



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>
 - 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>

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>
- Queue Magazine - <http://queue.acm.org>
- Digital Library - <http://dl.acm.org>

Major Conferences, Events, & Recognition

- <https://www.acm.org/conferences>
- <https://www.acm.org/chapters>
- <https://awards.acm.org>



Welcome

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

Grigori Fursin

Twitter Hashtag: [#ACMLearning](#)

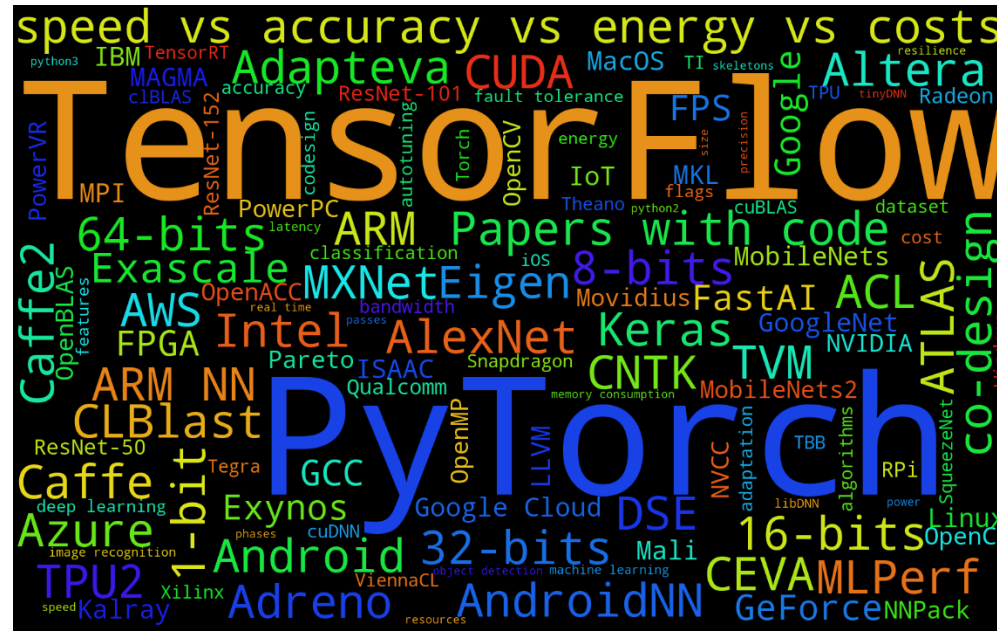
Tweet questions & comments to: [@ACMeducation](#)

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

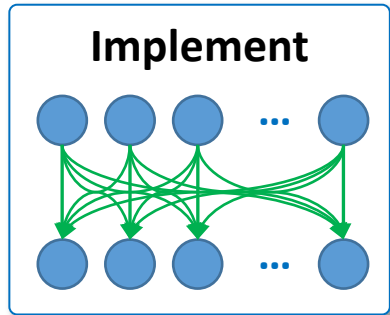


Outline

- 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 Hopfield Network in the real world

Research paper



Test in production

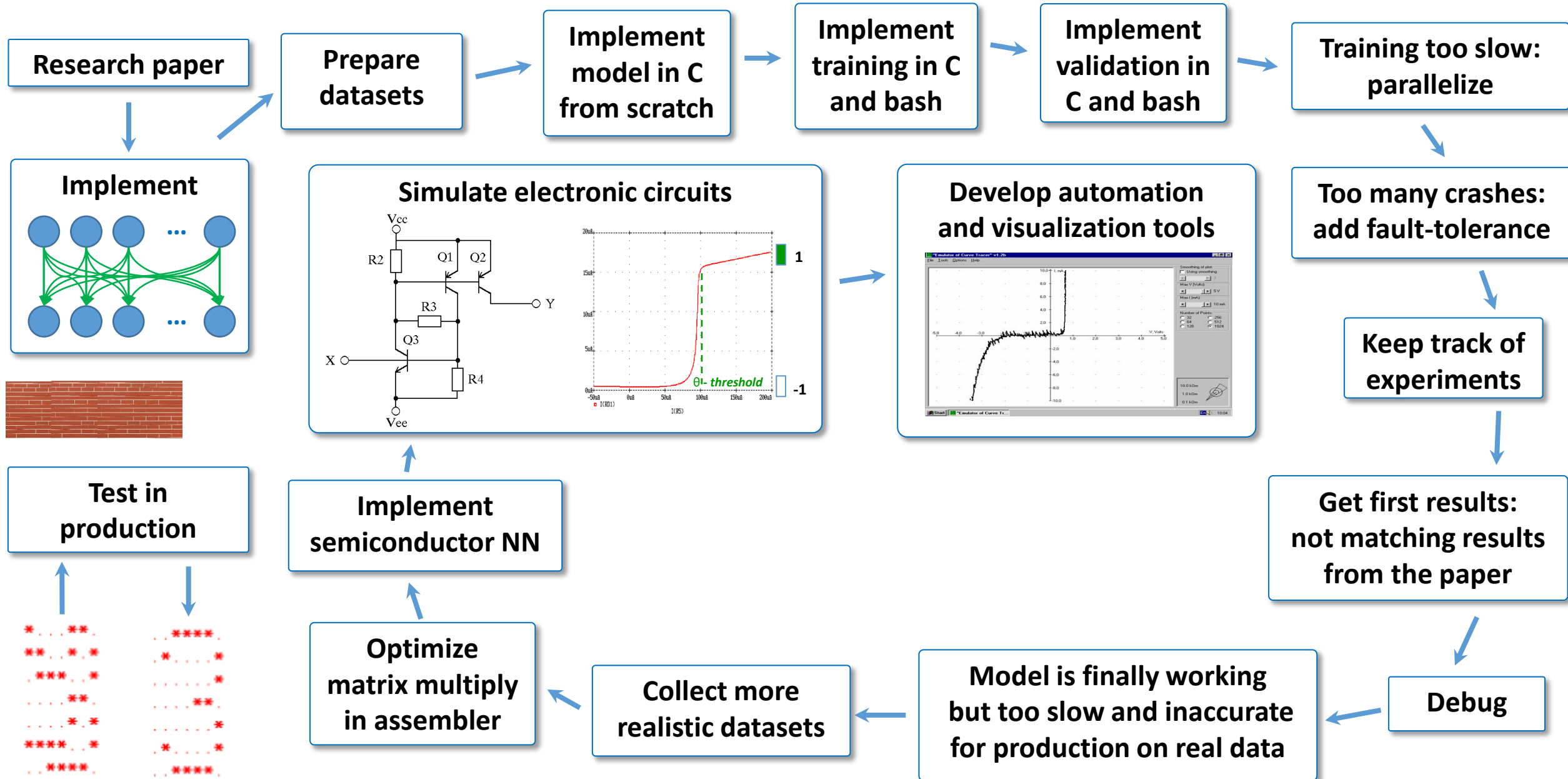


* . . . * *
* * . . * *
. * * * . *
. . . . * *
. . . . * *
* * * * * *
. . * * * *
. . * * * *

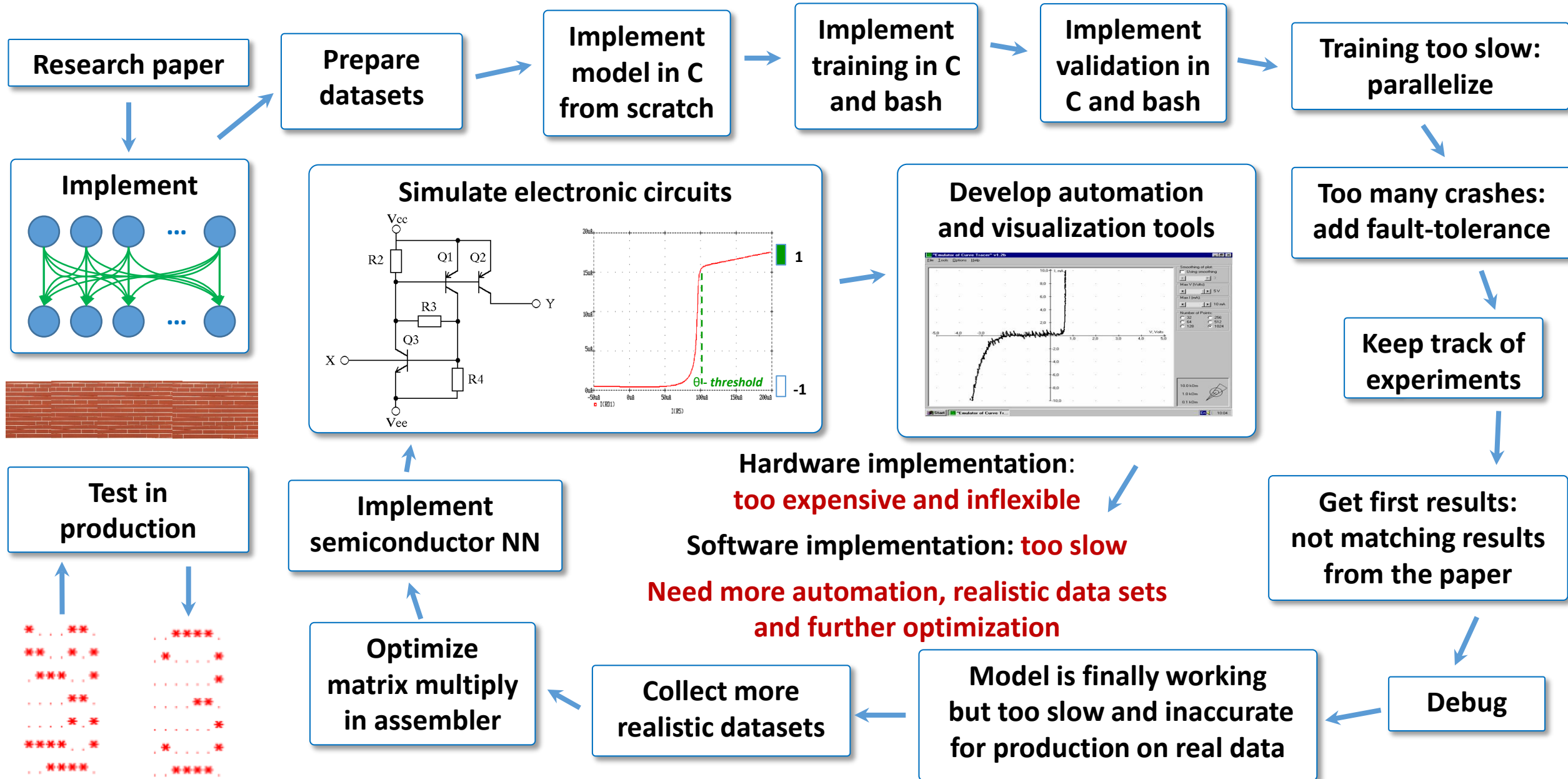
. . . * * * *
* * * * * *
. *
. *
. *
* * * * * *
. . * * * *
. . * * * *

What could possibly go wrong?

1996: my first R&D project to test Hopfield Network in the real world



1996-1999: Took a bit longer than expected 😊



From research to production: many challenges and tradeoffs

Research papers



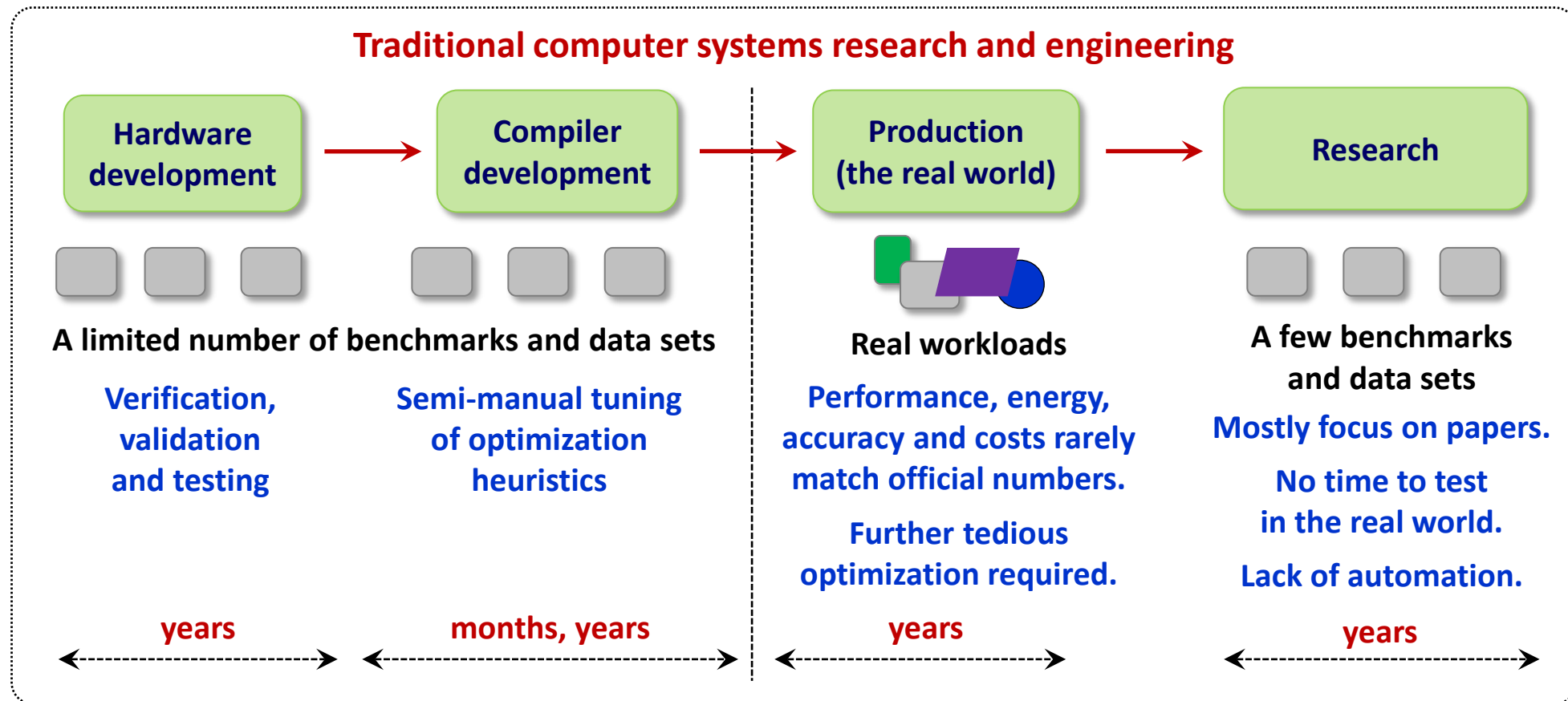
How to accelerate innovation?

- 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



Production

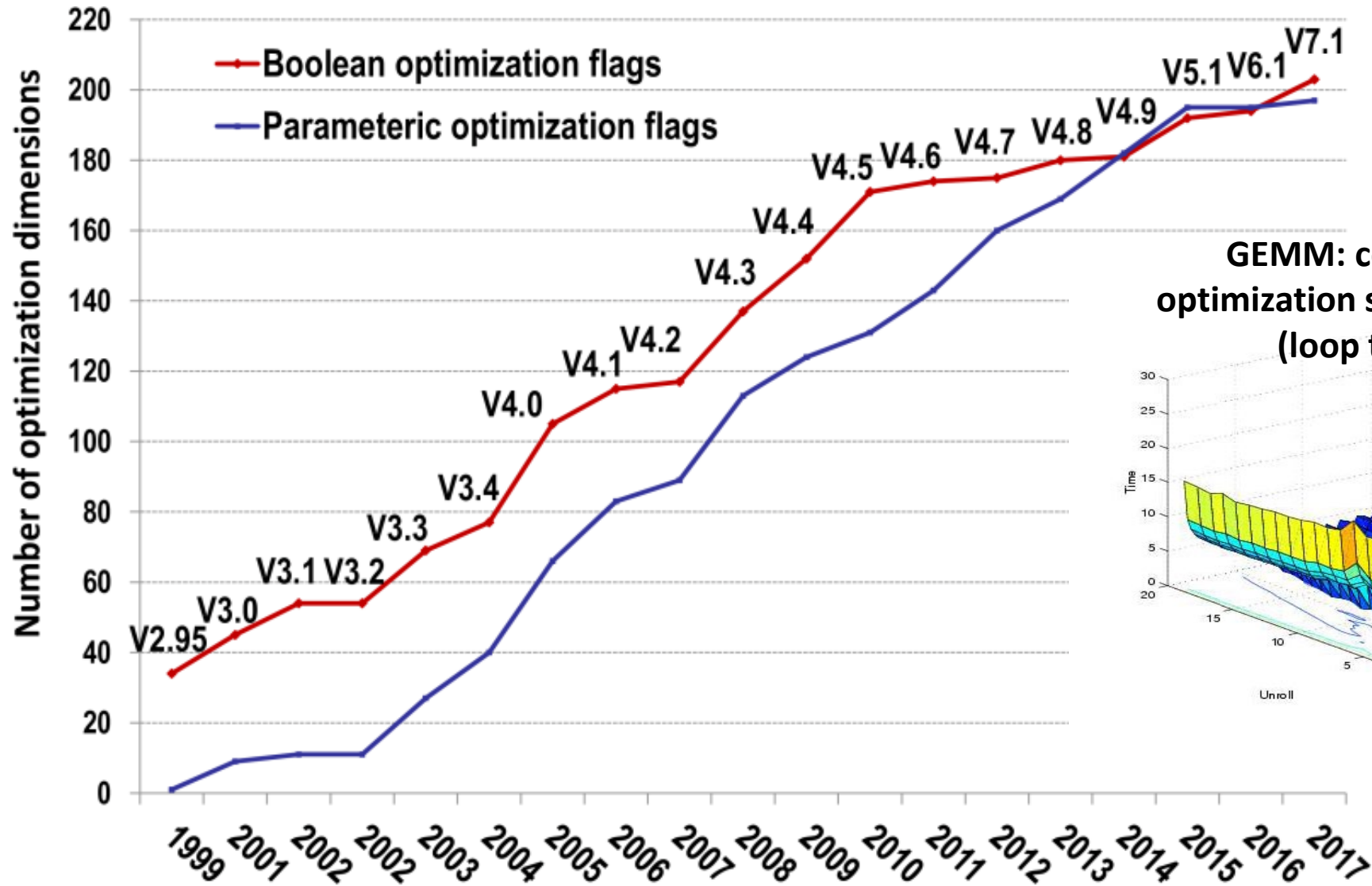
The methodology to design computer systems has hardly changed in decades



Challenges:

