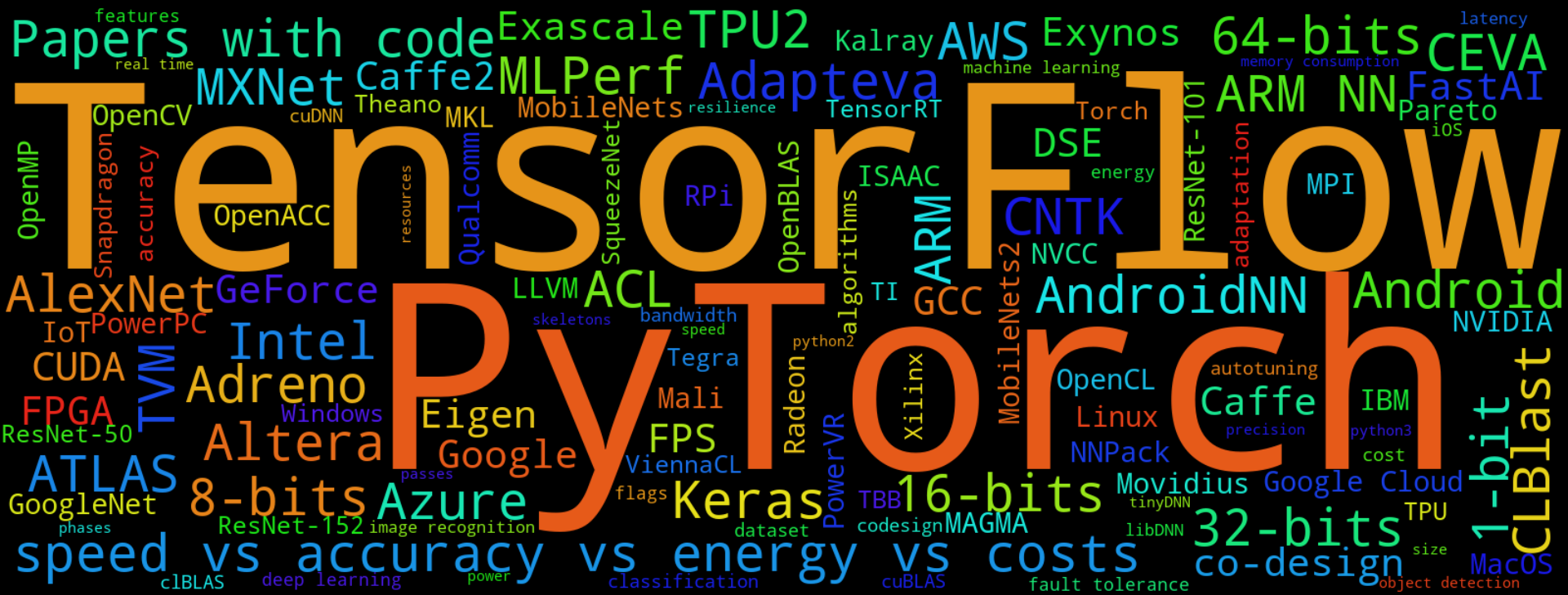
The Collective Knowledge project

connecting researchers and practitioners
to make it easier to reproduce the SOTA ML/AI/systems research
and deploy efficient systems in production



or our community attempt to bring order to AI/ML/systems chaos

Grigori Fursin, the founder of cKnowledge.org and cKnowledge.io

Many groups are working to co-design efficient AI / ML / SW / HW stacks

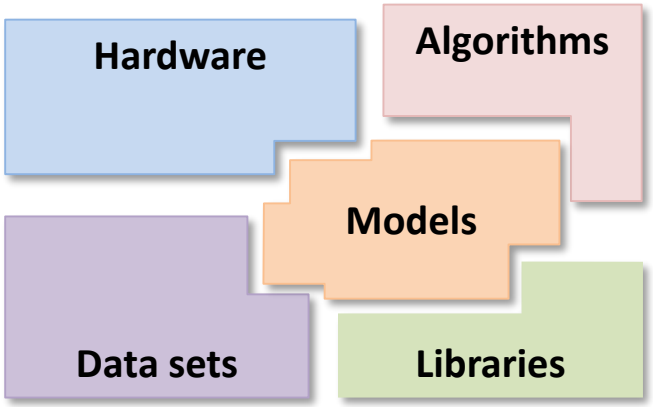
AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

AI models
Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

Efficient AI/ML system must be very carefully co-designed



for various form factors
(IoT, mobile, data centers)

while trading off multiple constraints
(accuracy, speed, energy, size, costs)

and maximizing ROI
(faster time to market, R&D sustainability, much better than all competitors)

Helping the society

Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

90K+ AI / ML / SW / HW papers are published each year!

AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

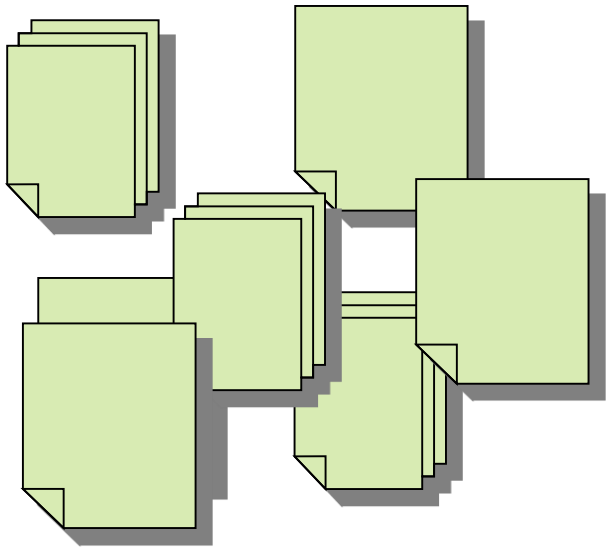
AI models
Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

→

Numerous papers with ad-hoc code



Numerous models, data sets, benchmarks, libraries and tools

Multiple competitions focusing mostly on accuracy (Kaggle, DawnBench)

A few benchmarks an competitions focusing on optimizing other metrics besides accuracy:

LPIRC, MLPerf

→

Helping the society

Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

90K+ AI / ML / SW / HW papers are published each year!

AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

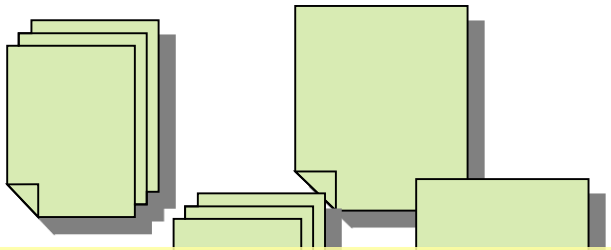
AI models
Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

→

Numerous papers with ad-hoc code



Can we now co-design efficient SW/HW/ML stacks and use them in production to support real-world applications?

Multiple competitions focusing mostly on accuracy (Kaggle, DawnBench)

A few benchmarks an competitions focusing on optimizing other metrics besides accuracy:

LPIRC, MLPerf

→

Helping the society

Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

The adoption of novel AI / ML techniques in production is extremely slow

AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

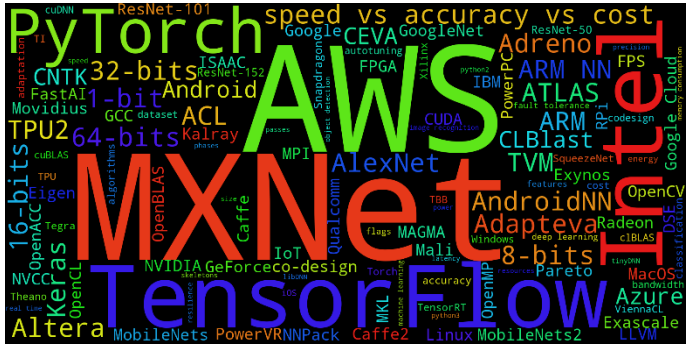
AI models

Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

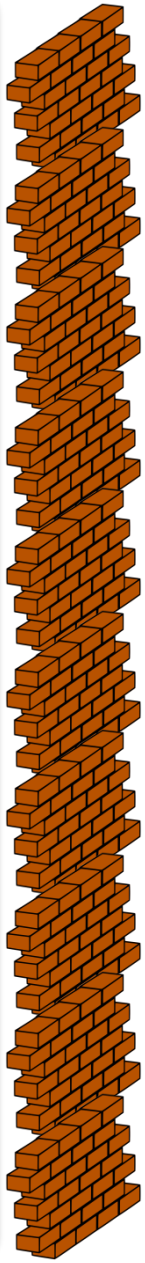
AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

- Technological chaos: continuously changing algorithm/model/SW/HW stack



- Non-representative / outdated training sets
- No common experimental frameworks and established methodologies which can adapt to this chaos
- Numerous reproducibility issues
- Very little artifact reuse in 1000+ ML papers
- Very little tech. transfer from academia (toy examples and too many papers)



Helping the society

Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

The adoption of novel AI / ML techniques in production is extremely slow

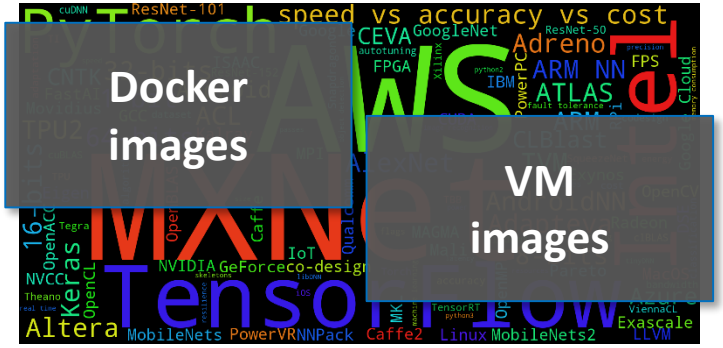
AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

AI models
Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

- Technological chaos: continuously changing algorithm/model/SW/HW stack



- Non-representative / outdated training sets
- No common experimental frameworks and established methodologies which can adapt to this chaos
- Numerous reproducibility issues
- Very little artifact reuse in 1000+ ML papers
- Very little tech. transfer from academia (toy examples and too many papers)
- Docker, Kubernetes and VM images hide the mess but do not solve above problems

Public outcry about reproducibility, portability and reusability crisis

Helping the society

Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

Many great tools, data sets and models to help researchers …

Applications
- Meteorology
- Health
- Robotics
- Automotive
- Economics
- Physics
- Astronomy
- Education

Scientific tools
- MATLAB
- Scilab
- Simulink
- LabVIEW
- Gnuplot
- LaTeX
- Ipython

Build tools
- Make
- Cmake
- SCons
- Bazel
- Gradle
- Ninja

Languages
- C++
- C#
- C
- Go
- PHP
- Fortran
- Java
- Python

Compilers
- LLVM
- GCC
- Intel
- PGI
- TVM
- CUDA

DevOps tools
- Git
- Jenkins
- Docker
- Kubernetes
- Singularity

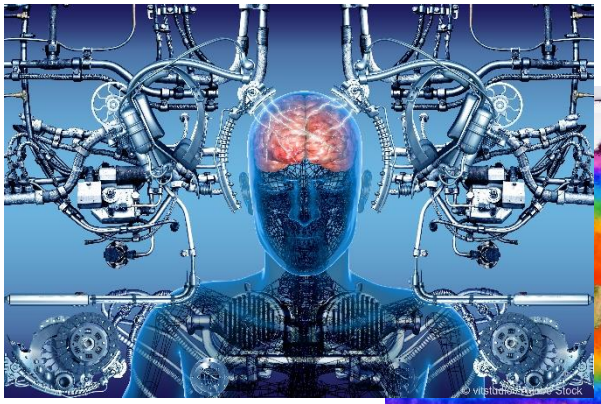
Package managers
- Anaconda
- Go
- Npm
- Pip
- Sbt
- dpkg
- Spack
- EasyBuild

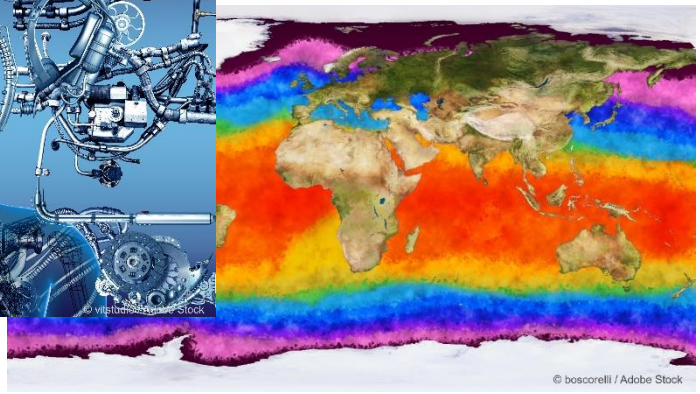
Libraries
- SciPy
- TFLite
- OpenBLAS
- MAGMA
- cuDNN
- cuFFT
- ArmNN
- CLBlast
- gemmlowp
- Boost
- HDF5
- MPI
- OpenCV
- Protobuf

