "tags": "lib,arm-compute-library,vopencl"},
    "opencl": {"name": "OpenCL Library","tags": "lib,opencl"}
},
run_cmds: {
    "dataset_tags": ["dataset", "nntest", "tensor-conv"],
    "pre_process_via_ck": {
        "module_uoa": "script","script_name": "process","data_uoa": "3b59f57d587e82f6"},
    "run_cmd_main": "$#BIN_FILE#$",
    "post_process_cmds": [ ... ],
    "run_correctness_output_files": ["tmp-ck-output.json"],
},
run_vars: {"CK_ABS_DIFF_THRESHOLD": 1, "CK_IN_SHAPE_N": 1,
            "CK_OUT_RAW_DATA": "tmp-ck-output.bin",
            "CK_OUT_RAW_DATA_BINARY_FORMAT": "B", "CK_SEED": 42 },
"tags": ["nntest", "armcl", "conv","uint32", "vopencl"]
```

cKnowledge.io/c/module/program

cKnowledge.io/c/program/conv-armcl-opencl-uint8
We can plug in and reuse compatible components from different projects now!

GitHub repo for research paper\textsubscript{1} as a CK DB
- CK modules, programs, packages, soft, experiments

Docker image for research paper\textsubscript{2} as a CK DB
- CK modules, programs, packages, soft, experiments

ZIP file with datasets as a CK DB
- CK packages and data sets

GitHub private company repo as a CK DB
- CK modules, programs, packages, soft, experiments

Colab/Jupyter notebook querying CK DB

Auto-generated paper from CK DB

“Pull, plug & play” multiple repositories in the CK format

Always use the same CK API/CLI

$ \texttt{ck pull repo}\textsubscript{-url=\url{https://github.com/ctuning/ck-crowdtuning}}$

$ \texttt{ck ls program}$

$ \texttt{ck ls dataset}$

$ \texttt{ck load program:cbench-automotive-susan --min}$

$ \texttt{ck compile program:cbench-automotive-susan --fast}$

$ \texttt{ck run program:cbench-automotive-susan}$

$ \texttt{ck autotune program:cbench-automotive-susan}$

$ \texttt{ck crowdtune program:cbench-automotive-susan}$

$ \texttt{ck replay experiment}$

Automatically adapt to user environments
We can plug in and reuse compatible components from different projects now!

GitHub repo for research paper₁ as a CK DB
- CK modules, programs, packages, soft, experiments

Docker image for research paper₂ as a CK DB
- CK modules, programs, packages, soft, experiments

ZIP file with datasets as a CK DB
- CK packages and data sets

GitHub private company repo as a CK DB
- CK modules, programs, soft, experiments

Colab/Jupyter notebook querying CK DB

Auto-generated paper from CK DB

“Pull, plug & play” multiple repositories in the CK format

$ ck pull repo
$ ck ls program
$ ck ls dataset
$ ck load program:cbench
$ ck compile program:cbench
$ ck run program:cbench
$ ck autotune program:cbench
$ ck crowdtune program:cbench
$ ck replay experiment

Always use the same CK API/CLI
Automatically adapt to user environments

Bringing in DevOps and FAIR principles (extended to code):
en.wikipedia.org/wiki/FAIR_data

Findable
Accessible
Interoperable
Reusable
+ Portable
cKnowledge.io platform

Share portable workflows, adaptive containers, automation actions and plug&play components along with research papers:

cKnowledge.io/browse
cKnowledge.io/reusable-research

Enable “live” research papers that can be validated and improved by the community across diverse models, data sets, software and hardware:
cKnowledge.io/reproduced-results

Quickly prototype and test ideas on any tech. stack

Initialize  Build  Run  Validate
Collaboratively expose optimizations, characteristics and features in different components

Program

User system

TensorFlow

PyTorch

Gradually expose all available algorithmic, design and optimization choices

Expose additional information

Requirements

System state

Features

Continuously observe behavior (characteristics); check for normality

Predict optimizations

If unexpected behavior, share with the community, improve models, expose optimizations, engineer features

Model and learn program and system behavior

\[ \overline{b} = B(\overline{c}, \overline{f}, \overline{s}) \]
Enabled universal crowd-tuning and ML workflow connected with SOTA dashboards

Connected CK components into customizable auto-tuner for the whole ML/SW/HW stack (co-design)

Choose exploration strategy
Explore exposed design and optimization choices (auto-tuning / DSE)
Compile source code
Run code
Test behavior normality
Pareto filter
Modeling and prediction
Complexity reduction

Proof-of-concept of live papers validated by the community

GitHub
GitLab
ACM DL
ArXiv
...