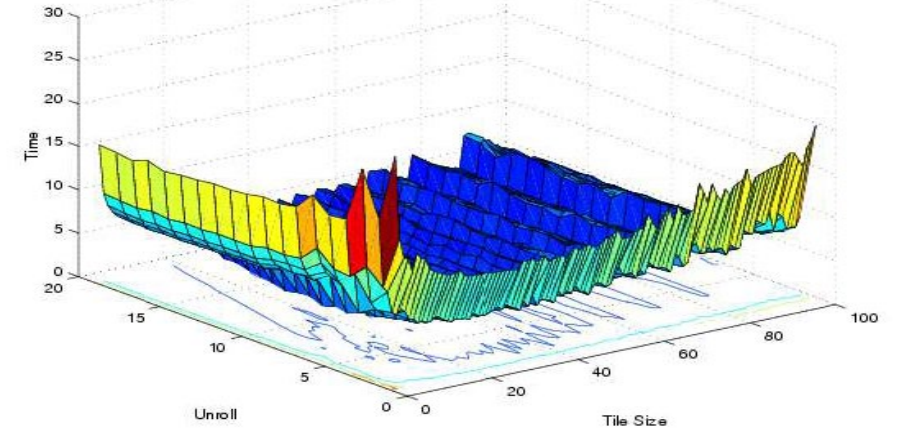
- Engineers and researchers spend too much time on repetitive and ad-hoc tasks rather than innovation.
- Simply no time to validate new techniques on realistic benchmarks, data sets, compiler/OS versions, hardware.
- A growing gap between research and production; increasing time to market for new ideas.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

Too many design and optimization choices

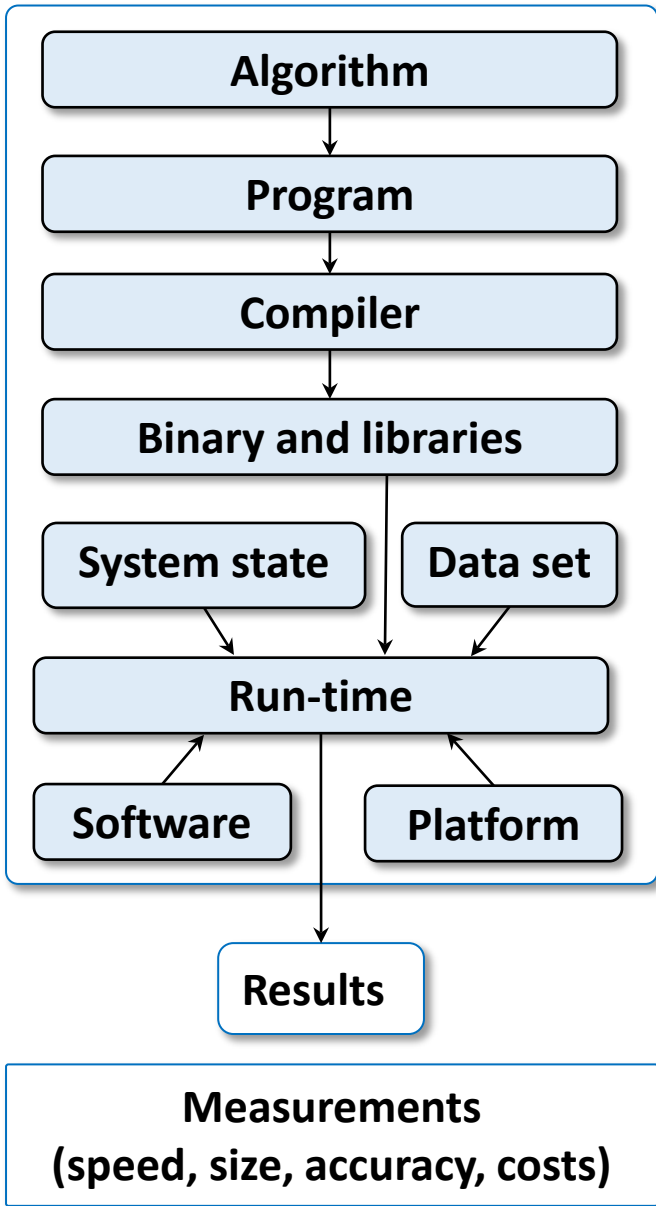


GCC compiler (similar trends in LLVM)

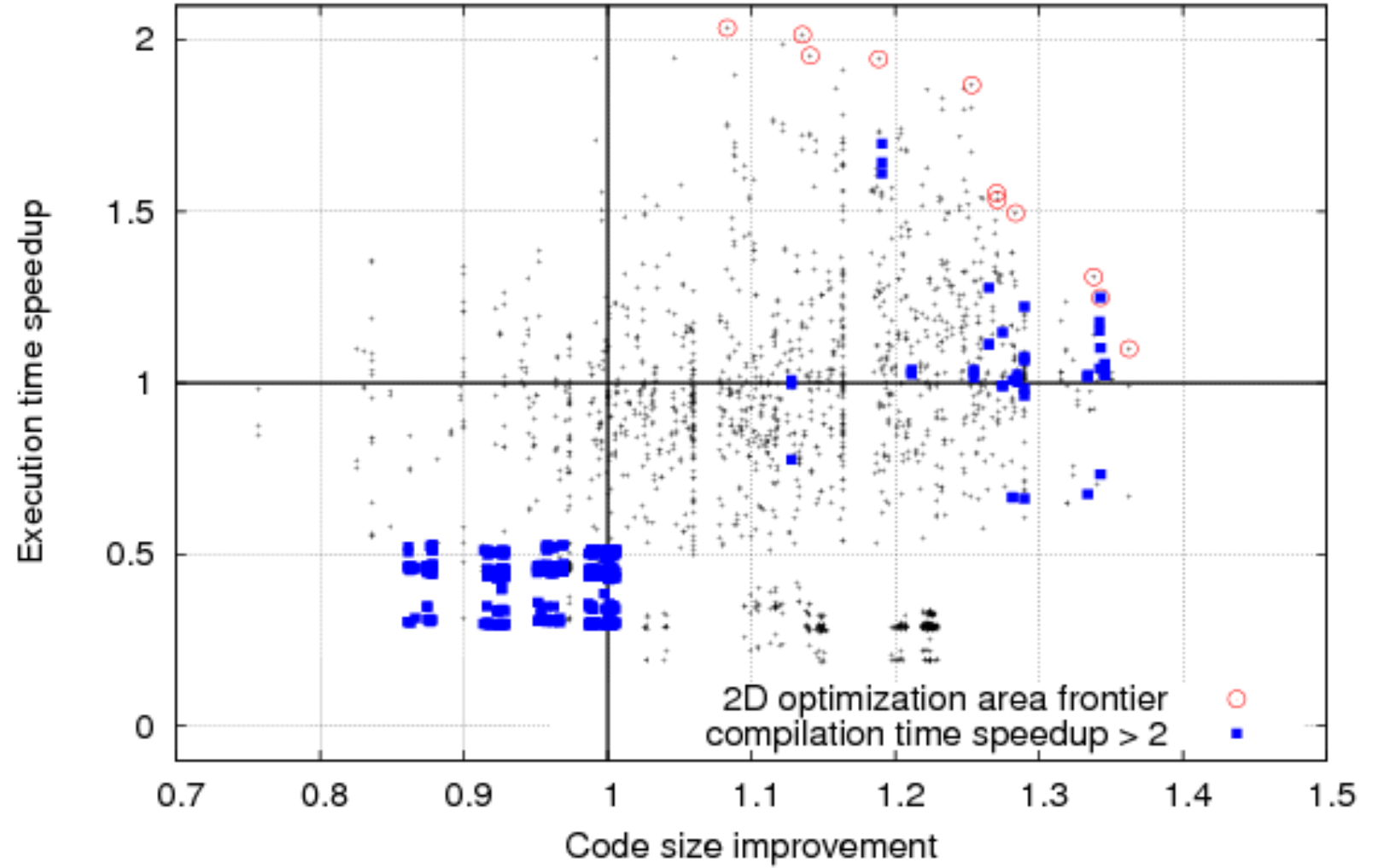
GEMM: complex and non-linear optimization space for 2 transformations (loop tiling and unrolling)



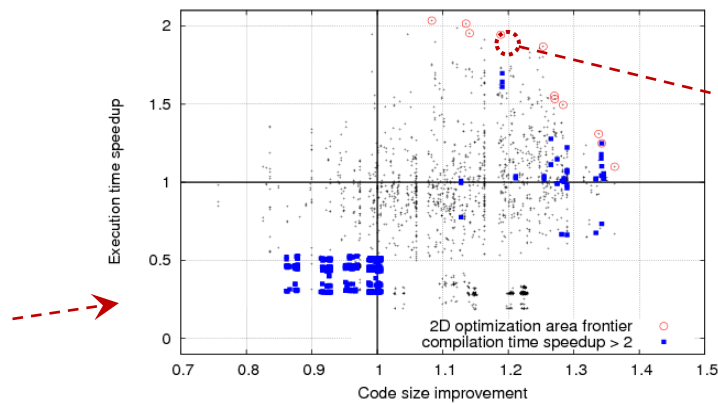
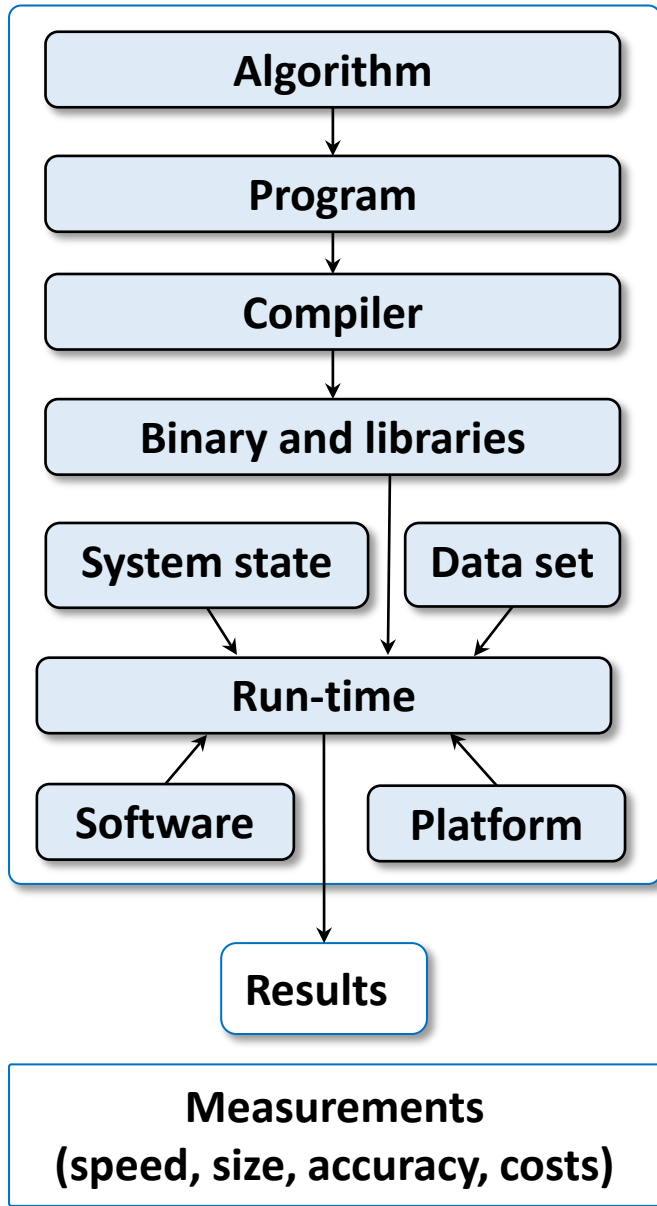
How to accelerate design and optimization space exploration?



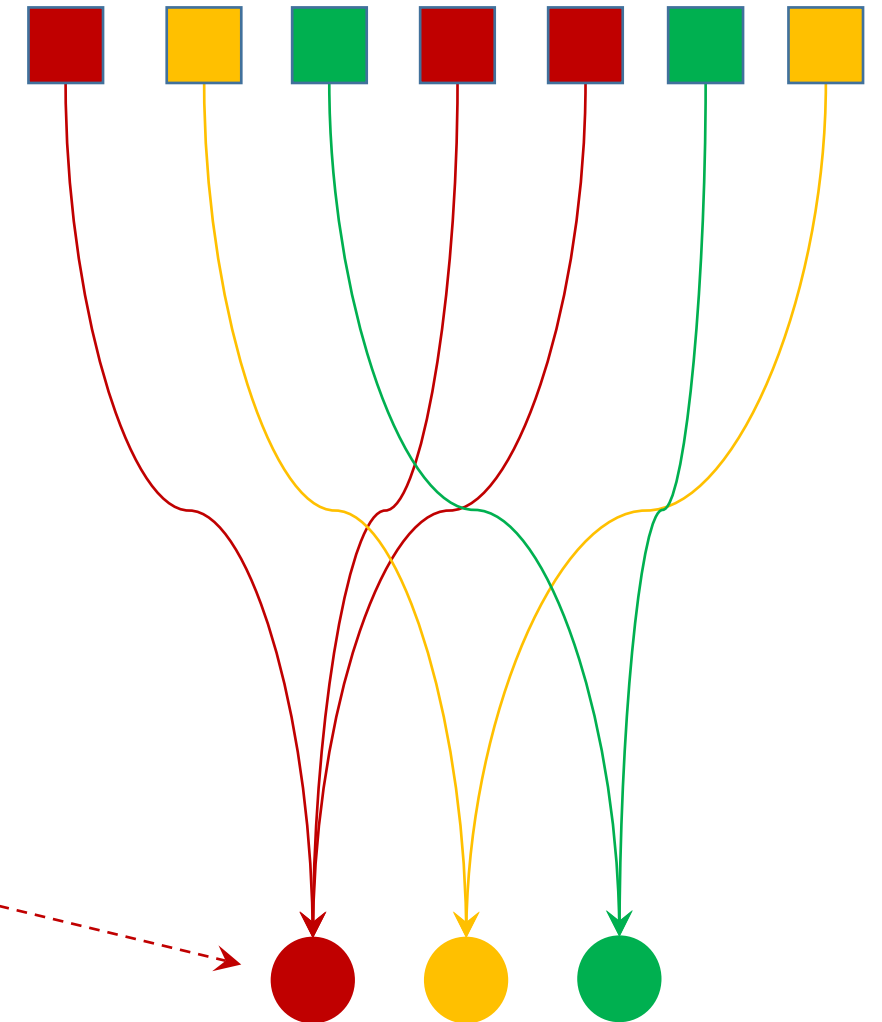
1 program, 1 data set, 1 platform, 1 compiler,
1000 random combinations of optimizations



What about using ML to improve system efficiency?

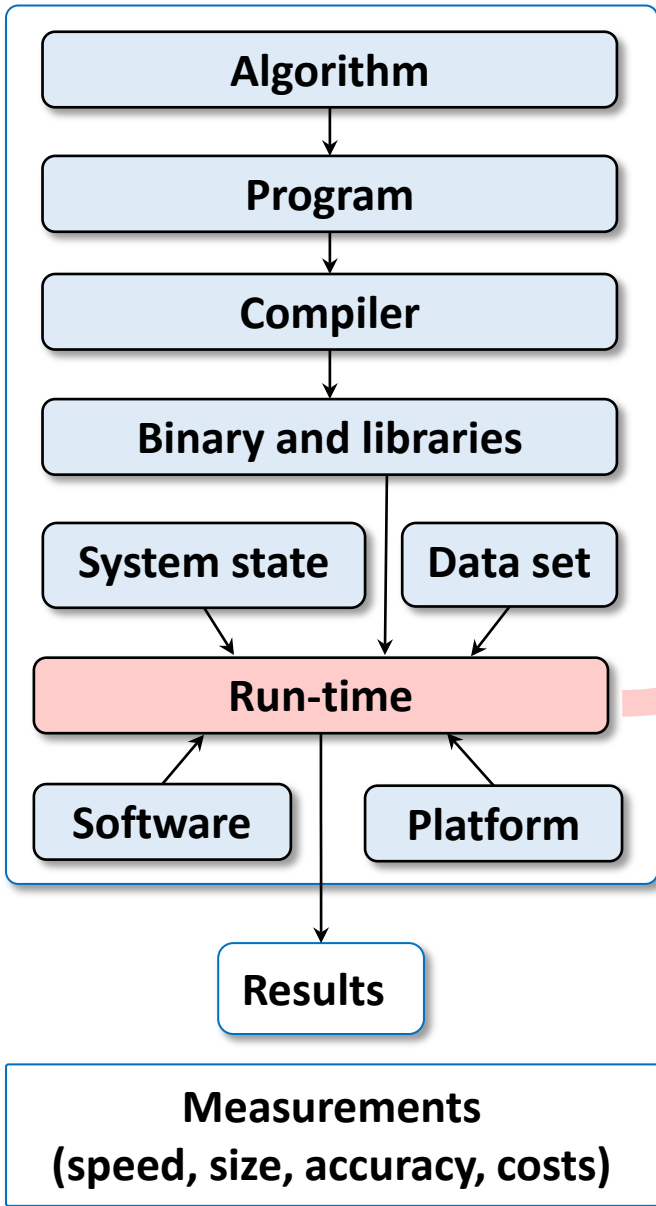


Similar programs, data sets and platforms



may benefit from similar optimizations

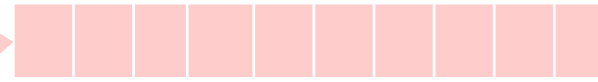
What about using ML to improve system efficiency?