OS
- Linux
- MacOS
- BSD
- Windows
- Android

Shells
- bash
- sh
- csh
- ksh
- Windows shell

Programs
- Image classification
- Object detection
- Natural Language processing
- Text processing
- Video processing
- Personal assistant





Benchmarks
- SPEC
- EEMBC
- HPCG
- LINPACK
- cBench
- MLPerf

Hardware
- CPU
- GPU
- TPU / NN
- DSP
- FPGA
- Quantum
- Simulators
- Interconnects

AI/ML frameworks
- TensorFlow
- PyTorch
- MXNet
- Caffe
- MCT (CNTK)
- Keras
- Kubeflow
- AutoML
- SageMaker
- Apache Spark

Models
- GoogleNet
- AlexNet
- VGG
- ResNet
- MobileNets
- SSD
- SqueezeNet
- DeepSpeech

Datasets
- ImageNet
- KITTI
- COCO
- MiDataSets
- Human Cell Atlas
- 1000 Genomes
- Earth models
- OpenStreetMap

Workload managers
- MPI
- SLURM
- PBS
- FLUX

Web services
- GitHub
- GitLab
- BitBucket
- Travis
- JupyterHub
- Codelabs
- SageMaker

Databases / experiments
- MySQL
- PostgreSQL
- MongoDB
- CouchDB
- Text files
- JSON files
- XLS files

Knowledge sharing
- ArXiv
- ACM DL
- IEEE DL
- GitHub
- Zenodo
- FigShare
- Web pages

Platforms
- HPC
- Desktops
- IoT
- Mobile
- Cloud sevices

… but it's not easy to connect them together into reproducible AI / ML workflows

Applications
- Meteorology
- Health
- Robotics
- Automotive
- Economics
- Physics
- Astronomy
- Education

Scientific tools
- MATLAB
- Scilab
- Simulink
- LabVIEW
- Gnuplot
- LaTeX
- Ipython

Build tools
- Make
- Cmake
- SCons

Languages
- C++
- C#
- C
- Go

Compilers
- LLVM
- GCC
- Intel
- PGI

DevOps tools
- Git
- Jenkins
- Docker

Package managers
- Anaconda
- Go
- Npm

Libraries
- SciPy
- TFLite
- OpenBLAS
- MAGMA
- cuDNN
- cuFFT
- ArmNN
- CLBlast
- gemmlowp
- Boost
- HDF5
- MPI
- OpenCV
- Protobuf

OS
- Linux
- MacOS
- BSD
- Windows
- Android

Shells
- bash
- sh
- csh
- ksh
- Windows shell

Programs
- Image classification
- Object detection
- Natural Language processing
- Text processing
- Video processing
- Personal assistant

AI/ML frameworks
- TensorFlow
- PyTorch
- MXNet
- Caffe
- MCT (CNTK)
- Keras
- Kubeflow
- AutoML
- SageMaker
- Apache Spark

Models
- GoogleNet
- AlexNet
- VGG
- ResNet
- MobileNets
- SSD
- SqueezeNet
- DeepSpeech

Benchmarks
- SPEC
- EEMBC
- HPCG
- LINPACK
- cBench
- MLPerf

Hardware
- CPU
- GPU
- TPU / NN
- DSP
- FPGA
- Quantum
- Simulators
- Interconnects

Knowledge sharing
- ArXiv
- ACM DL
- IEEE DL
- GitHub
- Zenodo
- FigShare
- Web pages

Platforms
- HPC
- Desktops
- IoT
- Mobile
- Cloud sevices

- MiDataSets
- Human Cell Atlas
- 1000 Genomes
- Earth models
- OpenStreetMap

managers
- MPI
- SLURM
- PBS
- FLUX

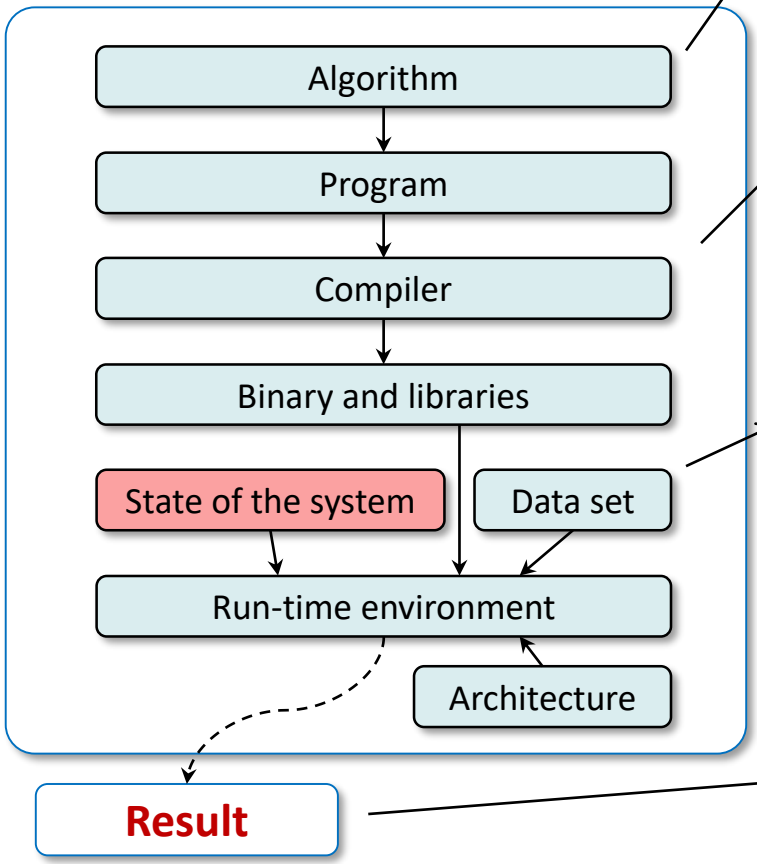
- GitLab
- BitBucket
- Travis
- JupyterHub
- Codelabs
- SageMaker