Raspberry Pi

cKnowledge.io/rpi-ml-crowd-tuning

github.com/ctuning/reproduce-milepost-project
cKnowledge.org/repo-beta
Apply Machine Learning (try different ML algorithms, features and hyperparameters)

Training set: distinct combination of compiler optimizations (clusters)

- \( \vec{c} \) (choices)

- \( \vec{f} \) (features)
  - MILEPOST GCC features
  - hardware counters

Optimization cluster

Unseen program

Test different features

Apply different models (KNN, SVM, DNN ...)

Prediction

Optimization cluster

<table>
<thead>
<tr>
<th>Number of code and dataset samples</th>
<th>Prediction accuracy using optimized SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>87%</td>
</tr>
</tbody>
</table>

Current limited studies
Apply Machine Learning (try different ML algorithms, features and hyperparameters)

- Training set: distinct combination of compiler optimizations (clusters)

- \( \bar{c} \) (choices)
- \( \bar{f} \) (features)
  - MILEPOST GCC features, hardware counters

- Optimization cluster
- Unseen program

- Prediction accuracy
<table>
<thead>
<tr>
<th>Number of code and dataset samples</th>
<th>Prediction accuracy using optimized SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>87%</td>
</tr>
<tr>
<td>285</td>
<td>56%</td>
</tr>
</tbody>
</table>

Test different features
Apply different models (KNN, SVM, DNN ...)

Optimization cluster
Expose unexpected behaviour and learn features with the community

<table>
<thead>
<tr>
<th>Experiment</th>
<th>-O3</th>
<th>-O3 -fno-if-conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared experiment&lt;sub&gt;1&lt;/sub&gt;</td>
<td>*matrix_ptr2++ = (temp1 &gt; T) ? 255 : 0;</td>
<td>reference execution time</td>
</tr>
<tr>
<td>Shared experiment&lt;sub&gt;2&lt;/sub&gt;</td>
<td>no change</td>
<td></td>
</tr>
</tbody>
</table>
Expose unexpected behaviour and learn features with the community

<table>
<thead>
<tr>
<th>Experiment</th>
<th>-O3</th>
<th>-O3 -fno-if-conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared experiment&lt;sub&gt;1&lt;/sub&gt;</td>
<td>*matrix_ptr2++ = (temp1 &gt; T) ? 255 : 0;</td>
<td>reference execution time</td>
</tr>
<tr>
<td>Shared experiment&lt;sub&gt;2&lt;/sub&gt;</td>
<td>no change</td>
<td>+17.3% improvement</td>
</tr>
</tbody>
</table>
Expose unexpected behaviour and learn features with the community

<table>
<thead>
<tr>
<th>Experiment</th>
<th>-O3</th>
<th>-O3 -fno-if-conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared experiment \textsubscript{1}</td>
<td>reference execution time</td>
<td>no change</td>
</tr>
<tr>
<td>Monitored during \textit{day}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared experiment \textsubscript{2}</td>
<td>no change</td>
<td>+17.3% improvement</td>
</tr>
<tr>
<td>Monitored during \textit{night}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Feature “\texttt{TIME\textunderscore OF\textunderscore THE\textunderscore DAY}” related to algorithm, data set and run-time
Can’t be found by ML - simply does not exist in the system!

if get\_feature(\texttt{TIME\textunderscore OF\textunderscore THE\textunderscore DAY})==\texttt{NIGHT}    bw\_filter\_codelet\_day(buffers);
else                                          bw\_filter\_codelet\_night(buffers);
Started converting artifacts from deep learning papers to CK

Assemble portable workflows with plug&play components as LEGO bricks

cKnowledge.io/browse  github.com/ctuning/ai
**ACM ReQuEST: reproducible ML/SW/HW co-design tournaments**

Open competitions to co-design Pareto-efficient AI/SW/HW stacks for real-world user tasks across diverse models, data sets and platforms, convert them to the CK format and reproduce results by the community!