Similar programs, data sets and platforms



Feature vector



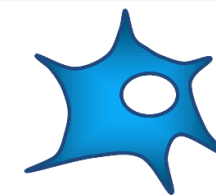
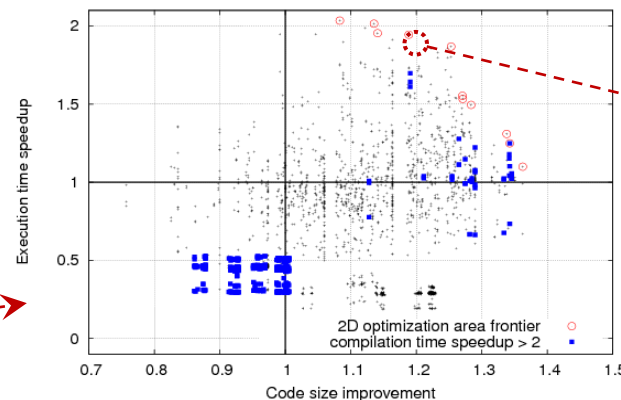
Hardware counters
(dynamic features)



Program embeddings
(semantic features)

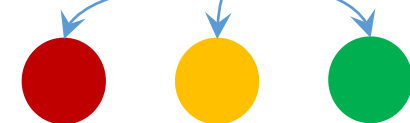
Minimize cost function

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



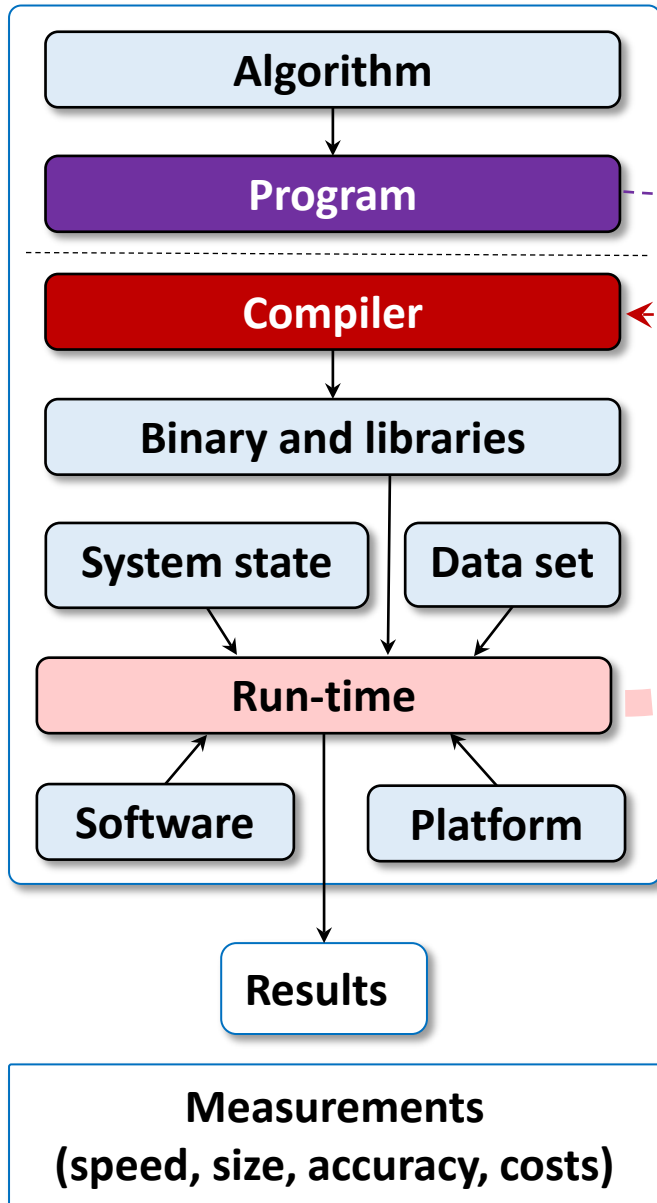
*KNN, decision trees,
SVM, DNN ...*

Train model on N benchmarks



may benefit from similar optimizations

Predict optimizations based on semantic and dynamic features



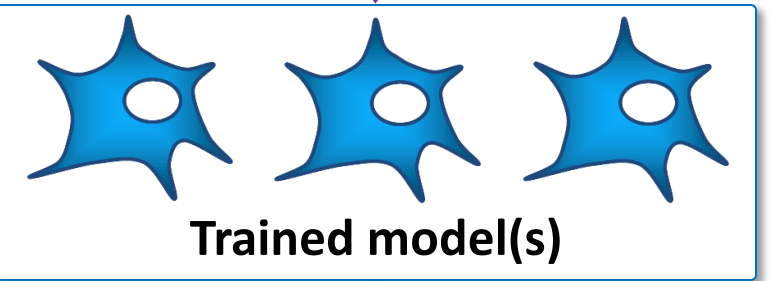
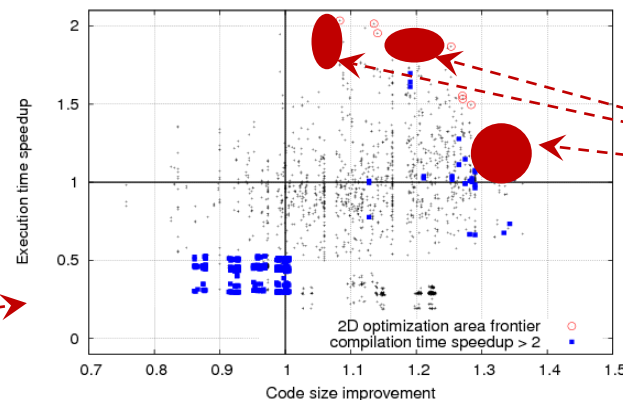
Previously unseen program

Feature vector

Hardware counters
(dynamic features)

Program embeddings
(semantic features)

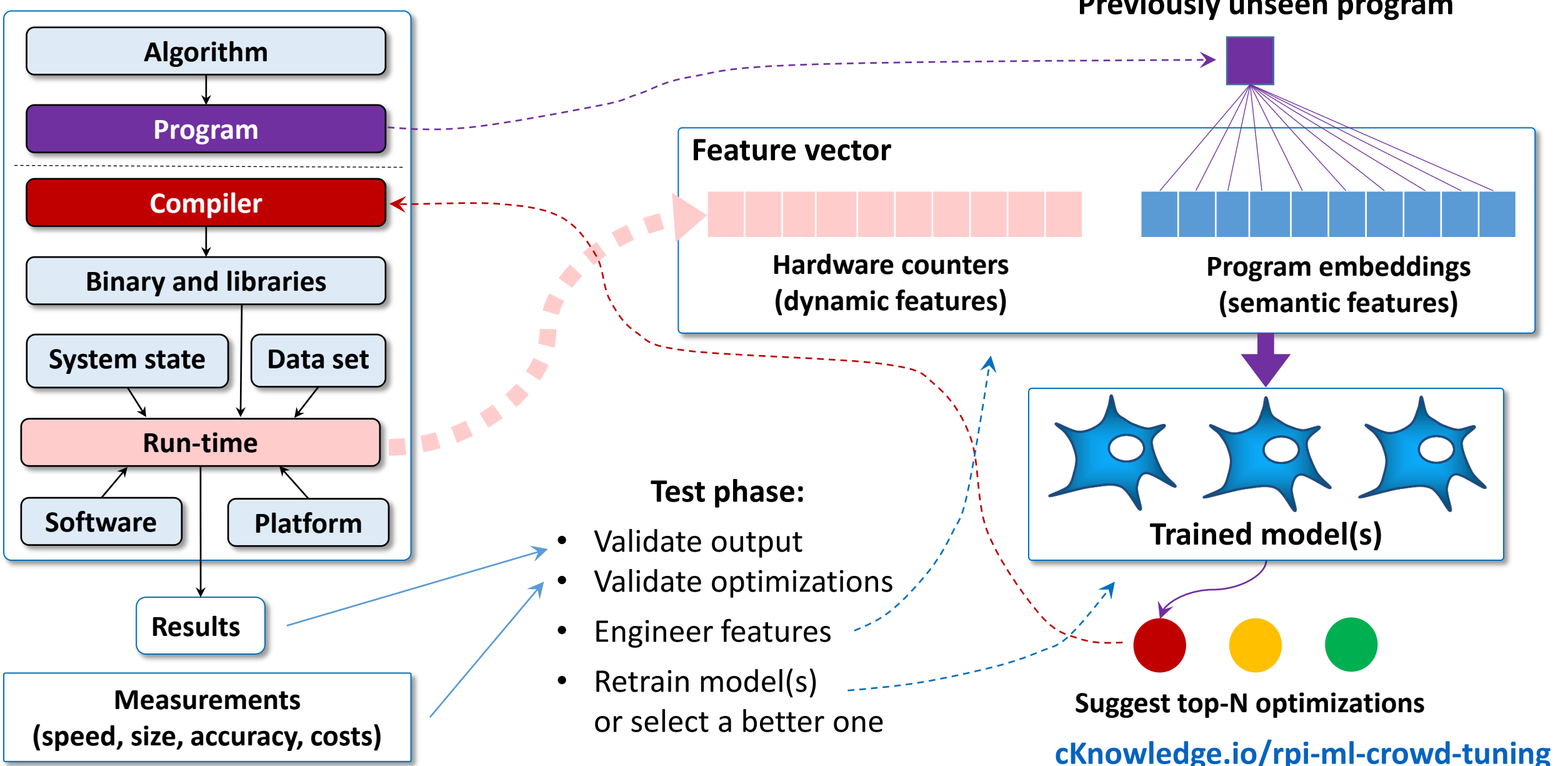
Accelerate auto-tuning and DSE
(probabilistic focused search)



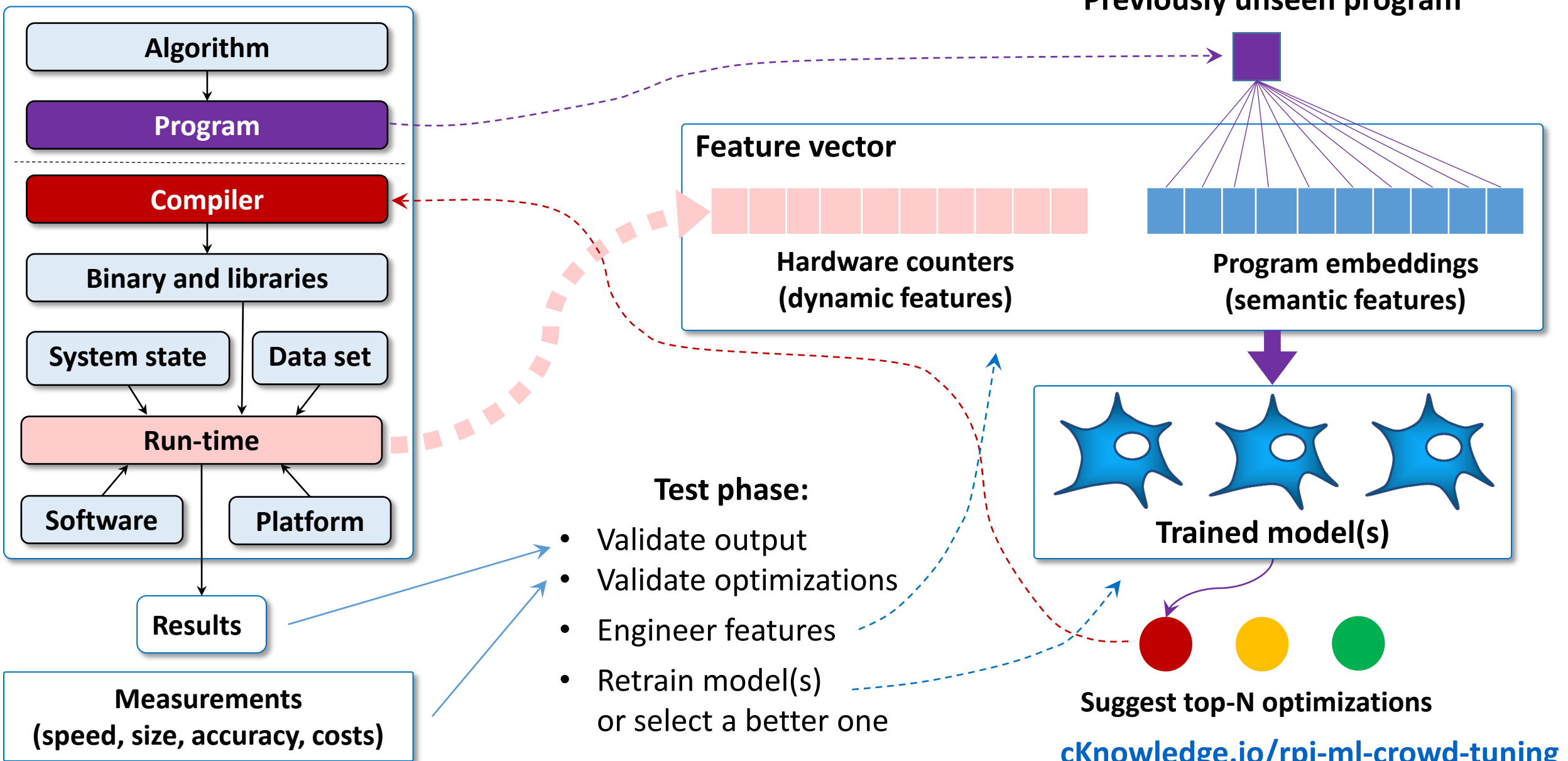
Suggest top-N optimizations

cKnowledge.io/rpi-ml-crowd-tuning

Predict optimizations based on semantic and dynamic features



MILEPOST project (2006-2009): test ML in a production compiler



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

ft1 - Number of basic blocks in the method

Static features

...

ft19 - Number of direct calls in the method

ft20 - Number of conditional branches in the method

ft21 - Number of assignment instructions in the method

ft22 - Number of binary integer operations in the method

ft23 - Number of binary floating point operations in the method

ft24 - Number of instructions in the method

ft25 - Average of number of instructions in basic blocks

...

ft29 - Number of basic blocks with phi nodes in the interval [0, 3]

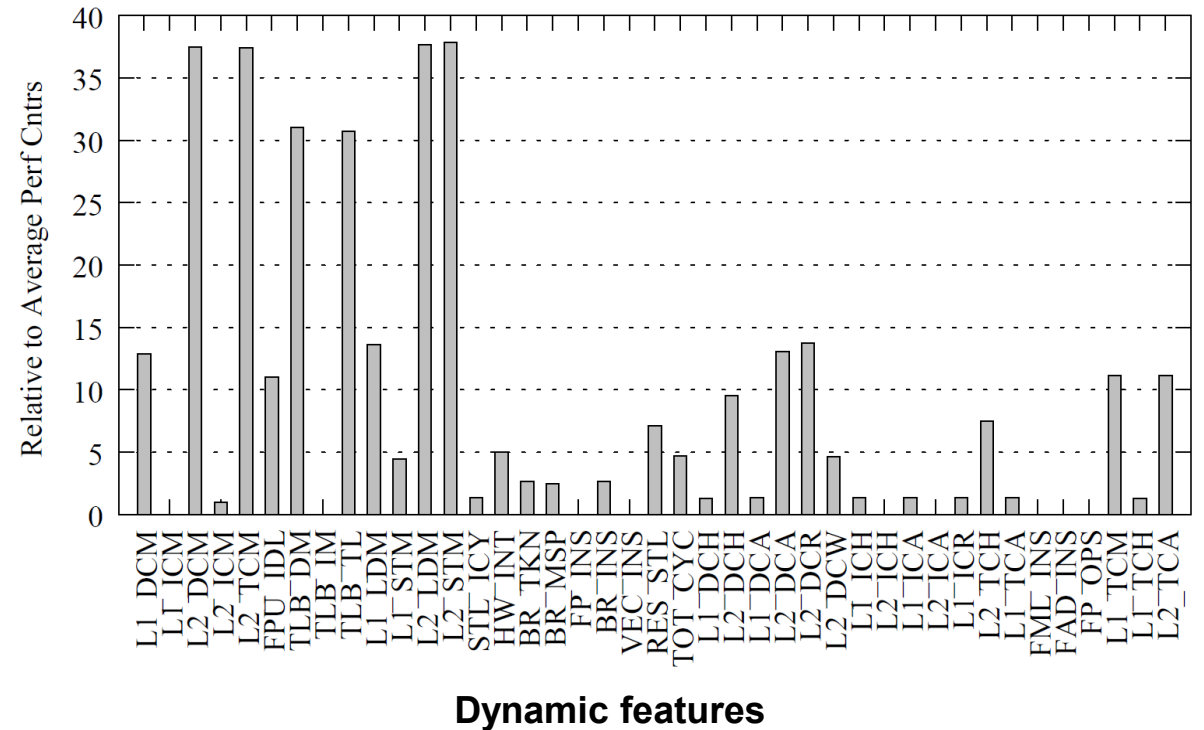
...

ft54 - Number of local variables that are pointers in the method

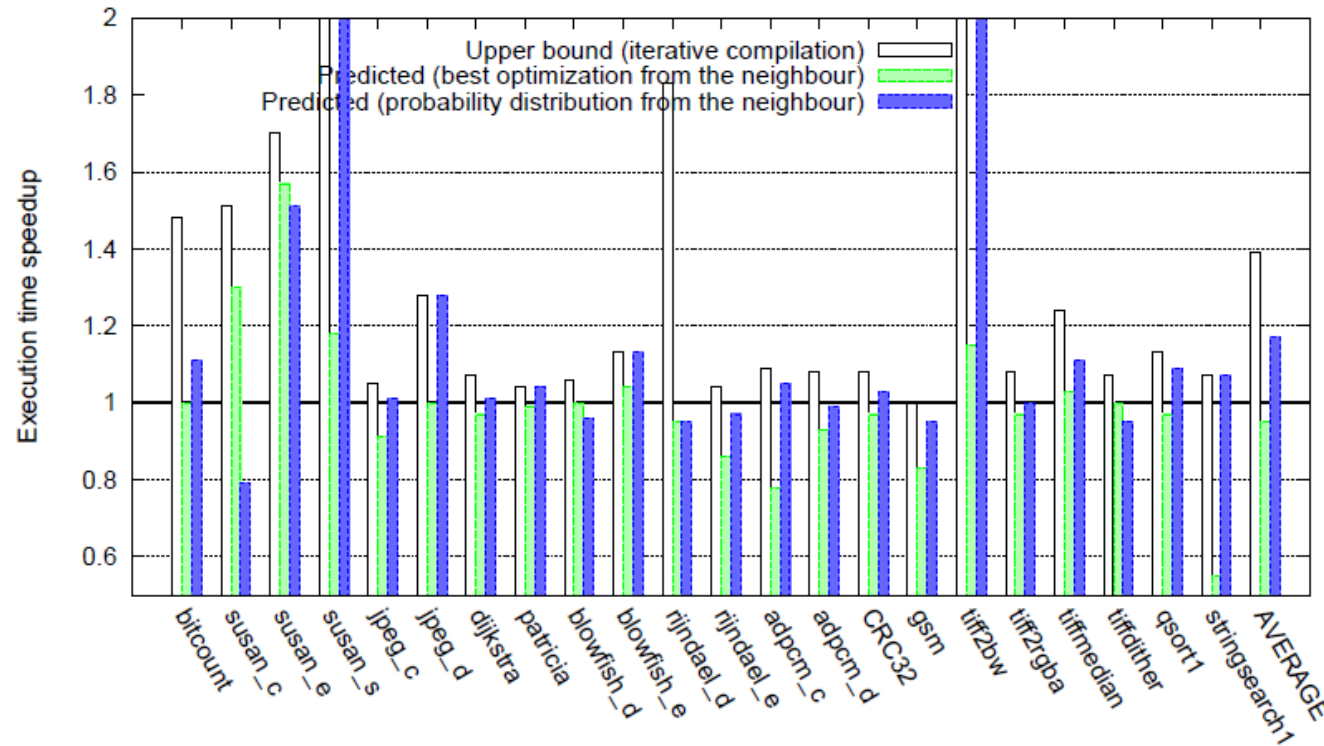
ft55 - Number of static/extern variables that are pointers in the method

...