- PostgreSQL
- MongoDB
- CouchDB
- Text files
- JSON files
- XLS files

What I've noticed when reproducing papers at ACM and IEEE conferences since 2014

Authors develop their own ad-hoc scripts to do exactly the same "actions" across nearlly all artifact submissions:

- Detect target hardware properties
- Detect software dependencies
- Install missing packages (code/data)
- Build code; run experiments
- Plot graphs and validate results
- Generate papers



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

- image corner detection
- matmul OpenCL
- compression
- neural network CUDA

Have some common meta: *which datasets can use, how to compile, CMD, …*

Ad-hoc scripts to install packages or set up environment for code and data deps on a given platform

- GCC V8.1
- LLVM V7.0
- Intel Compilers 2017

Have some common meta: *compilation, linking and optimization flags*

Ad-hoc dirs for data sets with some ad-hoc scripts to find them, extract features, etc

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

Have some (common) meta: *filename, size, width, height, colors, …*

Ad-hoc dirs and scripts to record and analyze experiments

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

Have some common meta: *features, characteristics, optimizations*

cTuning.org/ae

What I've noticed when reproducing papers at ACM and IEEE conferences since 2014

Authors develop their own ad-hoc scripts to do exactly the same "actions" across nearlly all artifact submissions:

- Detect ta
- Detect so
- Install mi
- Build coc
- Plot and
- Generate

Ad-hoc scripts to compile and

image corner detection

Have some common meta:

ets to ... e eta: n, d on e ze, nt, a:

State

Result

and analyze experiments

xls table with graph

characteristics, optimizations

The reason I started developing the open-source Collective Knowledge framework (CK) was

to help researchers share their artifacts
(code, data sets, models, scripts, experiments, papers)
as reusable, portable and customizable packages and workflows
with a simple Python API, CLI and JSON meta description.

I needed CK to support reproducibility initiatives
and help the community crowd-benchmark published techniques
across diverse software, hardware, data sets and models,
collaboratively reproduce and compare results,
and help companies quickly test and deploy new techniques
in production while supporting any technology and legacy code

cKnowledge.org

CK concept : create, share and reuse automation "actions" as Python API, CLI and JSON

Such approach helped us to apply standard DevOps practices and Continuous Integration to research and experimentation!

Python module
"program"
with functions:
compile and run

Python module
"soft"
with function:
detect

Python module
"dataset"
with function:
extract_features

Python module
"experiment"
with function:
add, get, analyze

data UID and alias

- image corner detection
- matmul OpenCL
- compression
- neural network CUDA

meta.json

- GCC V8.1
- LLVM V7.0
- Intel Compilers 2017

meta.json

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

meta.json

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

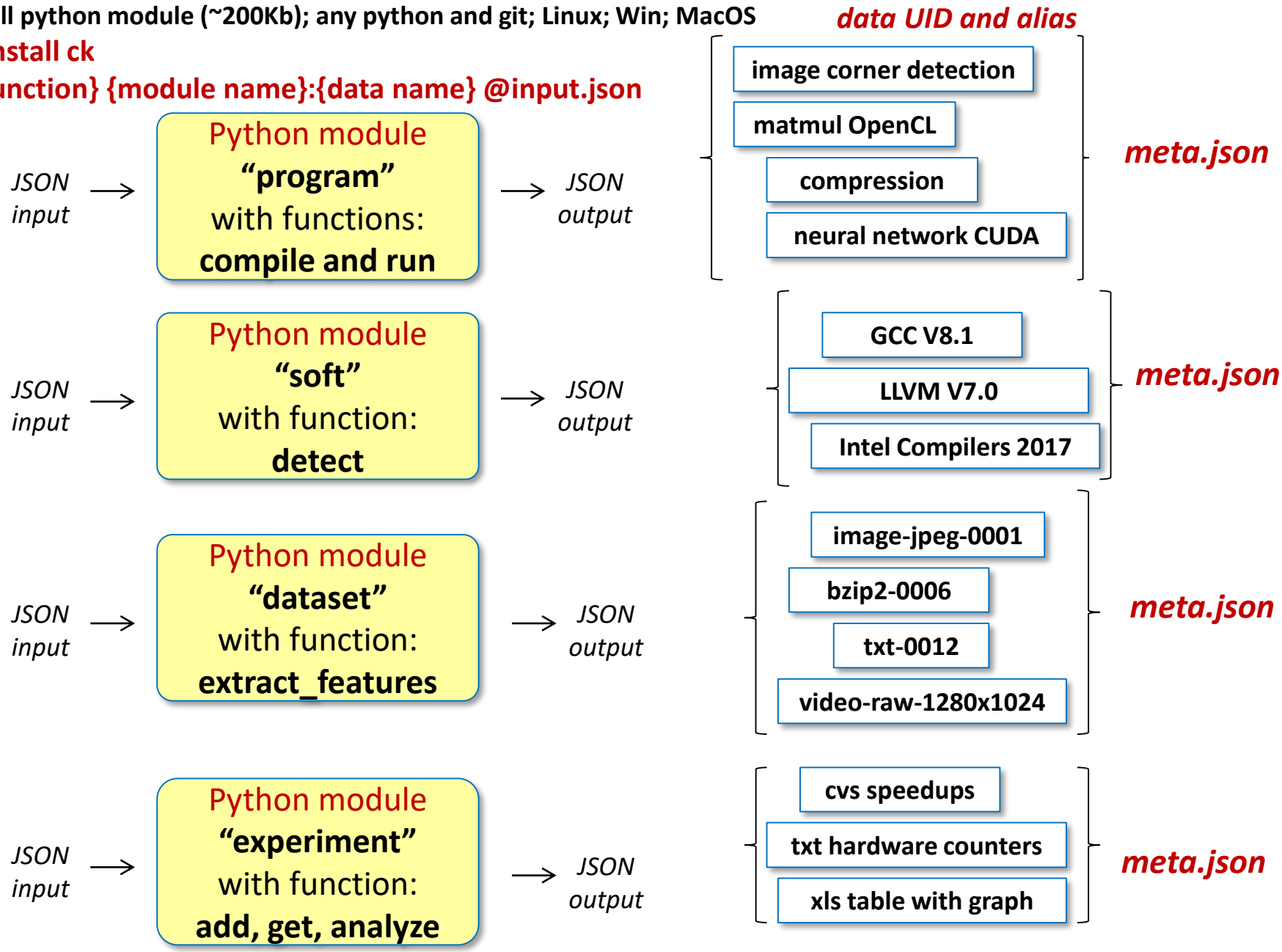
meta.json

CK framework: simple CLI to create and access APIs (very portable - minimal dependencies)