[cKnowledge.io/event/repro-request-asplos2018](https://cKnowledge.io/event/repro-request-asplos2018)

1st competition at ACM ASPLOS’18: 8 intentions to submit and 5 submissions

<table>
<thead>
<tr>
<th>CK workflow$_1$</th>
<th>CK workflow$_2$</th>
<th>CK workflow$_3$</th>
<th>CK workflow$_4$</th>
<th>CK workflow$_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Caffe, BVLC Caffe</td>
<td>TensorFlow; Keras; Avro</td>
<td>MXNet; NNVM/TVM</td>
<td>MXNet; NNVM/TVM</td>
<td></td>
</tr>
<tr>
<td>ResNet-50, SSD Inception-v3; 32-bit, 8-bit</td>
<td>AlexNet; VGG16</td>
<td>OpenBLAS vs ArmCL</td>
<td>ResNet-50, VGG16, MobileNet-v1-1.0-224</td>
<td></td>
</tr>
<tr>
<td>Intel C++ Compiler 17.0.5 20170817</td>
<td>GCC</td>
<td>GCC; LLVM; CUDA</td>
<td>GCC; LLVM</td>
<td></td>
</tr>
<tr>
<td>AWS + Intel Xeon® Platinum 8124M</td>
<td>NVIDIA Jetson TX2; 12x Raspberry Pi 3</td>
<td>Firefly-RK3399</td>
<td>Pynq (Xilinx FPGA)</td>
<td></td>
</tr>
<tr>
<td>ArmCL (OpenCL) 18.01 vs 18.03</td>
<td></td>
<td></td>
<td>GCC; LLVM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NVIDIA Jetson TX2; 12x Raspberry Pi 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Firefly-RK3399</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pynq (Xilinx FPGA)</td>
</tr>
</tbody>
</table>

Open reviewing at [https://github.com/ctuning/ck-request-asplos18-results](https://github.com/ctuning/ck-request-asplos18-results) via GitHub issues.
Published validated papers with reusable workflows in the ACM DL

dl.acm.org/doi/proceedings/10.1145/3229762
Shared public CK dashboards connected with research papers

Multi-objective results for all AI/SW/HW stacks are presented on a live scoreboard and become available for public comparison and further customization, optimization and reuse!

cKnowledge.io/c/result/pareto-efficient-ai-co-design-tournament-request-acm-asplos-2018
cKnowledge.io/results

CK workflow with validated results
AWS with c5.18xlarge instance; Intel® Xeon® Platinum 8124M

From the authors: “The 8-bit optimized model is automatically generated with a calibration process from FP32 model without the need of fine-tuning or retraining. We show that the inference throughput and latency with ResNet-50, Inception-v3 and SSD are improved by 1.38X-2.9X and 1.35X-3X respectively with negligible accuracy loss from IntelCaffe FP32 baseline and by 56X-75X and 26X-37X from BVLC Caffe.”

https://github.com/ctuning/ck-request-asplos18-caffe-intel
DevOps and FAIR principles made it easier to adopt research in production.

Colleagues from Amazon tested and reused REQUEST workflows, ported them to the Amazon cloud and used CK API and JSON meta to connect them with Amazon SageMaker.


CK can also automatically generate a Docker image with CK workflow.

CK workflows and live papers enable a community effort to unify, automate, systematize and crowdsource development, optimization and comparison of efficient software/hardware stacks for emerging AI/ML workloads.
Quantum ML hackathons using CK workflows and dashboards

cKnowledge.org/quantum

Results from the Quantum Machine Learning Hackathon in Paris

<table>
<thead>
<tr>
<th>#</th>
<th>Problem index</th>
<th>Timestamp (UTC)</th>
<th>Team name</th>
<th>Training time (sec)</th>
<th>Training accuracy</th>
<th>Test accuracy</th>
<th>Solution's rank</th>
<th>Source code</th>
<th>Quantum circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>Sun Jan 27 12:19:42 2019</td>
<td>Optimize, adapt, overcome</td>
<td>47.20</td>
<td>100.0</td>
<td>100.0</td>
<td>1</td>
<td>continuous_solver</td>
<td>Show circuit</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Sun Jan 27 12:52:40 2019</td>
<td>provision.io</td>
<td>80.68</td>
<td>100.0</td>
<td>100.0</td>
<td>2</td>
<td>continuous_solver</td>
<td>Show circuit</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>Sun Jan 27 13:21:31 2019</td>
<td>rebecca</td>
<td>171.54</td>
<td>100.0</td>
<td>100.0</td>
<td>3</td>
<td>continuous_solver</td>
<td>Show circuit</td>
</tr>
</tbody>
</table>

cKnowledge.io/reproduced-results
Quantum ML hackathons using CK workflows and dashboards

cKnowledge.org/quantum

The most efficient design

cKnowledge.io/reproduced-results
Use Android app to crowdsource ML systems benchmarking and extend data sets

The number of distinct participated platforms: 800+
The number of distinct CPUs: 260+
The number of distinct GPUs: 110+
The number of distinct OS: 280+
Power range: 1-10W

Volunteers help to validate research ideas similar to SETI@HOME

Collect more data sets from users for misclassifications to build an open and continuously updated training set

cKnowledge.org/android-demo.html   cKnowledge.org/repo-beta   cKnowledge.io/negative-results
Crowd-tune whole ML/SW/HW stacks


- Parameterised CNN family using depthwise separable convolutions.
- Channel multiplier: 1.00, 0.75, 0.50, 0.25 - marker shape (see below).
- Input image resolution: 224, 192, 160, 128 - marker size.