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



Promising research results on 22 benchmarks and 2 platforms



Top-3 optimization
prediction accuracy:
~87%

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

Grigori Fursin, Yuriy Kashnikov, Abdul Wahid Memon, Zbigniew Chamski, Olivier Temam, Mircea Namolaru, Elad Yom-Tov, Bilha Mendelson, Ayal Zaks, Eric Courtois, Francois Bodin, Phil Barnard, Elton Ashton, Edwin Bonilla, John Thomson, Christopher K. I. Williams, Michael O’Boyle

“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?

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

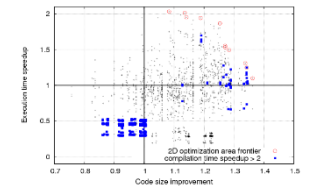
ML training/validation
framework

Communication
with cTuning.org

What could possibly go wrong?

cTuning.org/wiki

SOTA dashboards



Top optimizations



Top models



Mediawiki

Web API

cTuning.org/wiki

github.com/ctuning/reproduce-milepost-project

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

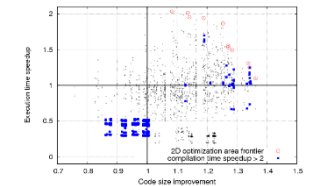
ML training/validation
framework

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*)

cTuning.org/wiki

SOTA dashboards



Top optimizations



Top models



Mediawiki

Web API

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.

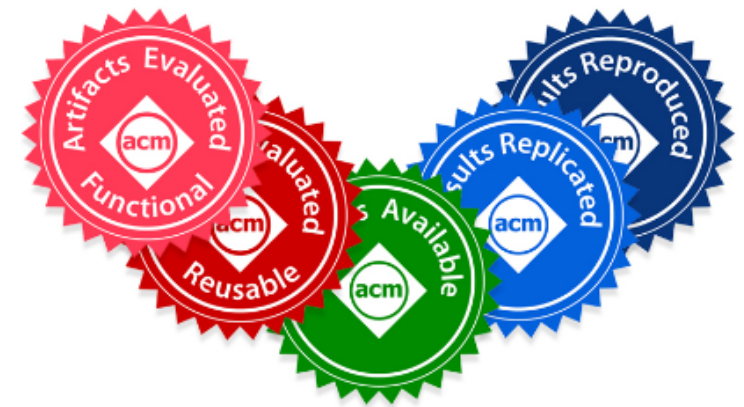
Encourage code and data sharing and test for artifact functionality, reproducibility and reusability separately.

Try new publication models with open reviewing: arxiv.org/pdf/1406.4020.pdf (adapt-workshop.org)

2015-now: ACM and NeurIPS/ICML initiatives

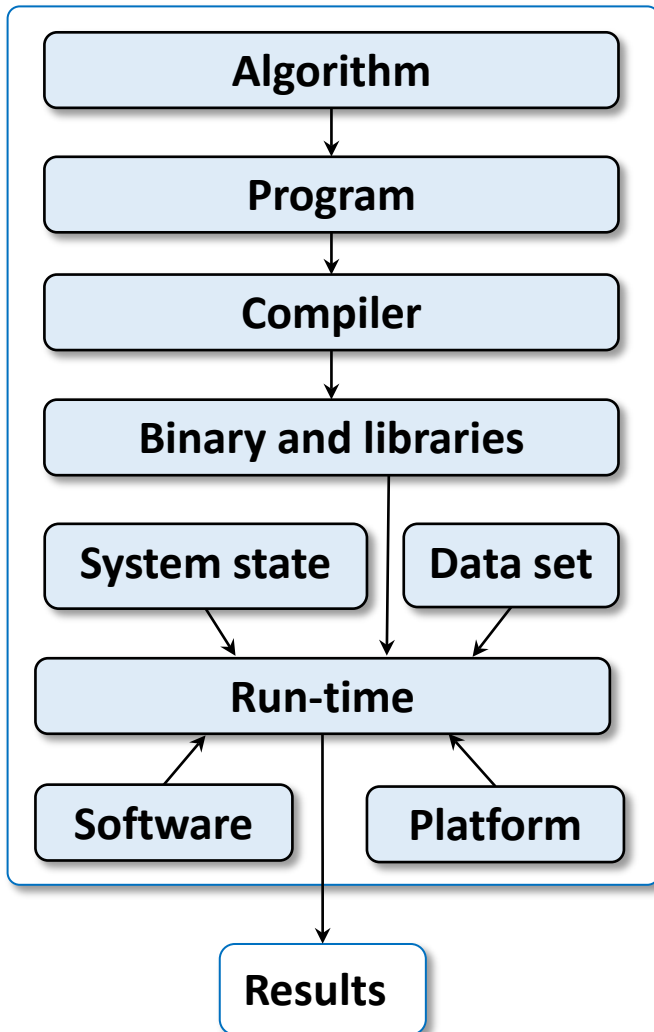
- **The ACM Task Force on Data, Software, and Reproducibility in Publication**
www.acm.org/publications/task-force-on-data-software-and-reproducibility
- **Common Artifact Review and Badging policy**
www.acm.org/publications/policies/artifact-review-and-badging-current
- **Artifacts and reproducibility badges in the ACM Digital Library**
dl.acm.org/doi/proceedings/10.1145/3229762
dl.acm.org/search/advanced
- **ACM SIGARCH Checklist for empirical evaluation**
bit.ly/sigarch-checklist
- **ACM Emerging Interest Group on Reproducibility**
reproducibility.acm.org

- **Reproducibility initiative at NeurIPS'19**
nips.cc/Conferences/2019/CallForPapers
- **PapersWithCode tips for publishing research code**
github.com/paperswithcode/releasing-research-code
- **NISO artifact badges**
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



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 was to learn how to automate artifact evaluation

cTuning.org/ae/checklist.html

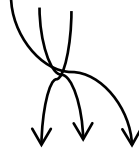
cKnowledge.io/reproduced-papers

dl.acm.org

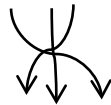
Started noticing some patterns across different projects

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

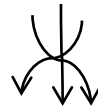
Ad-hoc scripts
to compile and
run a program
or a benchmark



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

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*

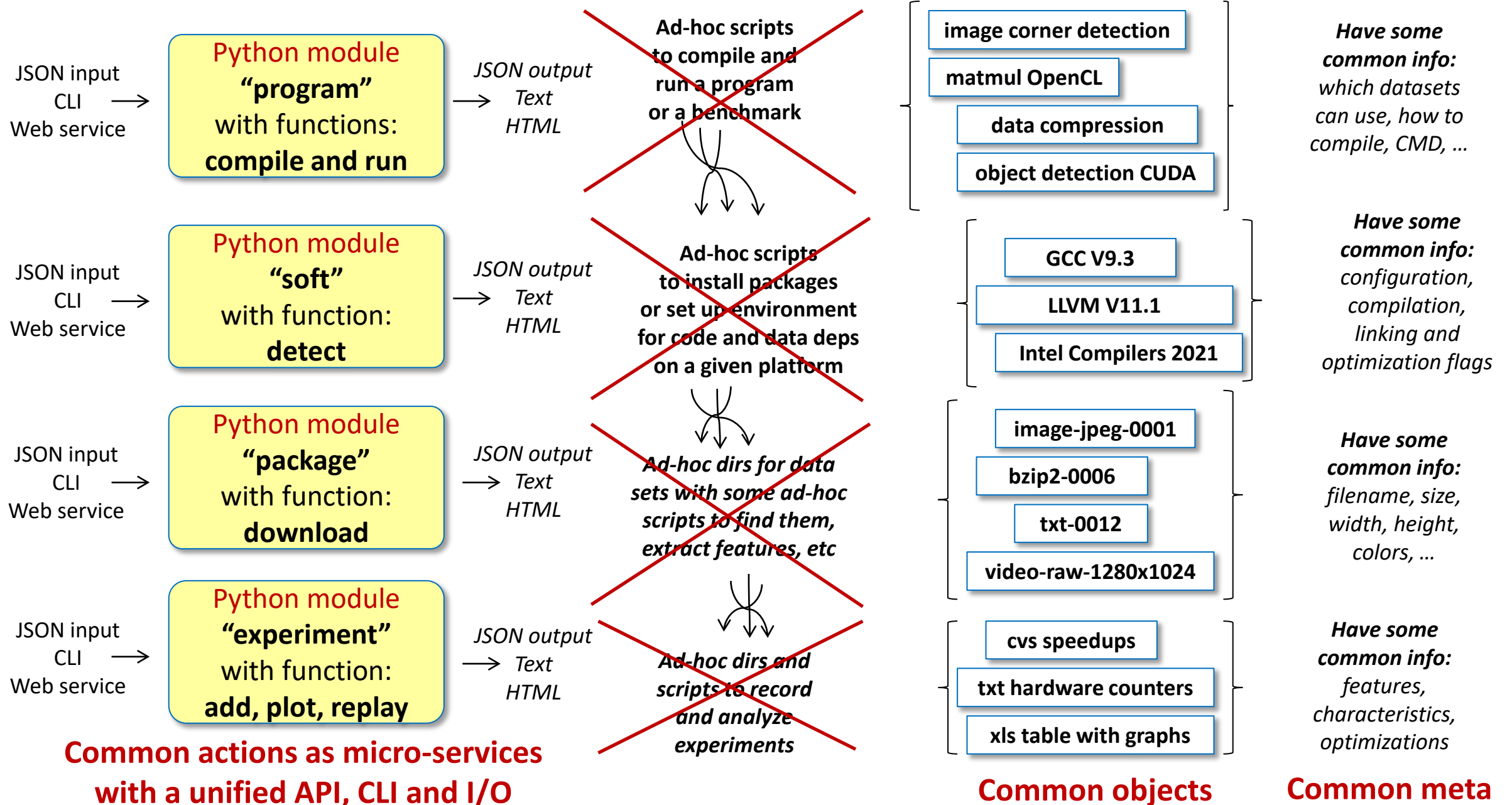
Could be reused across projects

Common (reusable) actions

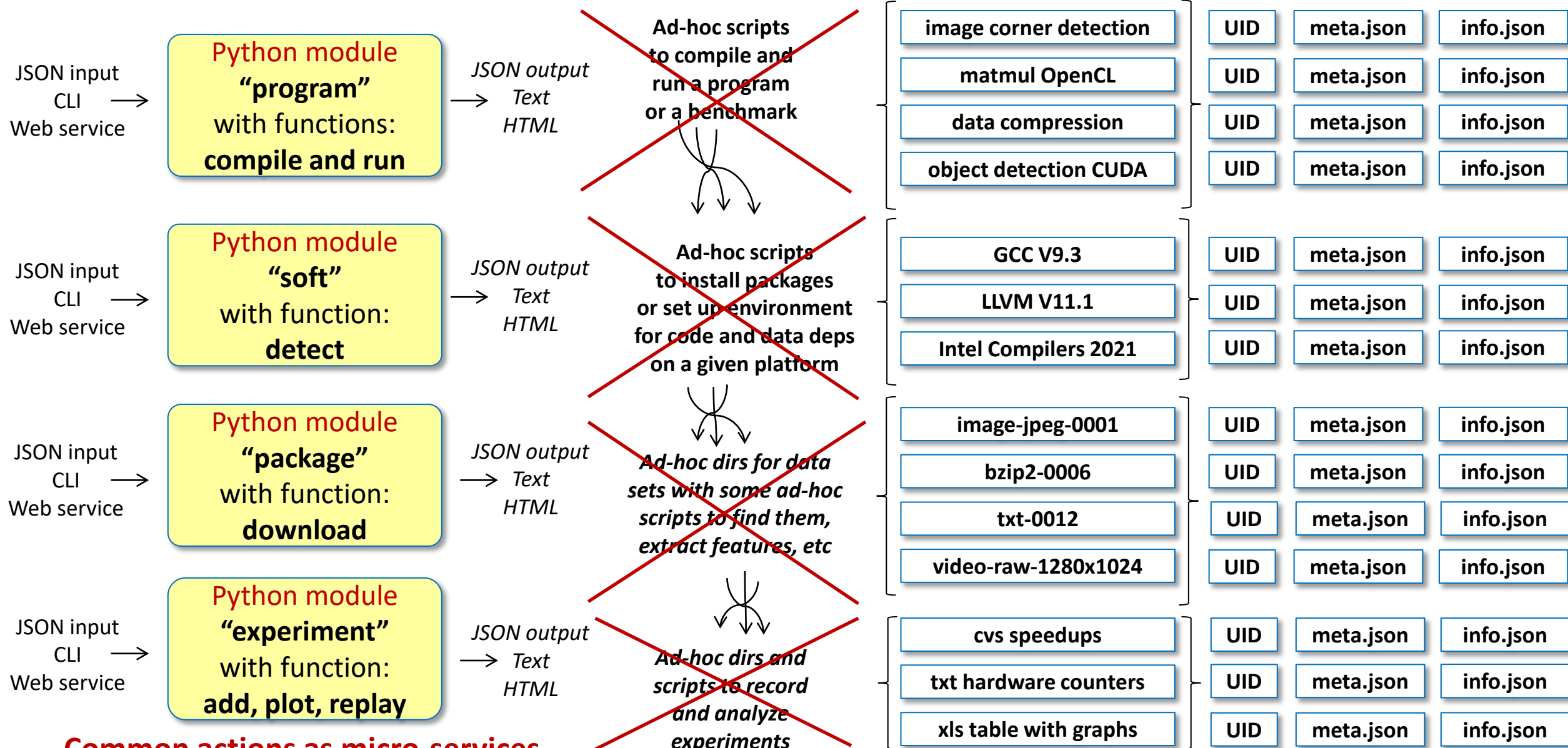
Common objects

Common meta

Can convert ad-hoc scripts into micro-services with a unified API



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

Can share 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

```
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:*
```

Artifact automated and reusable

Collective Knowledge COMPATIBLE

GitHub, GitLab, BitBucket, Docker, IPython notebook, ZIP file ...

/ 1st level dirs

program

program

program

soft

soft

soft

dataset

dataset

dataset

dataset

experiment

experiment

experiment

/ 2nd level dirs

image corner detection

matmul OpenCL

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

/ 3rd level dirs

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

meta.json

info.json

Started automating and sharing the most common actions from reproduced papers

```
ck {function} {module} (:{component}) @input.json
```

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 cknowledge.io/browse

```
# 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))

{
  "return": 0,
  "os_uoa": "windows-64", "os_uid": "7a95e0754c37610a",
  "host_os_uoa": "windows-64", "host_os_uid": "7a95e0754c37610a",
  "features": {
    "cpu_unique": [
      {
        "ck_cpu_name": "Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz",
      }
    ],
    "gpu": {
      "name": "NVIDIA GeForce 940MX", "vendor": "NVIDIA«
    },
    "platform": {
      "vendor": "LENOVO",
      "name": "LENOVO ThinkPad T470p (20J6CTO1WW)",
      "model": "20J6CTO1WW"
    }
  }
}
...
```

Enabled portable program workflows that can adapt to continuously changing SW/HW

ck compile **program:conv-armcl-openc1-uint8**

ck run **program:conv-armcl-openc1-uint8** --env. CK_SEED=123

CK module **program** (Python + meta JSON/YAML to implement portable workflows)

- **Query CK DB** to find **conv-armcl-openc1-uint8**
- Load **conv-armcl-openc1-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-openc1-uint8**

- *File **conv.cpp** with algorithm*
- *File **meta.json** describing SW/HW deps, configs and compile/run info*

```
"deps": {
  "compiler": {"name": "C++ compiler", "tags": "compiler,lang-cpp"},
  "lib-ntest": {"name": "NNTest library", "tags": "lib,ntest"},
  "library": {"name": "ARM Compute Library (OpenCL, uint8)",
    "or_tags": "armcl;vdefault;vconv-uint8", "tags": "lib,arm-compute-library,vopenc1"},
  "openc1": {"name": "OpenCL Library", "tags": "lib,openc1"}
},
"run_cmds": {
  "dataset_tags": ["dataset", "ntest", "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": ["ntest", "armcl", "conv", "uint32", "vopenc1"],
```

cknowledge.io/c/module/program

cknowledge.io/c/program/conv-armcl-openc1-uint8

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,

ZIP file with datasets as a CK DB

- CK packages and data sets

GitHub private company repo as a CK DB

- CK modules, programs,

Colab/Jupyter notebook as a CK DB

Auto-generated paper from CK DB

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

Bringing in DevOps and FAIR principles (extended to code):

en.wikipedia.org/wiki/FAIR_data

Findable
Accessible
Interoperable
Reusable
+
Portable

the same CK API/CLI

`ck run program:ck-crowduning`

`ck run automotive-susan --min`
`ck run automotive-susan --fast`

`ck run automotive-susan`

Automatically
adapt to user
environments

```
$ ck autotune program:cbench-automotive-susan
```

```
$ ck crowd tune program:cbench-automotive-susan
```

```
$ ck replay experiment
```