CK: small python module (~200Kb); any python and git; Linux; Win; MacOS

$ pip install ck

$ ck {function} {module name}:{data name} @input.json

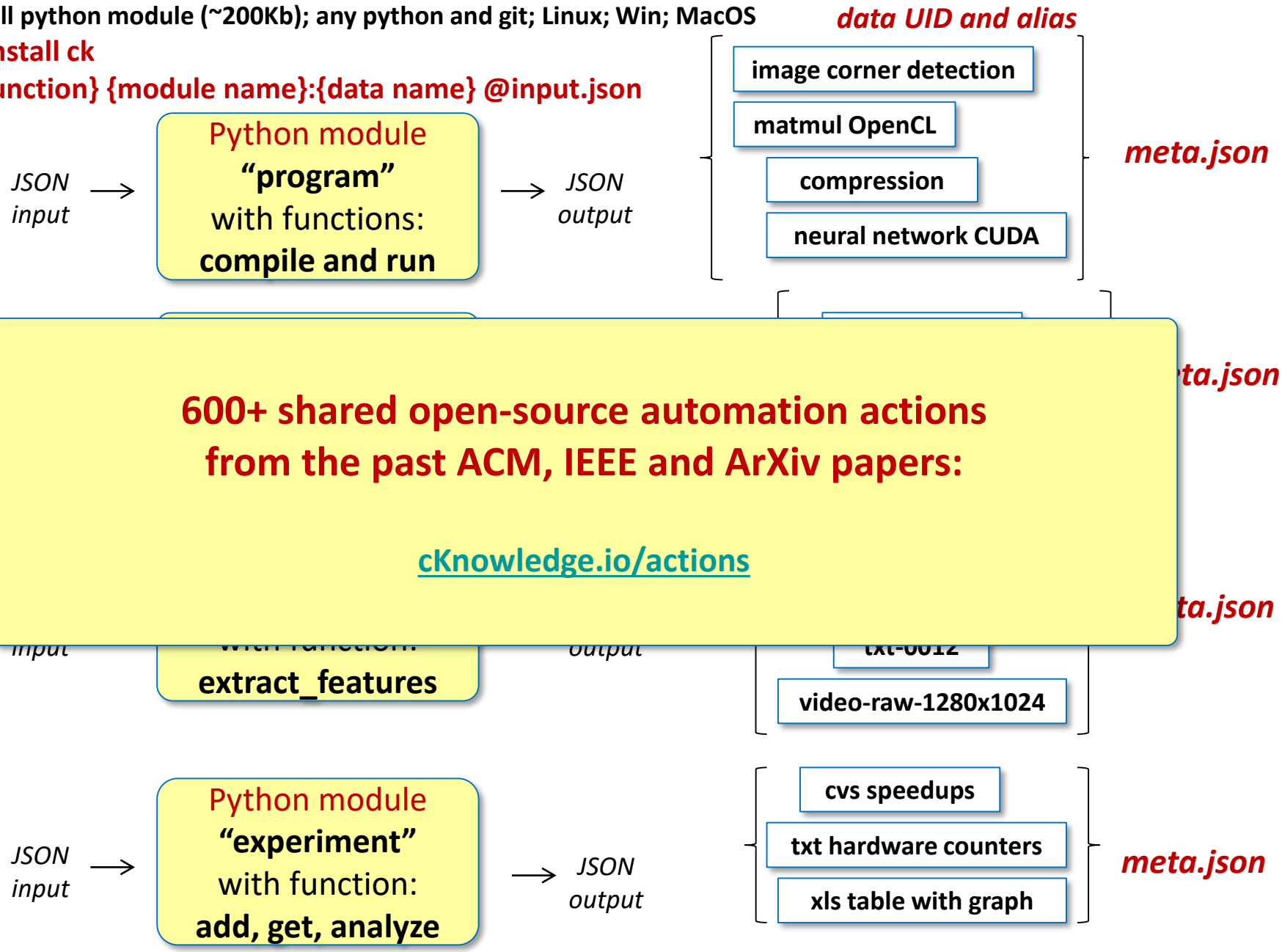


CK framework: simple CLI to create and access APIs (very portable - minimal dependencies)

CK: small python module (~200Kb); any python and git; Linux; Win; MacOS

$ pip install ck

$ ck {function} {module name}:{data name} @input.json



CK concept: assemble customizable workflows with JSON input and output

CK workflows describe dependencies on other CK packages using simple tags (code, data sets, models, frameworks, etc)

JSON input →

Python module
"program"
with functions:
compile and run

Python module
"soft"
with function:
detect

Python module
"dataset"
with function:
extract_features

Python module
"experiment"
with function:
add, get, analyze

→ *JSON output*

data UID and alias

- image corner detection
- matmul OpenCL
- compression
- neural network CUDA

meta.json

- GCC V8.1
- LLVM V7.0
- Intel Compilers 2017

meta.json

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

meta.json

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

meta.json

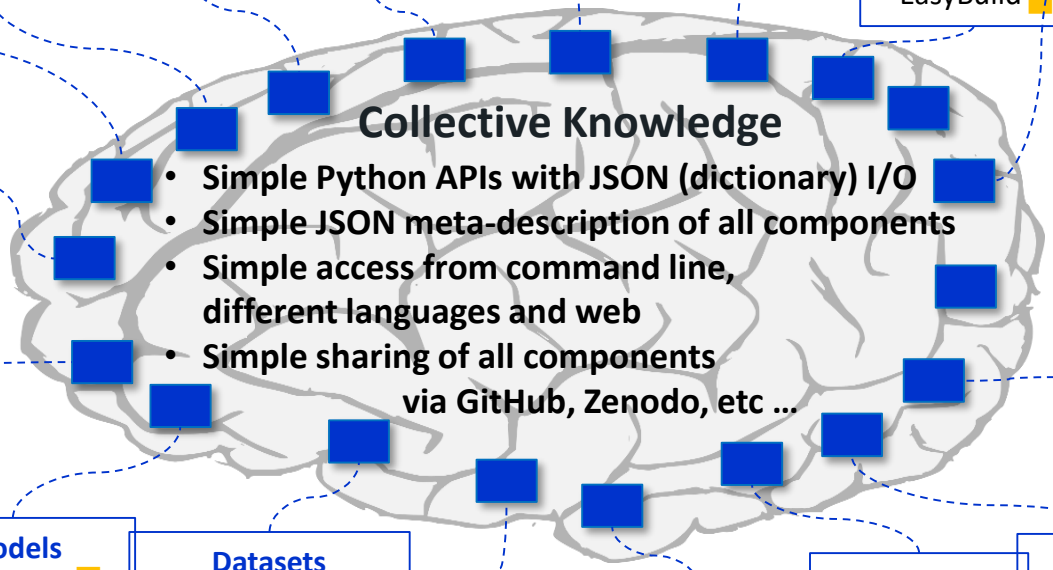
CK concept: provide simple and unified directory structure for shared artifacts