Multi-objective auto-tuning (speed, accuracy, size, energy, cost)

cKnowledge.io/crowdsource-ml-sw-hw-co-design

cKnowledge.io/c/lib/9a927e4ce9be41b4

cKnowledge.io/reproduced-results

cKnowledge.io/reproduced-papers
Universal CK workflow to compile, run, auto-tune and model programs across diverse data sets, libraries and hardware

Reuse CK auto-tuner to generate ML-based adaptive libraries for CNNs

- Generate ~1K inputs with random sizes or extract from AlexNet, GoogLeNet, SqueezDet
- Extract features
- Auto-tune GEMM for each input
- Run off-the shelf conv implementations
- Predict optimization parameters at run-time
- Test overall inference speed using Alexnet, FCN-16s, GoogLeNet, InceptionV3, Mobilenets, ResNet-50, VGG-16
- Train and optimize different ML models in terms of RT prediction speed (complexity) vs inference speed: SVM, KNN, decision trees (Random Forest, Gradient Tree Boosting, Naive Bayesian classifier, Multi Layer Perceptron)

“On the Anatomy of Predictive Models for Accelerating GPU Convolution Kernels and Beyond”, ACM TACO, January 2021
doi.org/10.1145/3434402 cKnowledge.io/nn-components
Collaboratively benchmark and optimize Deep Learning Implementations

Presentation from General Motors how to reuse CK workflows to co-design efficient ML/SW/HW stacks for self-driving cars

youtu.be/1ldgVZ64hEi
Support MLPerf and MLCommons

MLPerf
A broad ML benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms.
mlperf.org

MLCommons is an open engineering consortium with a mission to accelerate machine learning innovation, raise all boats and increase its positive impact on society.
mlcommons.org/en/news/mlcommons-launch
mlcommons.github.io/mlcube

Reusing CK AI workflows to automate and optimize ML inference submissions for edge devices
cknowledge.io/solutions  cknowledge.io/nn-components
cknowledge.io/adaptive-containers  cknowledge.io/reproduced-results
Conclusions: bridging the growing gap between research and production

- 50K papers every year
- 30K code repositories
- 3K Dashboards
- 3K Datasets
- Numerous Colab/Jupyter Notebooks

Continuously changing SW/HW/ML from different vendors

Numerous tools and libraries

- Numerous containers

cKnowledge.io platform prototype
to deal with this Cambrian ML/SW/HW explosion:

*Share portable and reusable workflow templates, plug&play components, automations and SOTA dashboards from reproduced papers*

Conflicting design and optimization goals in the real world:
speed vs accuracy vs energy vs size vs costs ...
Conclusions

• Sharing code, data, Docker containers and Colab/Jupyter notebooks is not enough! Invest into simple, automated, sustainable, reusable and portable software to make it easier to use it in the real world:
  • No hardwired paths – use CK-like database structure for projects
  • Simple APIs and meta descriptions for shared artifacts (collective benchmarks and data sets)
  • Reusable components (code, data and models) to avoid too much legacy code and technical debt
  • Portable workflows to adapt to continuously changing software, hardware, models and data sets
  • Apple-to-apple comparison of results

• Collective Knowledge framework and cKnowledge.io platform is a proof-of-concept that it is possible to address above challenges based on DevOps and FAIR principles for code, data and models and with the help of the community. But still a lot to be done!
  • Share novel techniques as portable, customizable, reproducible, reusable and production-ready workflows along with published papers that can be quickly validated in the real world and adopted in production.
  • Connect and support existing projects, tools, data sets, models and platforms
  • Support collaborative and reproducible R&D, improve the efficiency of ML Systems, accelerate innovation and enable open science!
Acknowledgments


Artifact evaluation committee: cTuning.org/ae/committee.html

ACM REQUEST committee and advisory board: cKnowledge.io/c/event/repro-request-asplos2018

ACM taskforce and EIG on reproducibility: www.acm.org/publications/task-force-on-data-software-and-reproducibility

Co-organizers of CK-based hackathons and tournaments: cKnowledge.io/events

CK collaborators: cKnowledge.io/partners

Microsoft for Azure sponsorship
Thank you!

cKnowledge.io/@gfursin

Recent papers


• “A Collective Knowledge workflow for collaborative research into multi-objective autotuning and machine learning techniques” Live paper: cKnowledge.io/rpi-ml-crowd-tuning
The Learning Continues...

TechTalk Discourse Forum: https://on.acm.org
TechTalk Inquiries: learning@acm.org
TechTalk Archives: https://learning.acm.org/techtalks
Learning Center: https://learning.acm.org
ACM Selects: https://selects.acm.org/
ACM ByteCast: https://learning.acm.org/bytecast/
Professional Ethics: https://ethics.acm.org
Queue Magazine: https://queue.acm.org