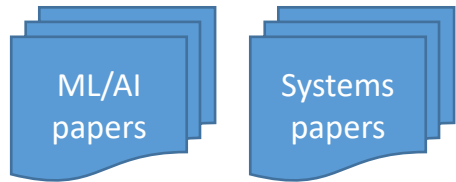
cknowledge.io/programs

cknowledge.io/c/docker

cknowledge.io/repos

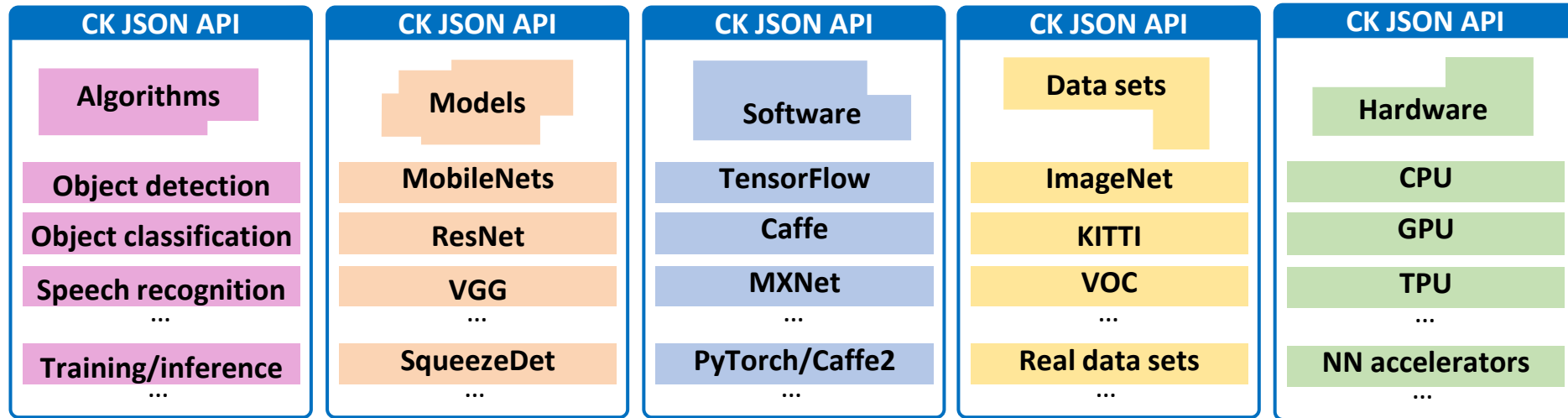
cknowledge.io/notebook

cKnowledge.io platform

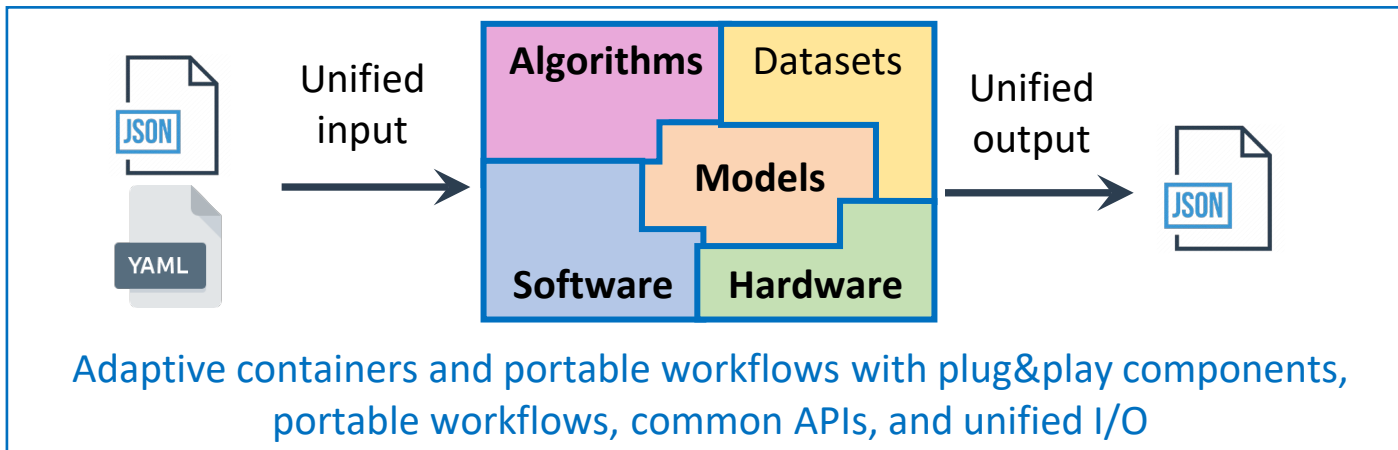


Share portable workflows, adaptive containers, automation actions and plug&play components along with research papers:

cKnowledge.io/browse



cKnowledge.io/reusable-research

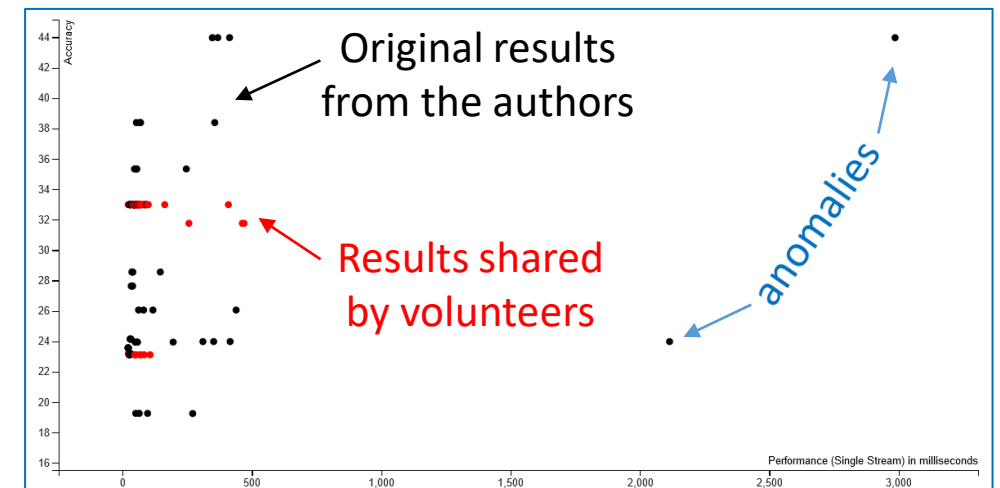


Quickly prototype and test ideas on any tech. stack

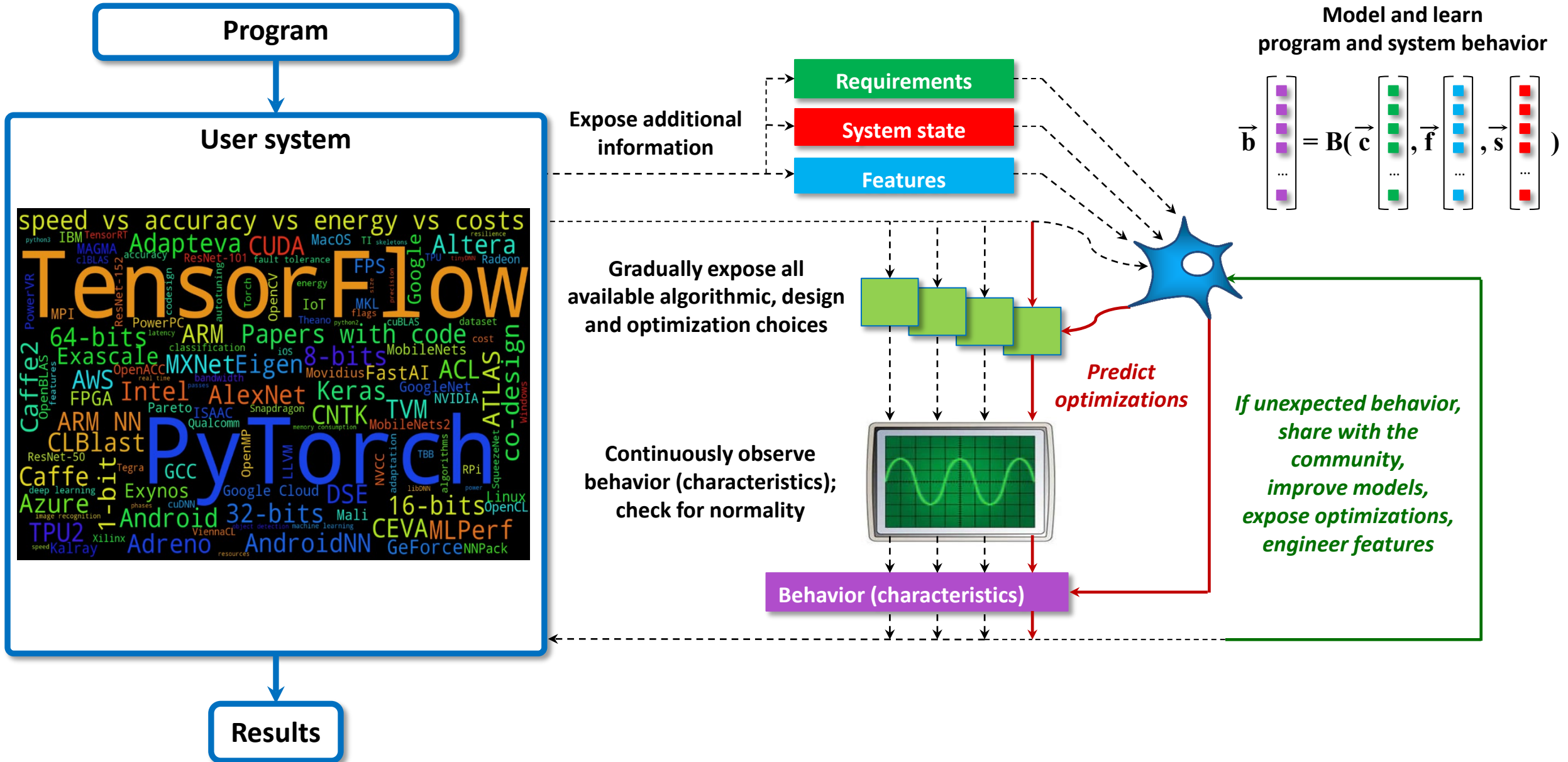


Enable “live” research papers that can be validated and improved by the community across diverse models, data sets, software and hardware:

cKnowledge.io/reproduced-results

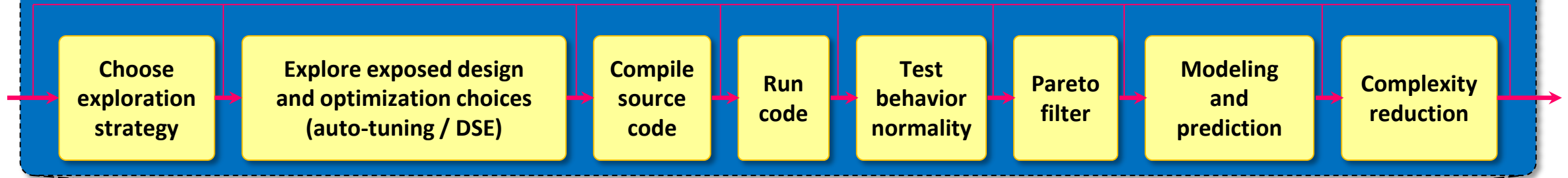


Collaboratively expose optimizations, characteristics and features in different components



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)



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

cKnowledge.io/rpi-ml-crowd-tuning

GitHub
GitLab
ACM DL
ArXiv

...



A Collective Knowledge workflow for collaborative research into multi-objective autotuning and machine learning techniques

We thank the *Raspberry Pi Foundation* and *cTuning Foundation* for supporting this proof-of-concept project to automatically generate "live" papers with portable CK workflows and reusable components that can be reproduced and build upon by the community.

Corresponding author: *Grigori Fursin*

[Auto-generated ArXiv PDF] [Optimization repository] [GCC] [LLVM] [CK project overview]

[Andrew Ng's ML course] [Google ML crash course] [Facebook Field Guide to ML]

Developing efficient software and hardware has never been harder whether it is for a tiny IoT device or an Exascale supercomputer. Apart from the ever growing design and optimization complexity, there exist even more fundamental problems such as lack of interdisciplinary knowledge required for effective software/hardware co-design, and a growing technology transfer gap between academia

github.com/ctuning/reproduce-milepost-project

Public optimization repository

Collective Knowledge Aggregator *proof-of-concept*

Crowd results Raw CK browser Graphs Reports Datasets Models Home

Select CK-powered unified experimental workflow: [auto/crowd-tune GCC compiler flags (minimize execution time)]

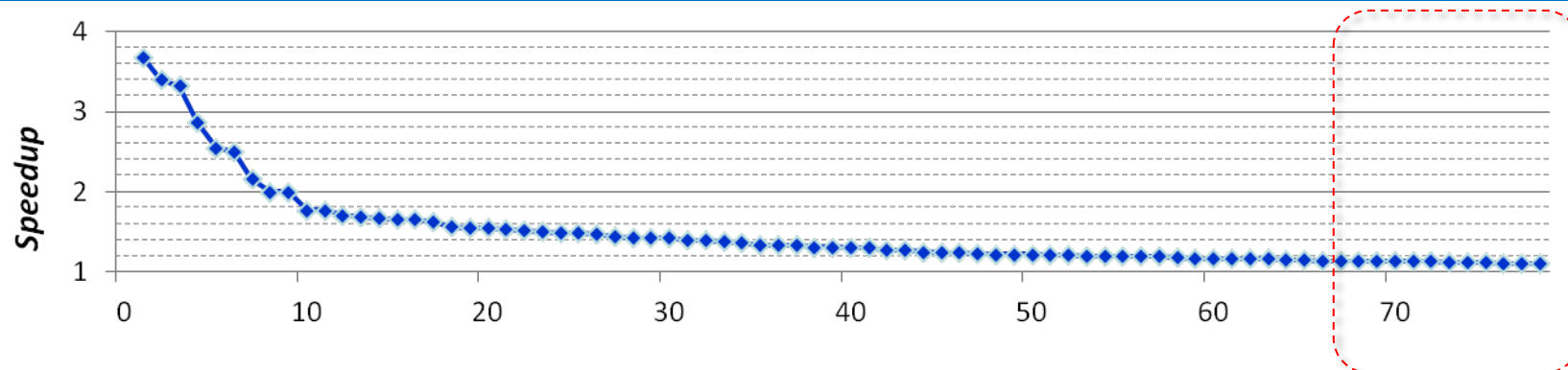
[Participated users, platforms, OS, CPU, GPU, GPGPU, NN, NPU] [How to participate] [Motivation (PPT) (PDF)] [Papers 1, 2, 3] [Android training set] [Unified AI]

Prune solutions by Compiler: [v] CPU: [v]

#	UID	Discuss	Number of distinct solutions	Max improvement (first characteristic)	Compiler	CPU
1	Click to see solutions (34aaf1afe34b7fb2)	Wiki	1	1.11	GCC 9.3.0	AMD Ryzen 5 3600 6-C
2	Click to see solutions (daa82885ae341308)	Wiki	15	3.33	GCC 8.0.1	Intel(R) Xeon(R) CPU I 2.30GHz
3	Click to see solutions (4863a4eb1b1f3f36)	Wiki	17	3.12	GCC 7.1.0	Intel(R) Xeon(R) CPU I 2.30GHz
4	Click to see solutions (79bca2b76876b5c6)	Wiki	14	8.00	GCC 7.1.0	BCM2709
5	Click to see solutions (3fbae8a0ae45d496)	Wiki	26	3.07	GCC 7.0.0	Intel(R) Core(TM) i5-2
6	Click to see solutions (e875114e0a89b28a)	Wiki	13	7.70	GCC 6.3.0	Intel(R) Xeon(R) CPU I 2.30GHz
7	Click to see solutions (8e9b764371ab7bec)	Wiki	17	4.36	GCC 6.2.0	Intel(R) Xeon(R) CPU I 2.30GHz
8	Click to see solutions (3dbc38b41cdbc742)	Wiki	1	1.29	GCC 6.2.0	Intel(R) Xeon(R) CPU I

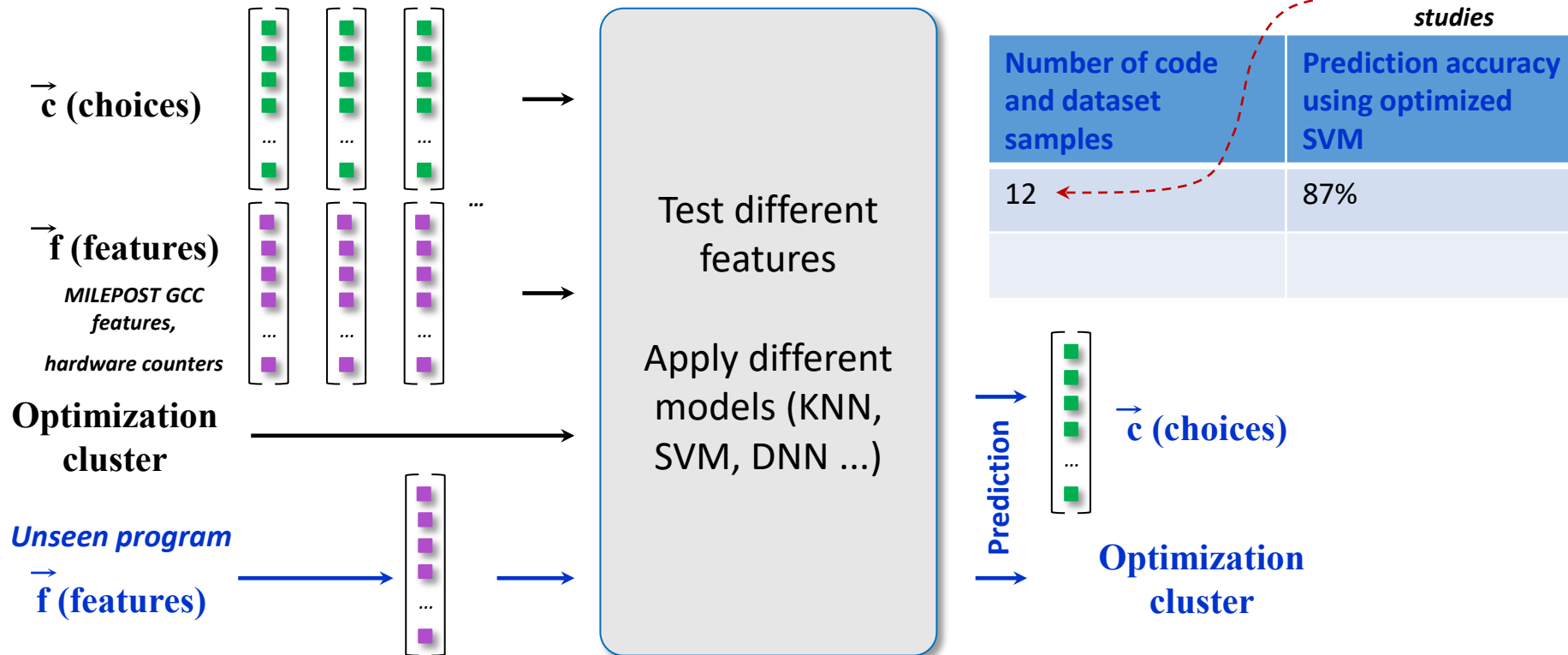
cKnowledge.org/repo-beta

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

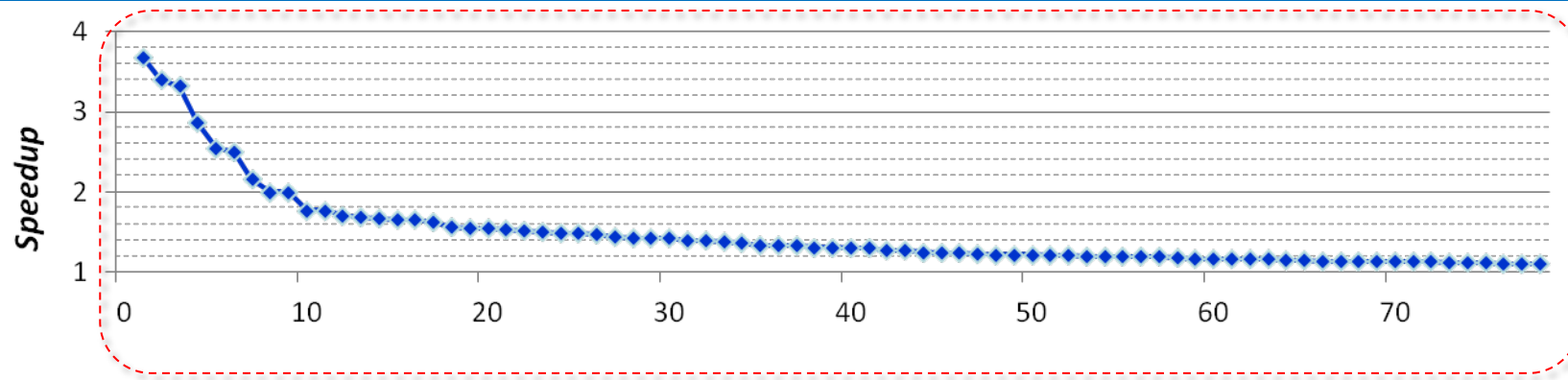


Training set: distinct combination of compiler optimizations (clusters)