cKnowledge.io/repos

Collective Knowledge COMPATIBLE

/ 1st level directory – CK modules		/ 2nd level dir - CK entries	/ CK meta info
setup	soft	TensorFlow	JSON file
		PyTorch	JSON file
		ARM compute lib	JSON file
find	dataset	ImageNet	JSON file
		Car video stream	JSON file
extract features		Real surveillance camera	JSON file
compile	program	image classification	JSON file
		object detection	JSON file
run		GEMM OpenCL	JSON file
autotune		convolution CPU	JSON file
add	experiment	performance results	JSON file
		training / accuracy	JSON file
replay		bugs	JSON file
JSON API	Python module	holder for original artifact	CK meta

I started working with the community to gradually standardize all research artifacts



Applications
- Meteorology
- Health
- Robotics
- Automotive
- Economics
- Physics
- Astronomy
- Education

Scientific tools
- MATLAB
- Scilab
- Simulink
- LabVIEW
- Gnuplot
- LaTeX
- Ipython

Build tools
- Make
- Cmake
- SCons
- Bazel
- Gradle
- Ninja

Languages
- C++
- C#
- C
- Go
- PHP
- Fortran
- Java
- Python

Compilers
- LLVM
- GCC
- Intel
- PGI
- TVM
- CUDA

DevOps tools
- Git
- Jenkins
- Docker
- Kubernetes
- Singularity

Package managers
- Anaconda
- Go
- Npm
- Pip
- Sbt
- dpkg
- Spack
- EasyBuild

Libraries
- SciPy
- TFLite
- OpenBLAS
- MAGMA
- cuDNN
- cuFFT
- ArmNN
- CLBlast
- gemmlowp
- Boost
- HDF5
- MPI
- OpenCV
- Protobuf

OS
- Linux
- MacOS
- BSD
- Windows
- Android

Shells
- bash
- sh
- csh
- ksh
- Windows shell

Programs
- Image classification
- Object detection
- Natural Language processing
- Text processing
- Video processing
- Personal assistant

Benchmarks
- SPEC
- EEMBC
- HPCG
- LINPACK
- cBench
- MLPerf

Hardware
- CPU
- GPU
- TPU / NN
- DSP
- FPGA
- Quantum
- Simulators
- Interconnects

AI/ML frameworks
- TensorFlow
- PyTorch
- MXNet
- Caffe
- MCT (CNTK)
- Keras
- Kubeflow
- AutoML
- SageMaker
- Apache Spark

Models
- GoogleNet
- AlexNet
- VGG
- ResNet
- MobileNets
- SSD
- SqueezeNet
- DeepSpeech

Datasets
- ImageNet
- KITTI
- COCO
- MiDataSets
- Human Cell Atlas
- 1000 Genomes
- Earth models
- OpenStreetMap

Workload managers
- MPI
- SLURM
- PBS
- FLUX

Web services
- GitHub
- GitLab
- BitBucket
- Travis
- JupyterHub
- Codelabs
- SageMaker

Databases / experiments
- MySQL
- PostgreSQL
- MongoDB
- CouchDB
- Text files
- JSON files
- XLS files

Knowledge sharing
- ArXiv
- ACM DL
- IEEE DL
- GitHub
- Zenodo
- FigShare
- Web pages

Platforms
- HPC
- Desktops
- IoT
- Mobile
- Cloud sevices

I collaborate with CK partners to solve AI / ML portability and adaptation to any platform

1) **Describe different operating systems (Linux, Android, Windows, MacOS, etc)**

```
ck pull repo:ck-env
ck ls os
ck load os:linux-64 --min
```

We implemented and shared such automations to support real use cases:
cKnowledge.org/partners

2) **Detect and unify information about platforms**

```
ck detect platform --help
ck detect platform --out=json
ck load os:linux-64 --min
```

All automation actions are now available at cKnowledge.io

3) **Detect installed software** (code, data, models, scripts)

```
ck search soft --tags=dataset
ck detect soft:compiler.llvm
ck show env --tags=llvm
```

250+ software detection plugins:
cKnowledge.io/soft

4) **Install missing packages** (code, data, models, scripts)

```
ck search package --tags=dataset,imagenet
ck install package --tags=dataset,imagenet,2012,min
ck show env --tags=dataset
ck virtual env –tags=dataset,imagenet
```