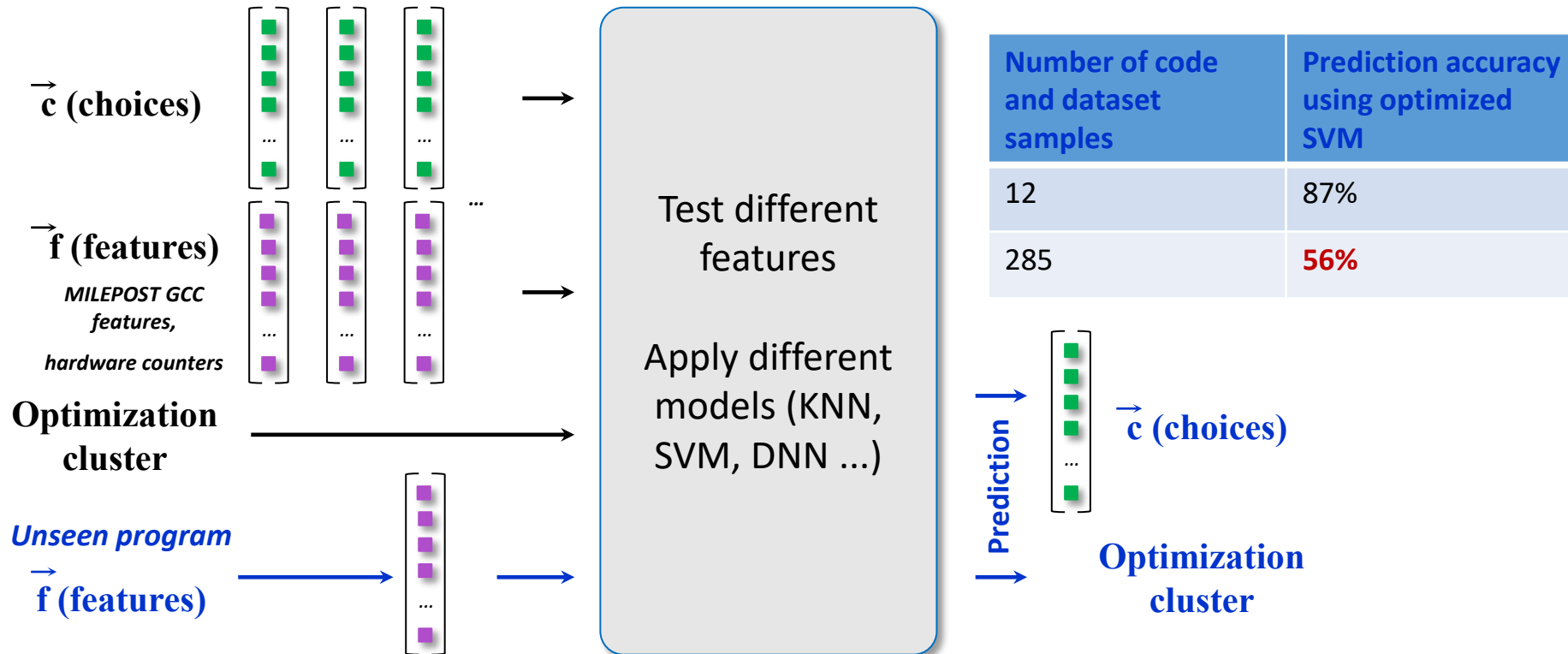
Current limited studies



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



Training set: distinct combination of compiler optimizations (clusters)



Expose unexpected behaviour and learn features with the community

Image **B&W** threshold filter

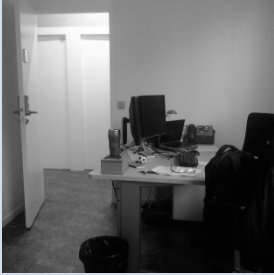
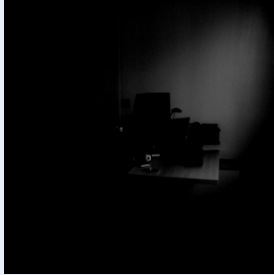
```
*matrix_ptr2++ = (temp1 > T) ? 255 : 0;
```

Experiment	-O3	-O3 -fno-if-conversion
Shared experiment ₁	<i>reference execution time</i>	no change
Shared experiment ₂	no change	+17.3% improvement

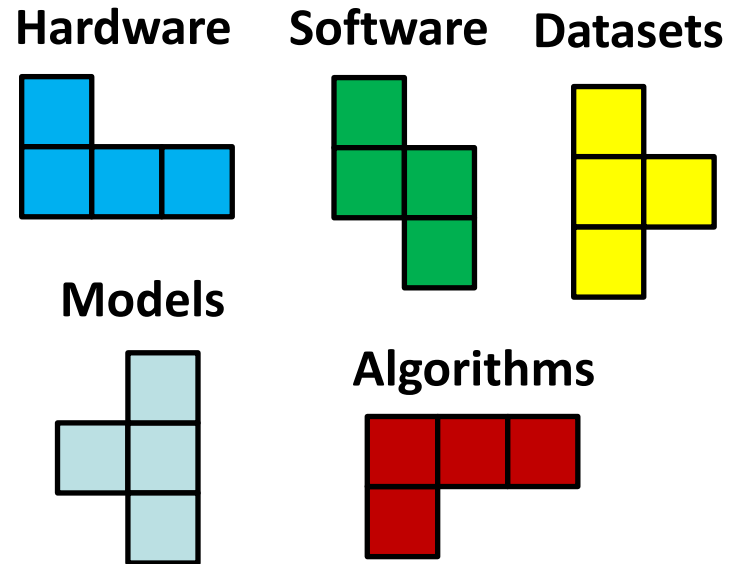
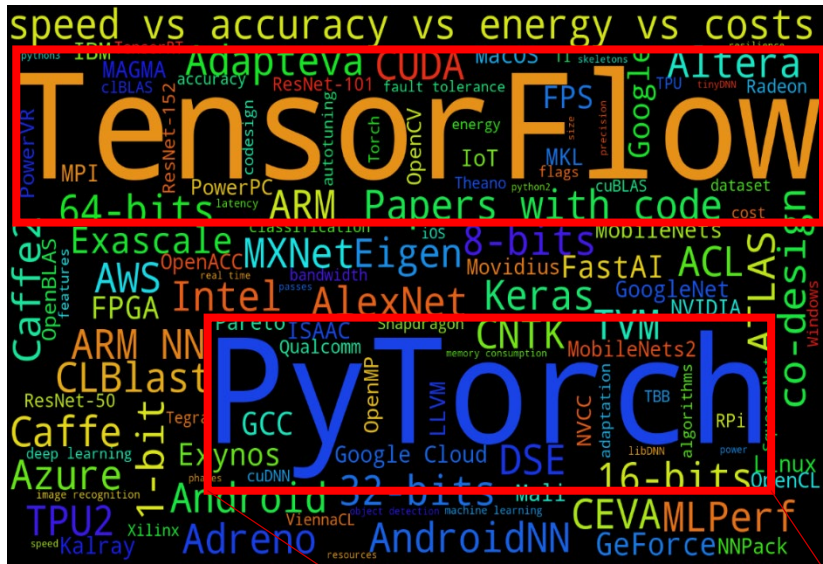
Expose unexpected behaviour and learn features with the community

Image **B&W** threshold filter

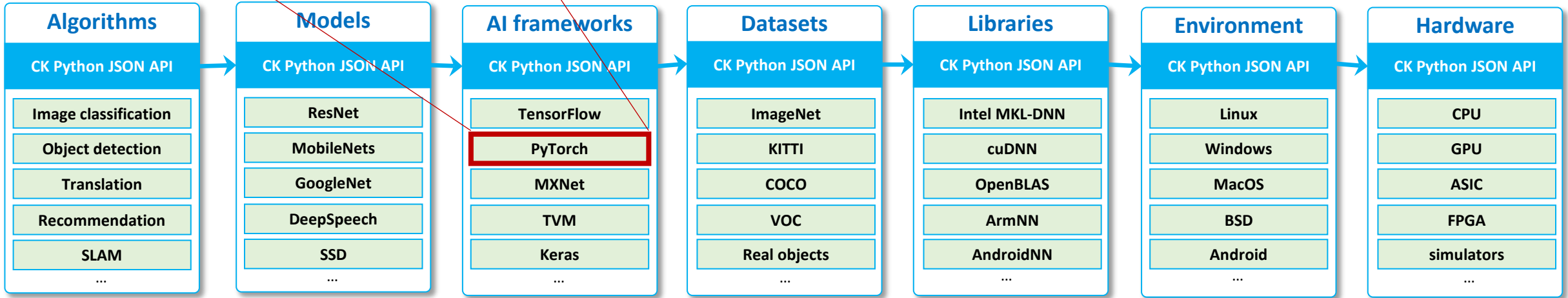
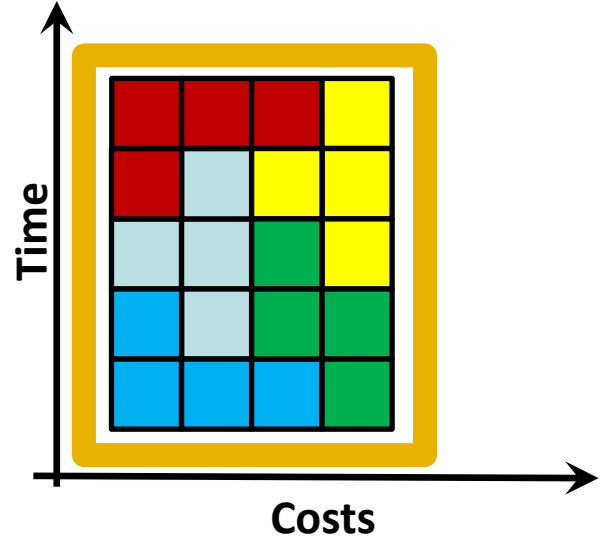
```
*matrix_ptr2++ = (temp1 > T) ? 255 : 0;
```

Experiment	-O3	-O3 -fno-if-conversion
Shared experiment ₁ 	<i>reference execution time</i>	no change
Shared experiment ₂ 	no change	+17.3% improvement

Started converting artifacts from deep learning papers to CK



Assemble portable workflows with plug&play components as LEGO bricks



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

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

CK workflow ₁	CK workflow ₂	CK workflow ₃	CK workflow ₄	CK workflow ₅
Intel Caffe, BVLC Caffe	TensorFlow; Keras; Avro	MXNet; NNVM/TVM OpenBLAS vs ArmCL	MXNet; NNVM/TVM	ArmCL (OpenCL) 18.01 vs 18.03
ResNet-50, SSD Inception-v3; 32-bit, 8-bit	AlexNet; VGG16	ResNet-50, VGG16, MobileNet-v1-1.0-224	ResNet-18	MobileNets-v1: {1.0,0.75,0.50,0.25} x {224,192,160,128}
Intel C++ Compiler 17.0.5 20170817	GCC	GCC; LLVM; CUDA	GCC; LLVM	GCC; LLVM
AWS + Intel Xeon® Platinum 8124M	NVIDIA Jetson TX2; 12x Raspberry Pi 3	Firefly-RK3399	Pynq (Xilinx FPGA)	HiKey 960 (Mali-G71 GPU)

Open reviewing at <https://github.com/ctuning/ck-request-asplos18-results> via GitHub issues.

Published validated papers with reusable workflows in the ACM DL

Conference Proceedings Upcoming Events Authors Affiliations Award Winners

Select All

Sections

ReQuEST '18: Proceedings of the 1st on Reproducible Quality-Efficient Systems ...
2018

← Previous Next →

[Abstract](#)

[Proceeding Downloads](#)

SESSION: Keynote

[SESSION: Artifact presentations](#)


[Contributors](#)

[Index Terms](#)

[Comments](#)




ACM DL DIGITAL LIBRARY

RESEARCH-ARTICLE [Highly Efficient 8-bit Low Precision Inference of Convolutional Neural Networks with IntelCaffe](#)


 [Jiong Gong](#), [Haihao Shen](#), [Guoming Zhang](#), [Xiaoli Liu](#), [Shane Li](#), [Ge Jin](#), + 3

June 2018, Article No.: 2, pp 1 • <https://doi.org/10.1145/3229762.3229763>

High throughput and low latency inference of deep neural networks are critical for the deployment of deep learning applications. This paper presents the efficient inference techniques of IntelCaffe, the first Intel(R) optimized deep learning framework ...




6 325    [Get Access](#)

RESEARCH-ARTICLE [Optimizing Deep Learning Workloads on ARM GPU with TVM](#)


 [Lanmin Zheng](#), [Tianqi Chen](#)

June 2018, Article No.: 3, pp 1 • <https://doi.org/10.1145/3229762.3229764>

With the great success of deep learning, the demand for deploying deep neural networks to mobile devices is growing rapidly. However, current popular deep learning frameworks are often poorly optimized for mobile devices, especially mobile GPU. In this ...

3 727    [Get Access](#)

RESEARCH-ARTICLE [Real-Time Image Recognition Using Collaborative IoT Devices](#)