600+ shared packages:
cKnowledge.io/packages

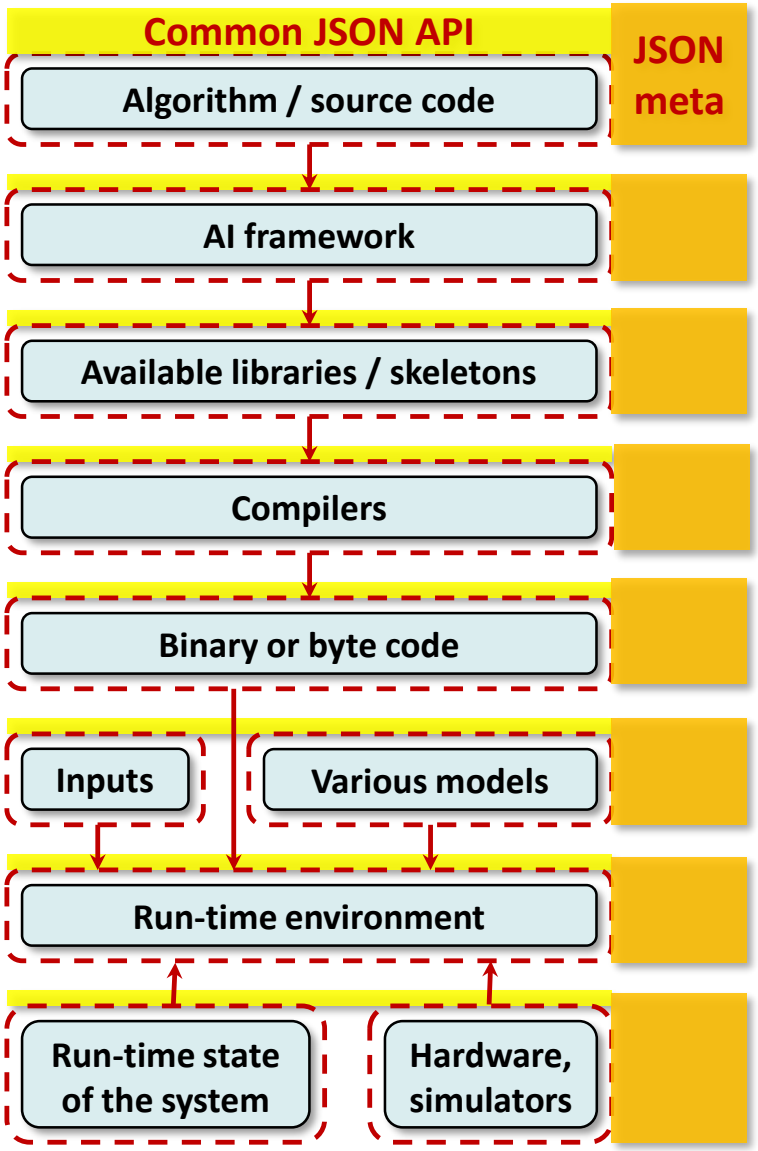
We now have reusable automation actions to adapt to any platform and environment while using containers to make stable snapshots

Artifact | automated and reusable

Collective Knowledge | COMPATIBLE

Workflow | CK

We started developing and sharing workflows for portable AI/ML/systems benchmarking



We developed a universal program workflow to compile, run, profile and autotune AI/ML applications across diverse models, data sets and platforms, validate results, record experiments, share and reproduce them, and report discrepancies

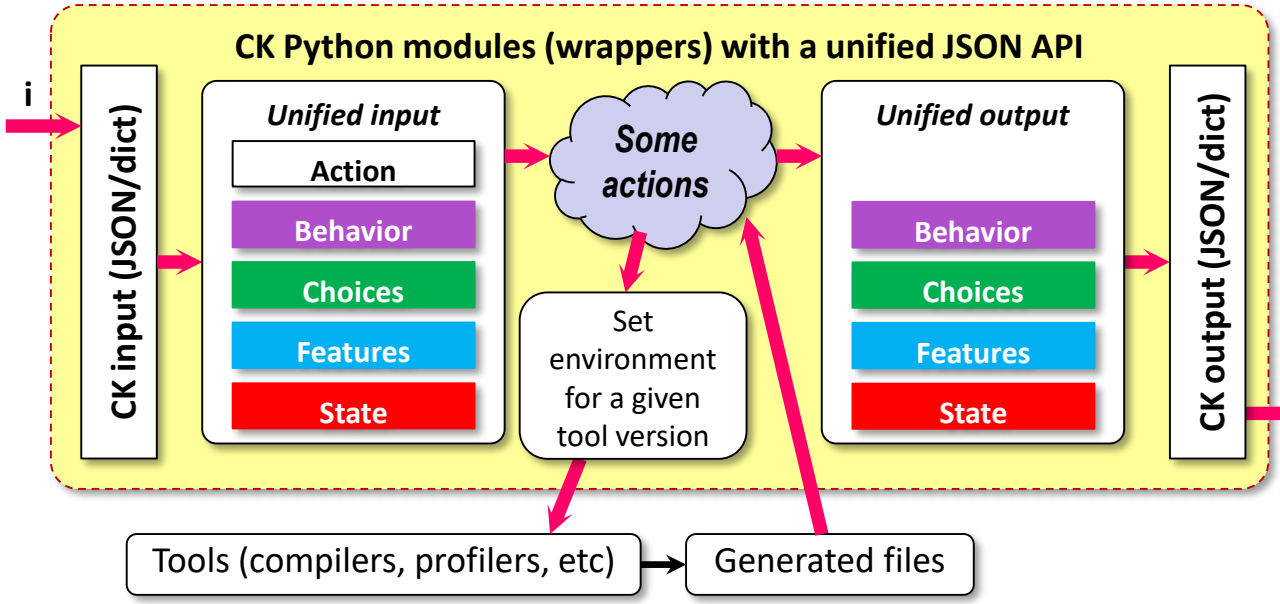
```
$ ck pull repo:ck-crowdtuning

$ ck ls program
$ ck ls dataset

$ ck load program:cbench-automotive-susan  --min
$ ck compile program:cbench-automotive-susan –fast

$ ck run program:cbench-automotive-susan

$ ck autotune program:cbench-automotive-susan

$ ck crowdtune program:cbench-automotive-susan

$ ck replay experiment
```

CK workflows describe dependencies on CK soft detection plugins and packages to automatically adapt to a given platform and environment

cKnowledge.io/ml-object-detection-coco-tf-dependencies

Customizable CK workflows can be also used to autotune the whole AI/ML/SW/HW stack!

First expose coarse grain high-level choices, features, system state and behavior characteristics via CK APIs

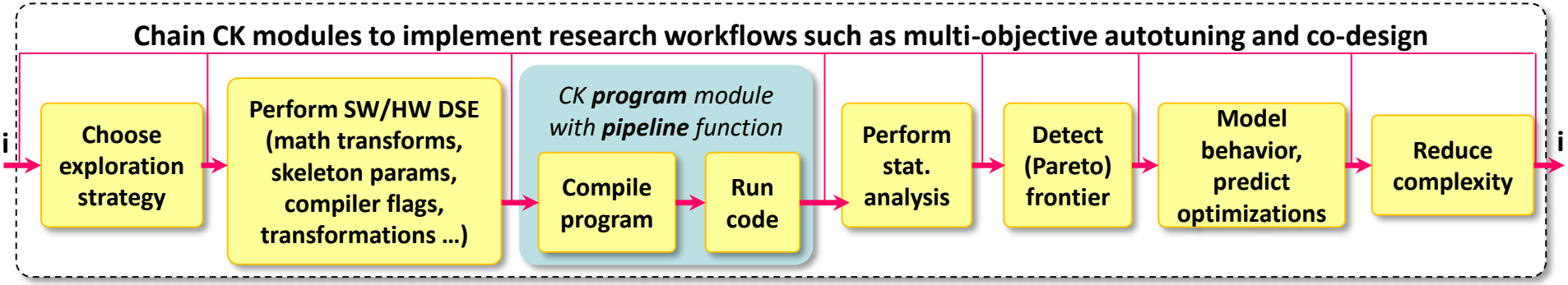


Formalized function B of a behavior of any CK object

$$\vec{b} = B(\vec{c}, \vec{f}, \vec{s})$$

Flattened CK JSON vectors
(dict converted to vector)
to simplify statistical analysis, machine learning and data mining