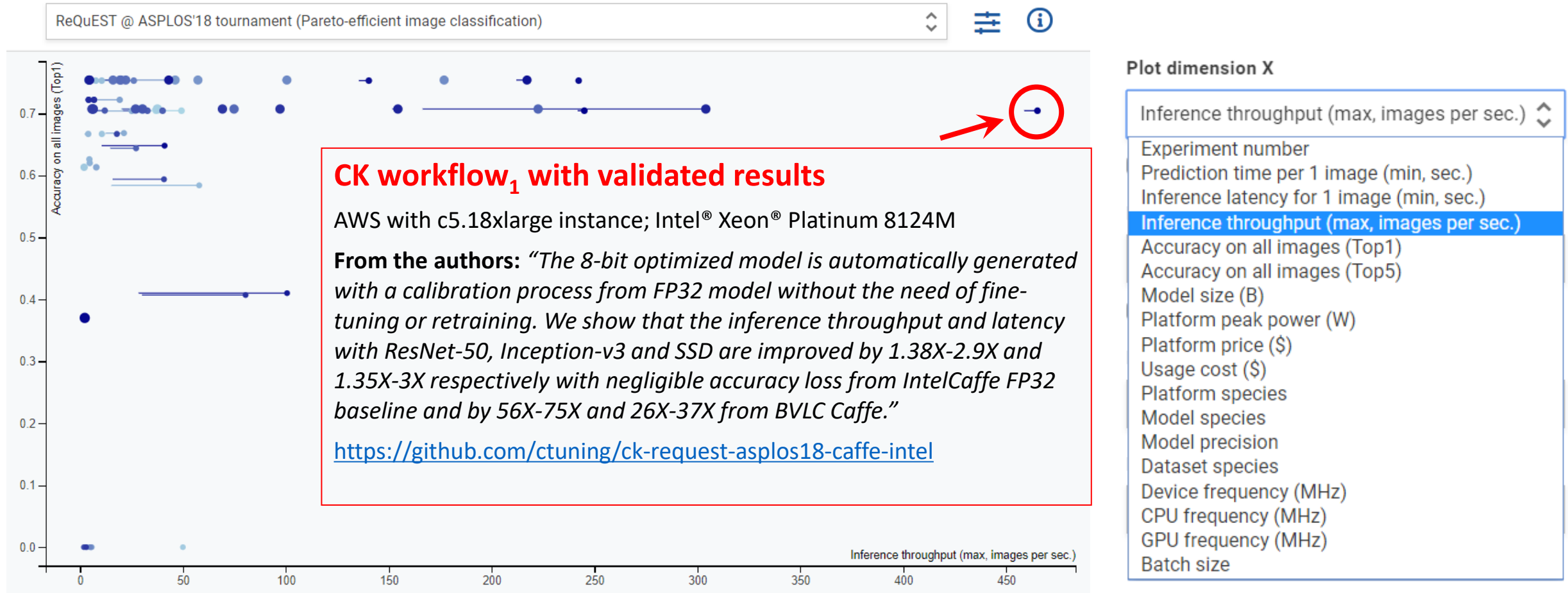
 Feedback

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

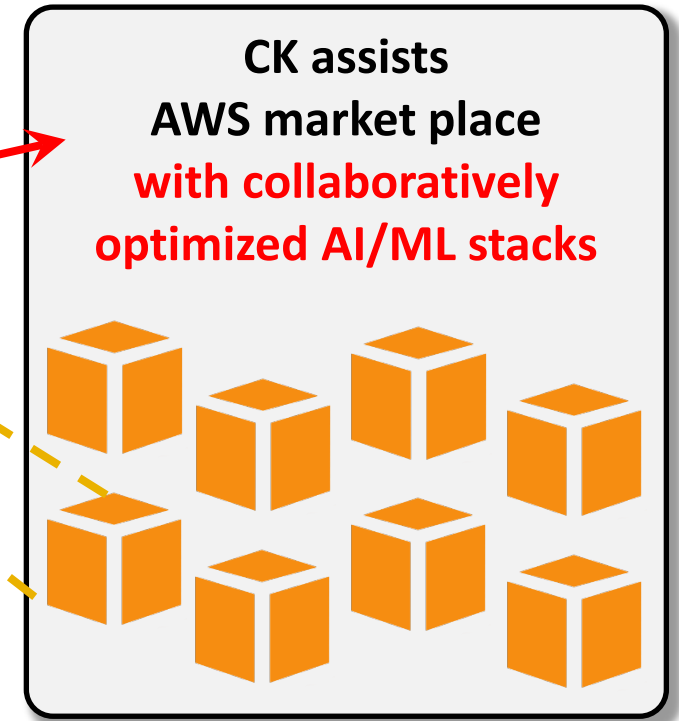
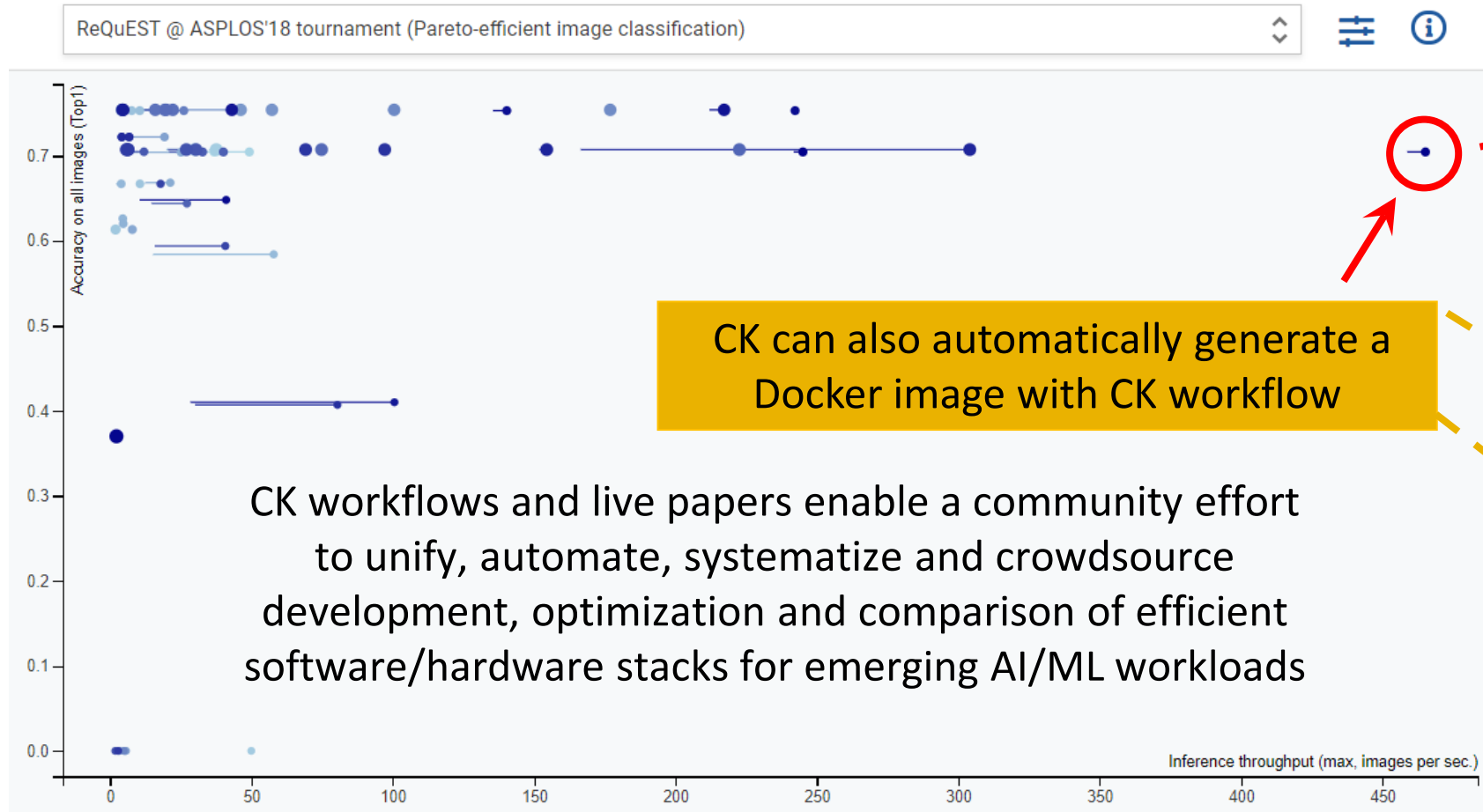
cknowledge.io/results



DevOps and FAIR principles made it easier to adopt research in production

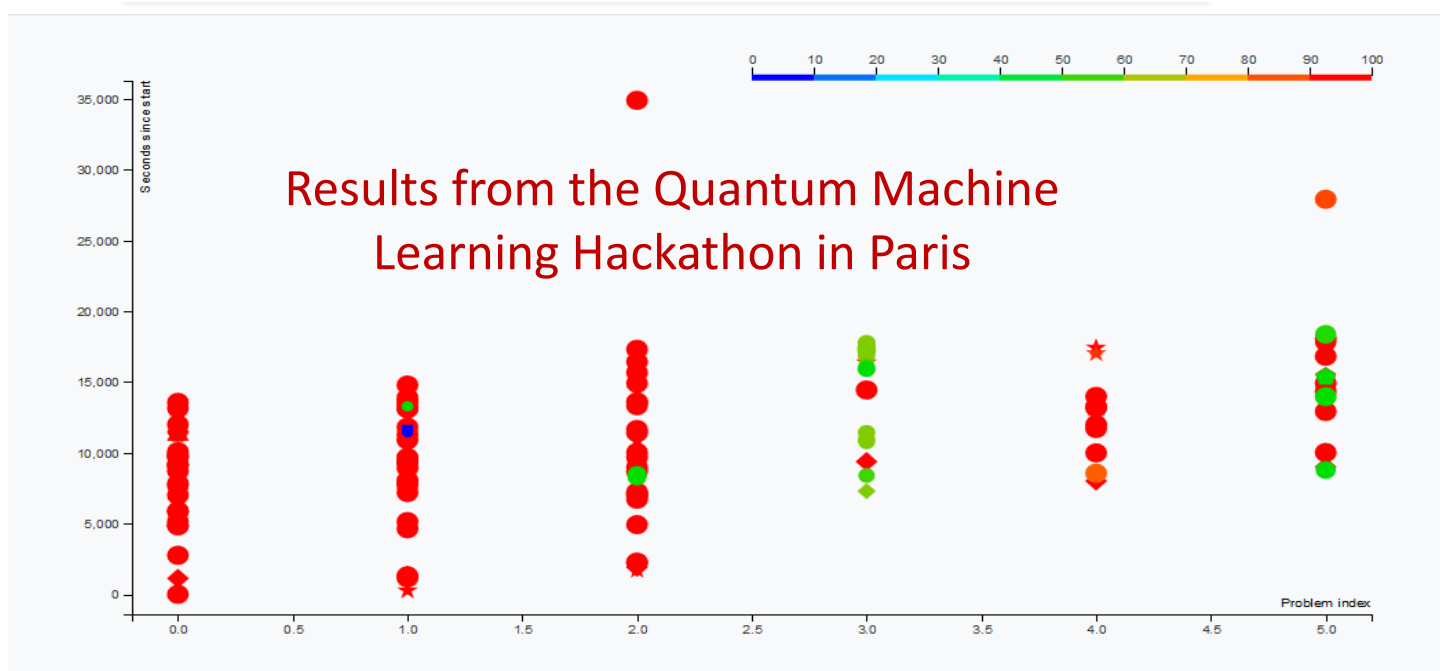
Colleagues from Amazon tested and reused REQUEST workflows, ported them to the Amazon cloud and used CK API and JSON meta to connect them with Amazon SageMaker

conferences.oreilly.com/artificial-intelligence/ai-eu-2018/public/schedule/detail/71549.html



Quantum ML hackathons using CK workflows and dashboards

cknowledge.org/quantum



#	Problem index	Timestamp (UTC)	Team name	Training time (sec)	Training accuracy	Test accuracy	Solution's rank	Source code	Quantum circuit
#1	4	Sun Jan 27 12:19:42 2019	Optimize, adapt, overcome	47.20	100.0	100.0	1	continuous_solver	Show circuit
#2	4	Sun Jan 27 12:52:49 2019	prevision.io	80.68	100.0	100.0	2	continuous_solver	Show circuit
#3	4	Sun Jan 27 13:21:31 2019	rebecca	171.54	100.0	100.0	3	continuous_solver	Show circuit

cknowledge.io/reproduced-results

RIVERLANE

IBM

rigetti

ThoughtWorks®

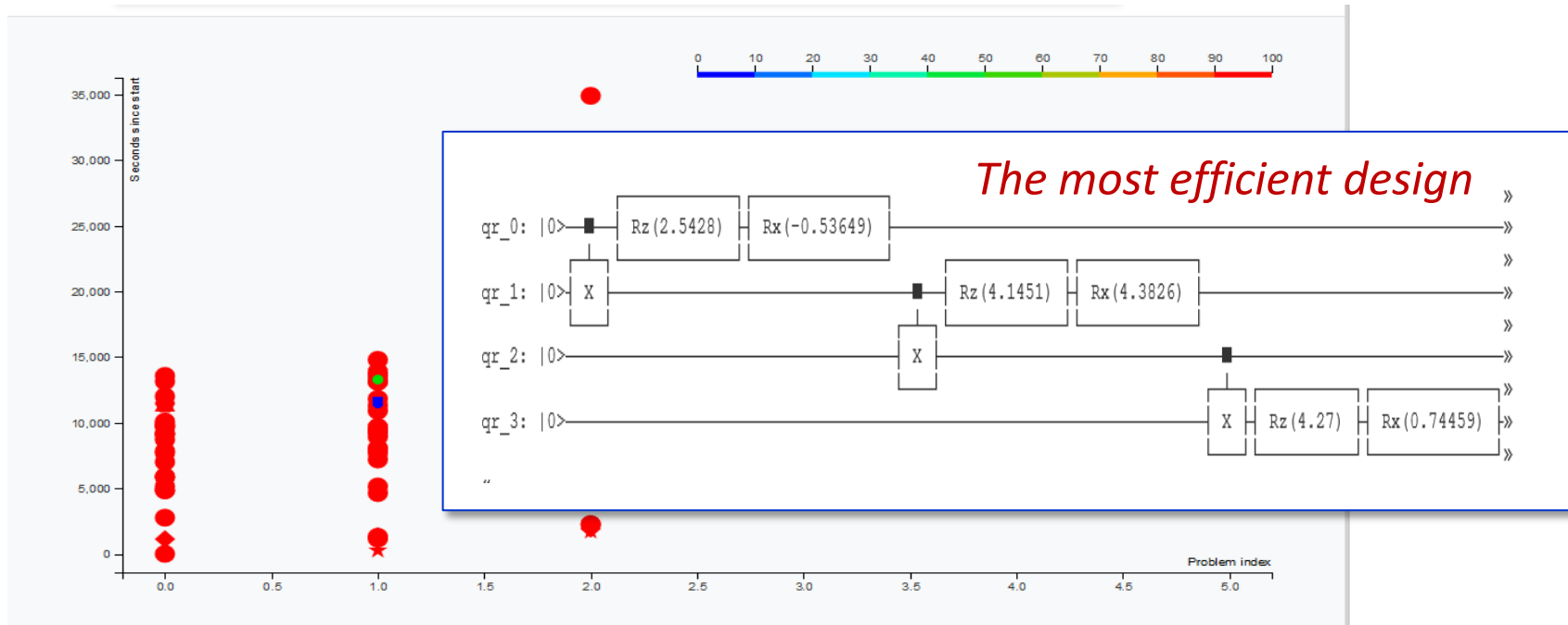
QUANTO
NATION

$$\frac{d\vec{v}}{dt}$$

Innovate UK

Quantum ML hackathons using CK workflows and dashboards

cKnowledge.org/quantum



#	Problem index	Timestamp (UTC)	Team name	Training time (sec)	Training accuracy	Test accuracy	Solution's rank	Source code	Quantum circuit
#1	4	Sun Jan 27 12:19:42 2019	Optimize, adapt, overcome	47.20	100.0	100.0	1	continuous_solver	Show circuit
#2	4	Sun Jan 27 12:52:49 2019	prevision.io	80.68	100.0	100.0	2	continuous_solver	Show circuit
#3	4	Sun Jan 27 13:21:31 2019	rebecca	171.54	100.0	100.0	3	continuous_solver	Show circuit

cKnowledge.io/reproduced-results

RIVERLANE



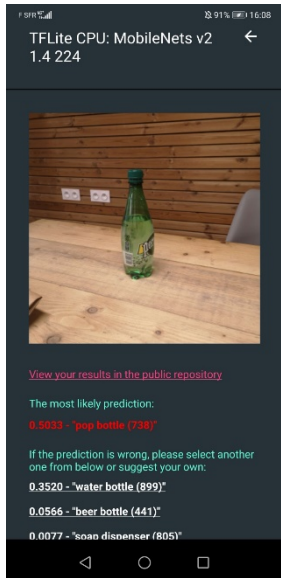
ThoughtWorks®



$$\frac{d\vec{v}}{dt}$$

Innovate UK

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+**

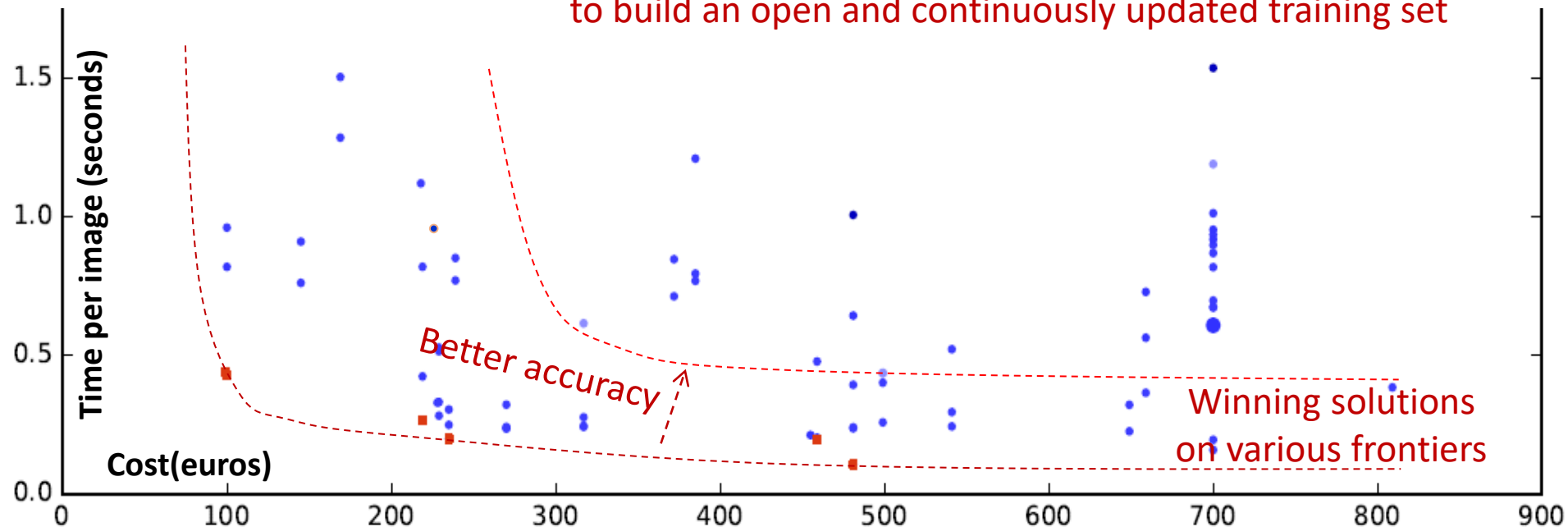
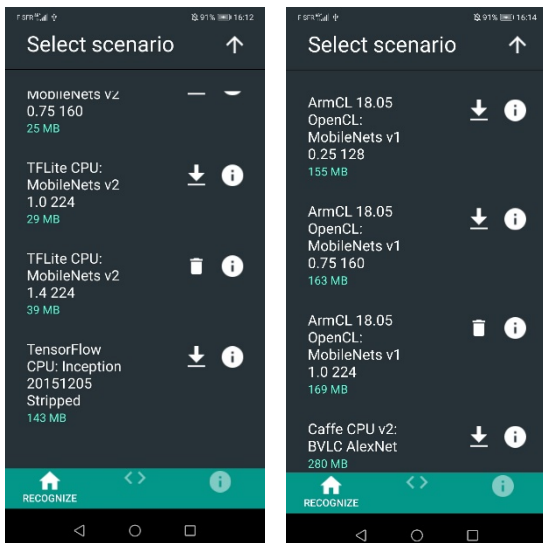
The number of distinct GPUs: **110+**

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



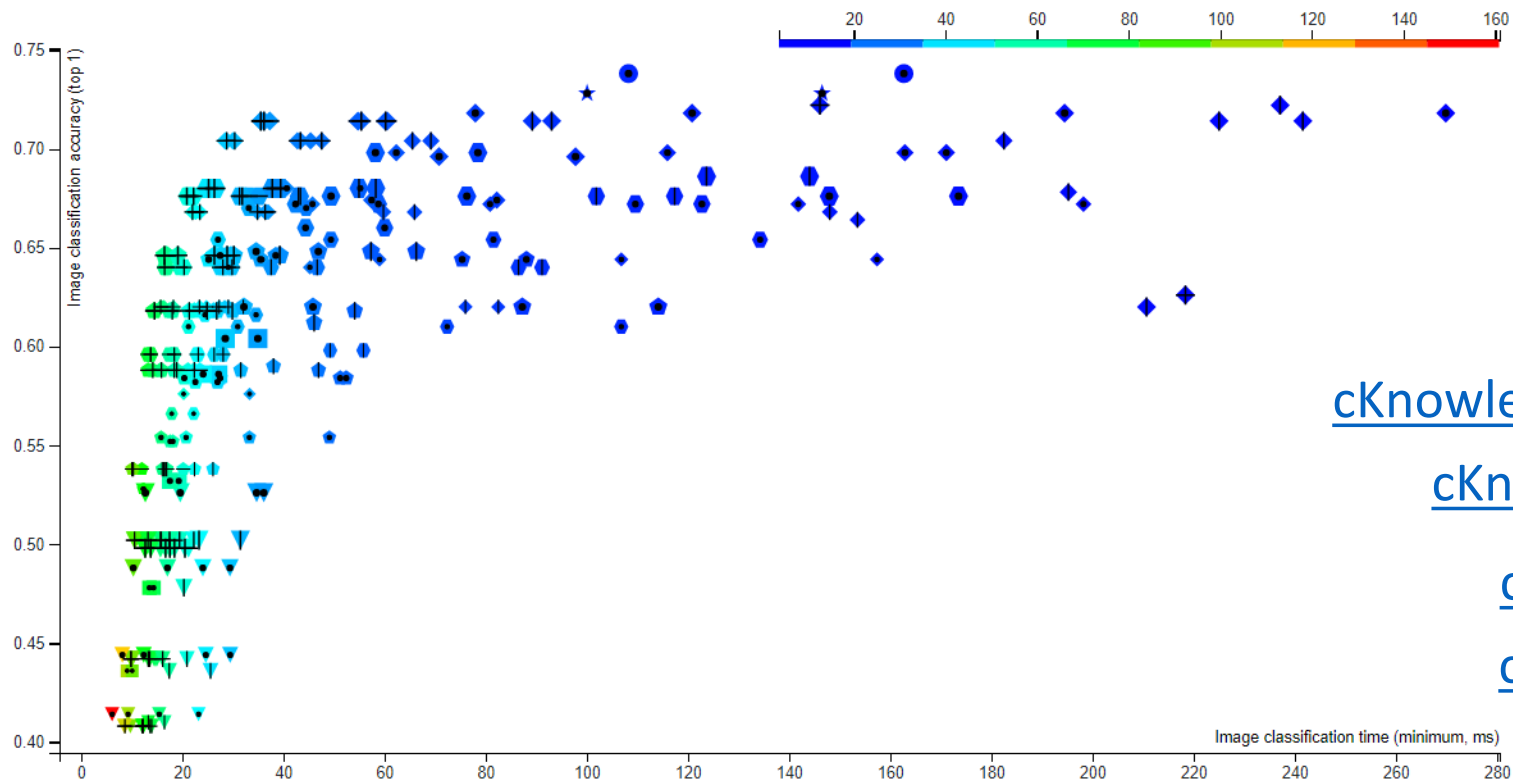
Crowd-tune whole ML/SW/HW stacks

“MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”

(Andrew G. Howard et al., 2017, <https://arxiv.org/abs/1704.04861>):

- 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.

[Customize graph](#) [Graph info](#)



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

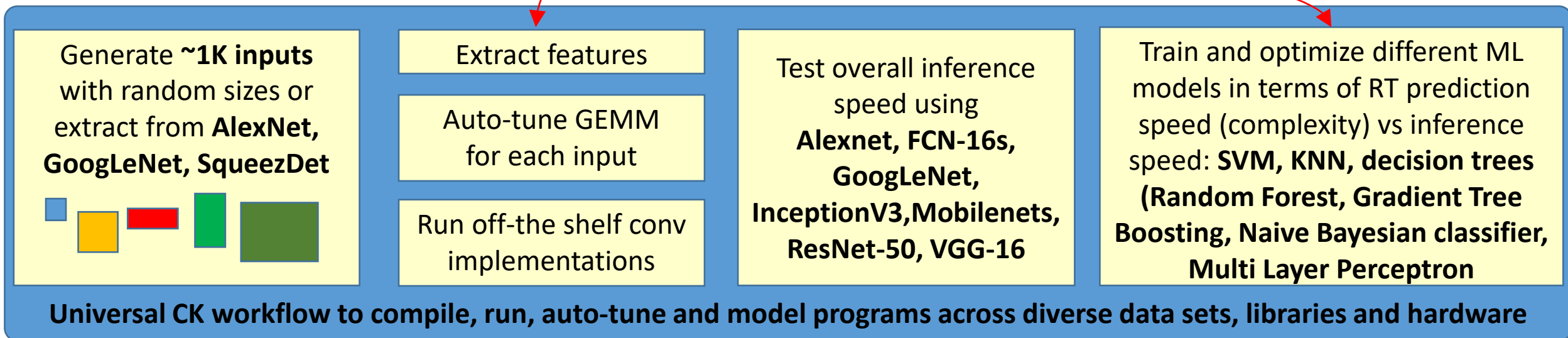
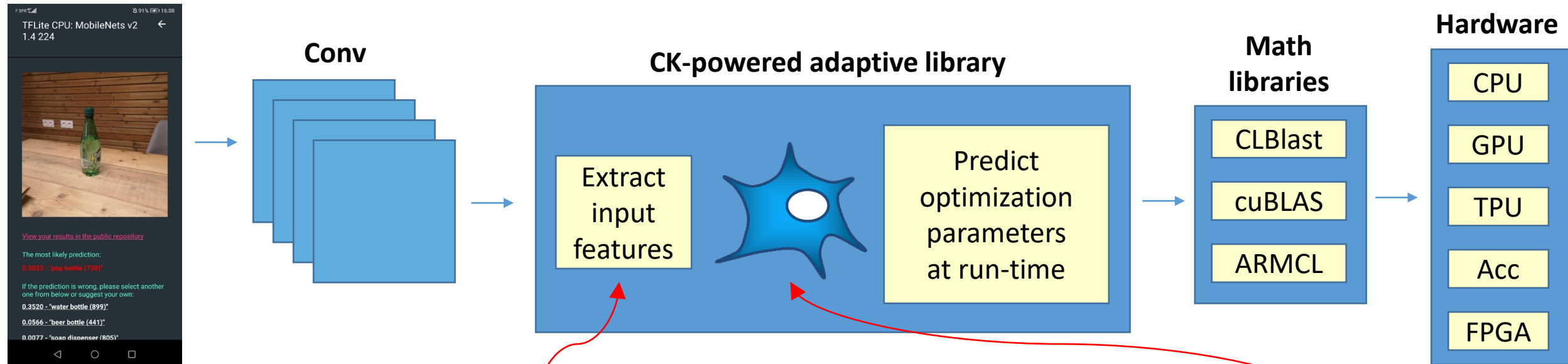
cKnowledge.io/crowdsorce-ml-sw-hw-co-design

cKnowledge.io/c/lib/9a927e4ce9be41b4

cKnowledge.io/reproduced-results

cKnowledge.io/reproduced-papers

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

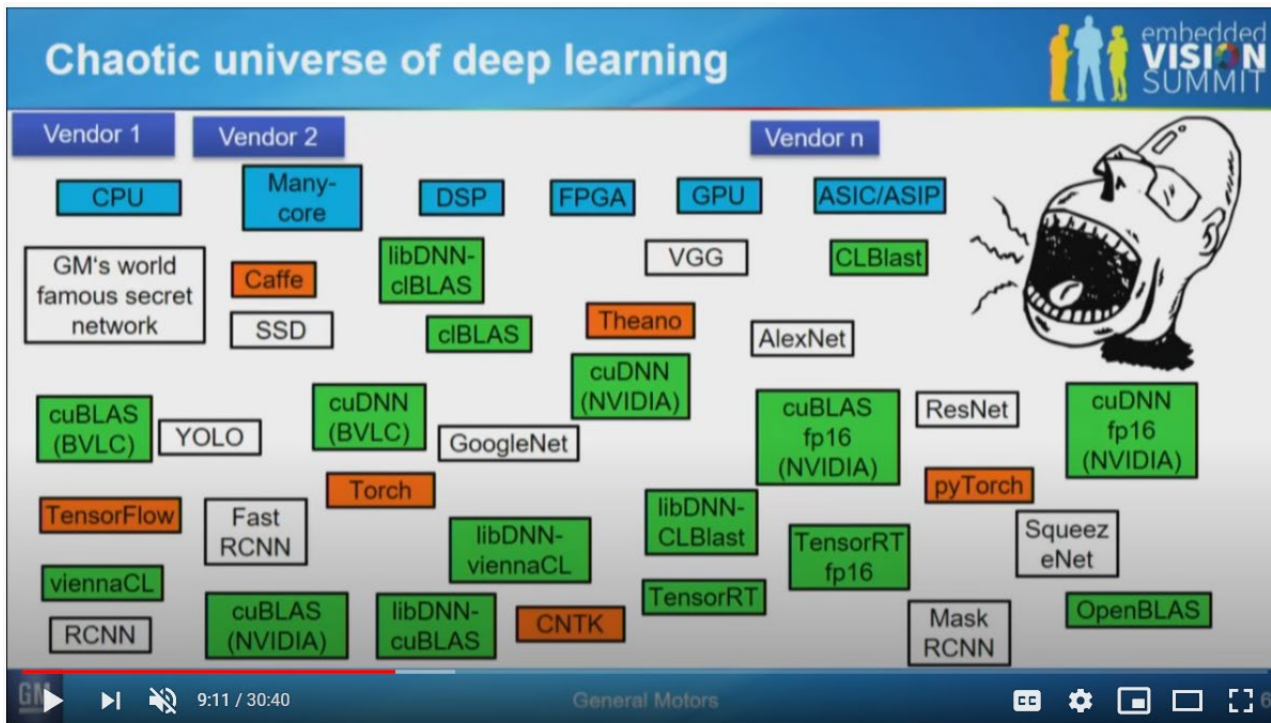


“On the Anatomy of Predictive Models for Accelerating GPU Convolution Kernels and Beyond”, ACM TACO, January 2021

doi.org/10.1145/3434402

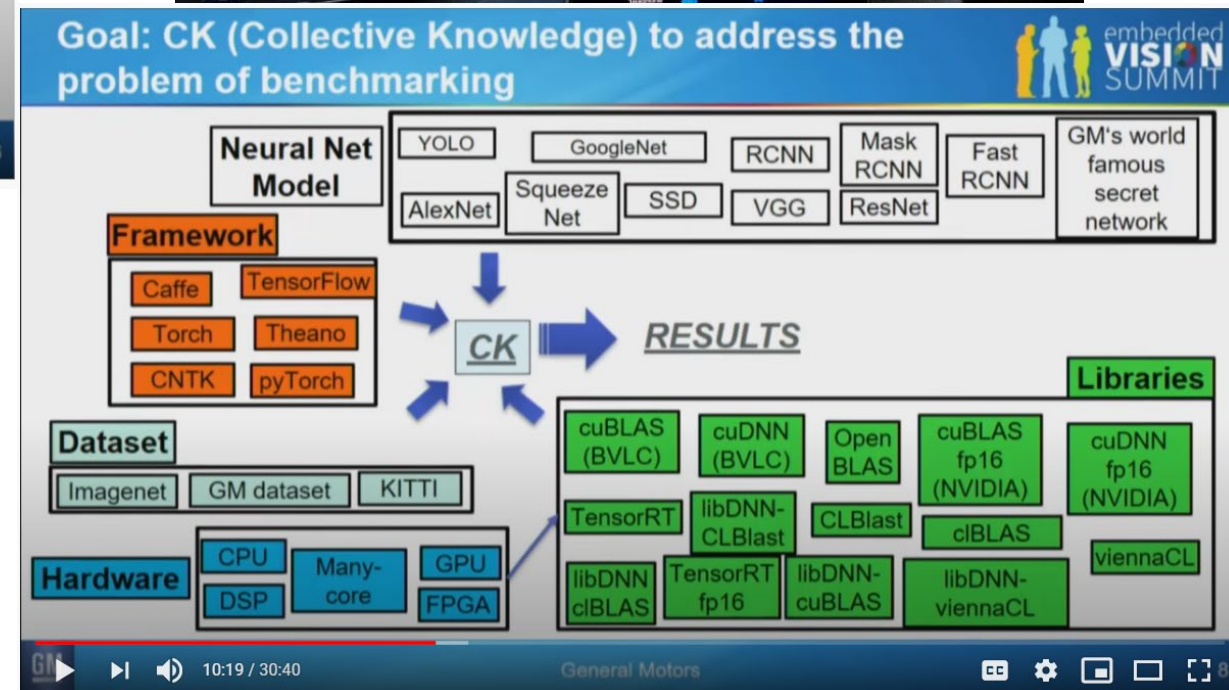
cknowledge.io/nn-components

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



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

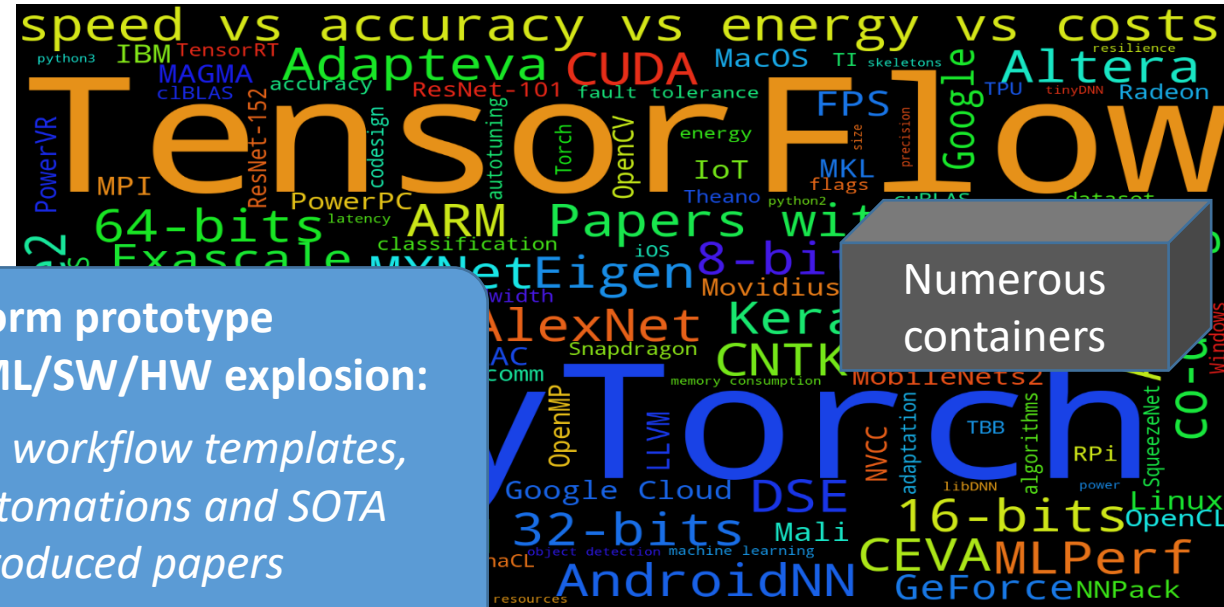
cknowledge.io/reproduced-results

Conclusions: bridging the growing gap between research and production

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



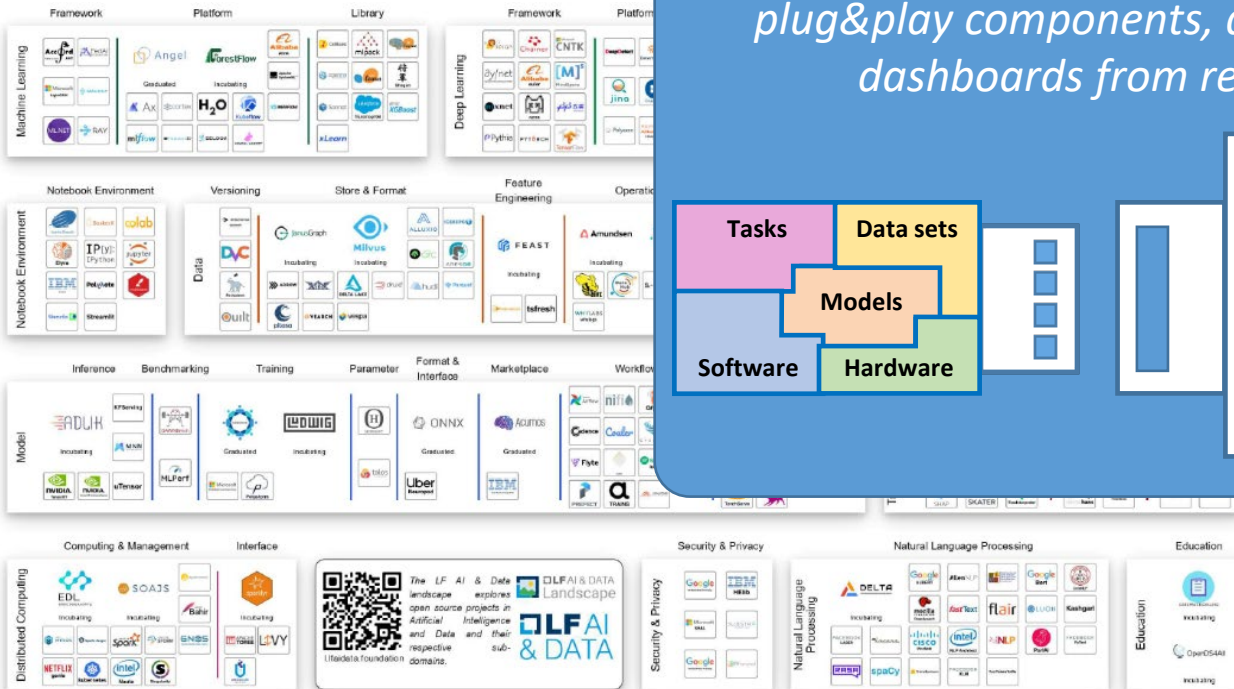
Continuously changing SW/HW/ML from different vendors



Numerous
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

Numerous tools and libraries



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

Sam Ainsworth, Erik Altman, Lorena Barba, Victor Bittorf, Unmesh D. Bordoloi, Steve Brierley, Luis Ceze, Milind Chabbi, Bruce Childers, Nikolay Chunosov, Marco Cianfriglia, Albert Cohen, Cody Coleman, Chris Cummins, Jack Davidson, Alastair Donaldson, Achi Dosanjh, Thibaut Dumontet, Debojyoti Dutta, Daniil Efremov, Nicolas Essayan, Todd Gamblin, Leo Gordon, Wayne Graves, Christophe Guillon, Herve Guillou, Stephen Herbein, Michael Heroux, Patrick Hesse, James Hetherington, Kenneth Hoste, Robert Hundt, Ivo Jimenez, Tom St. John, Timothy M. Jones, David Kanter, Yuriy Kashnikov, Gaurav Kaul, Sergey Kolesnikov, Shriram Krishnamurthi, Dan Laney, Andrei Lascu, Hugh Leather, Wei Li, Anton Lokhmotov, Peter Mattson, Thierry Moreau, Dewey Murdick, Mircea Namolaru, Luigi Nardi, Cedric Nugteren, Michael O'Boyle, Ivan Ospio, Bhavesh Patel, Gennady Pekhimenko, Massimiliano Picone, Ed Plowman, Ramesh Radhakrishnan, Ilya Rahkovsky, Vijay Janapa Reddi, Vincent Rehm, Catherine Roderick, Alka Roy, Shubhadeep Roychowdhury, Dmitry Savenko, Aaron Smith, Jim Spohrer, Michel Steuwer, Victoria Stodden, Robert Stojnic, Michela Taufer, Stuart Taylor, Olivier Temam, Eben Upton, Nicolas Vasilache, Flavio Vella, Davide Del Vento, Boris Veytsman, Alex Wade, Pete Warden, Dave Wilkinson, Matei Zaharia, Alex Zhigarev

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

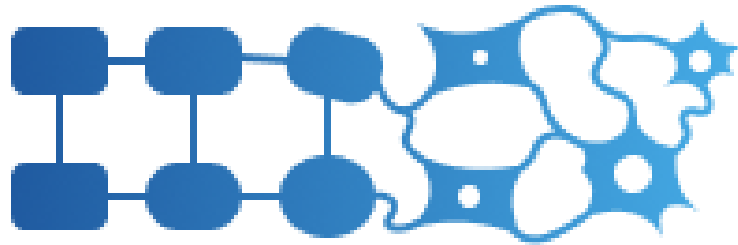
Co-organizers of CK-based hackathons and tournaments: cKnowledge.io/events

CK collaborators: cKnowledge.io/partners

Microsoft for Azure sponsorship

Thank you!

cKnowledge.io/@gfursin



Recent papers

- “Collective Knowledge: organizing research projects as a database of reusable components and portable workflows with common interfaces”.
To appear in Philosophical Transactions of the Royal Society A, March 2021 Pre-print: arxiv.org/pdf/2011.01149.pdf
- “On the Anatomy of Predictive Models for Accelerating GPU Convolution Kernels and Beyond”
ACM TACO (Transactions on Architecture and Code Optimization), January 2021 ACM DL: doi.org/10.1145/3434402
- “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>

ACM Reproducibility Task Force:

<https://www.acm.org/publications/task-force-on-data-software-and-reproducibility>