Then automate crowd-benchmarking and optimization across diverse models, datasets and platforms



Keep best species (AI/SW/HW choices); model behavior; predict better optimizations and designs

cKnowledge.io/reproduced-results

We can gradually expose more optimization parameters and characteristics via JSON files

Autotuning and machine learning specification:

```
{
  "characteristics":{
    "execution times": ["10.3","10.1","13.3"],
    "code size": "131938", ...},
  "choices":{
    "os":"linux", "os version":"2.6.32-5-amd64",
    "compiler":"gcc", "compiler version":"4.6.3",
    "compiler_flags":"-O3 -fno-if-conversion",
    "platform":{"processor":"intel xeon e5520",
          "l2":"8192“, ...}, ...},
  "features":{
    "semantic features": {"number_of_bb": "24", ...},
    "hardware counters": {"cpi": "1.4" ...}, ... }
  "state":{
    "frequency":"2.27", ...}
}
```

CK flattened JSON key

##characteristics#execution_times@1

```
"flattened_json_key":{
    "type": "text"|"integer" | “float" | "dict" | "list” | "uid",
    "characteristic": "yes" | "no",
    "feature": "yes" | "no",
    "state": "yes" | "no",
    "has_choice": "yes“ | "no",
    "choices": [ list of strings if categorical choice],
    "explore_start": "start number if numerical range",
    "explore_stop": "stop number if numerical range",
    "explore_step": "step if numerical range",
    "can_be_omitted" : "yes" | "no"
    ...
}
```

We organized the 1st reproducible ML/system optimization tournament in 2018

AI hardware
- All major vendors (Google, NVIDIA, ARM, Intel, IBM, Qualcomm, Apple, AMD …)

AI models
Many groups in academia & industry (Google, OpenAI, Microsoft, Facebook …)

AI software
- AI frameworks (TensorFlow, MXNet, PyTorch, CNTK, Theano)
- AI libraries (cuDNN, libDNN, ArmCL, OpenBLAS)

AI integration/services
- Cloud services (AWS, Google, Azure ...)

cKnowledge.org/request

Finding the most efficient AI/SW/HW stacks across diverse models, data sets and platforms via open competitions, share them as reusable CK components and visualize on a public scoreboard

Organizers (A-Z)

Luis Ceze, University of Washington
Natalie Enright Jerger, University of Toronto
Babak Falsafi, EPFL
Grigori Fursin, cTuning foundation
Anton Lokhmotov, dividiti
Thierry Moreau, University of Washington
Adrian Sampson, Cornell University
Phillip Stanley Marbell, University of Cambridge

Real use-cases

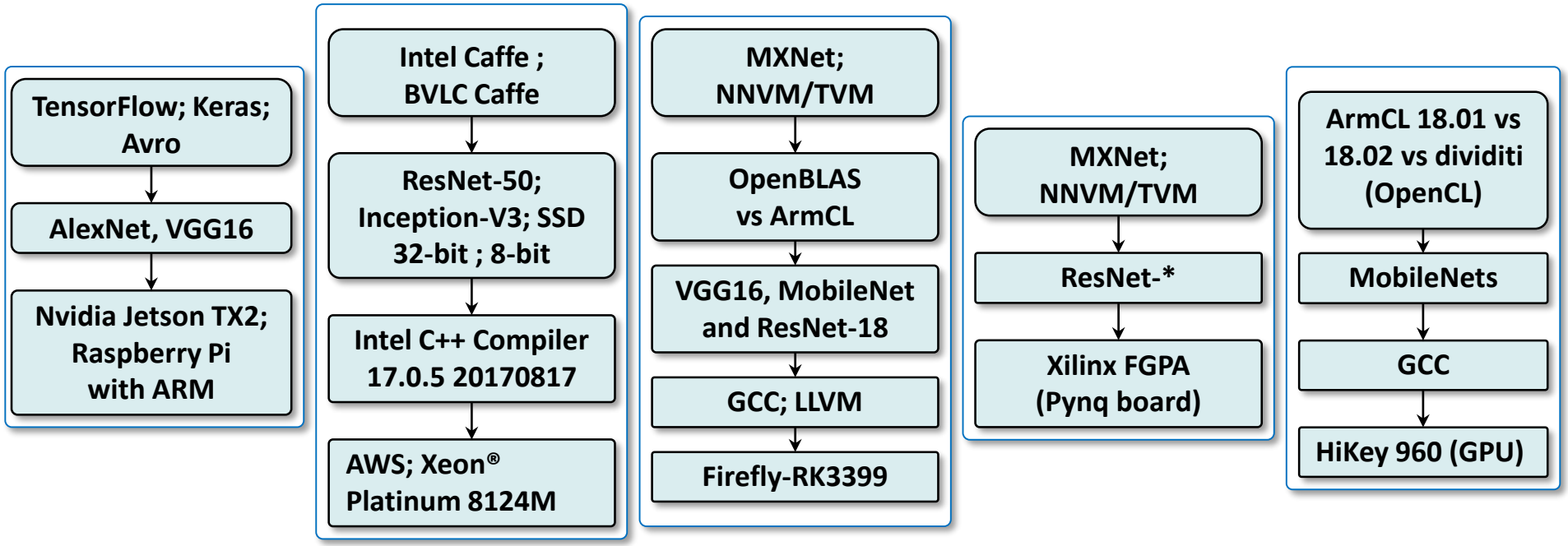
Healthcare
Agriculture
Finances
Automotive
Aerospace
Meteorology
Retail
Robotics
…

CK platform & CodeReef

Interdisciplinary community

We reproduced results from published ML papers and shared portable ML workflows!

8 intentions to submit and 5 submitted image classification workflows with unified Artifact Appendices



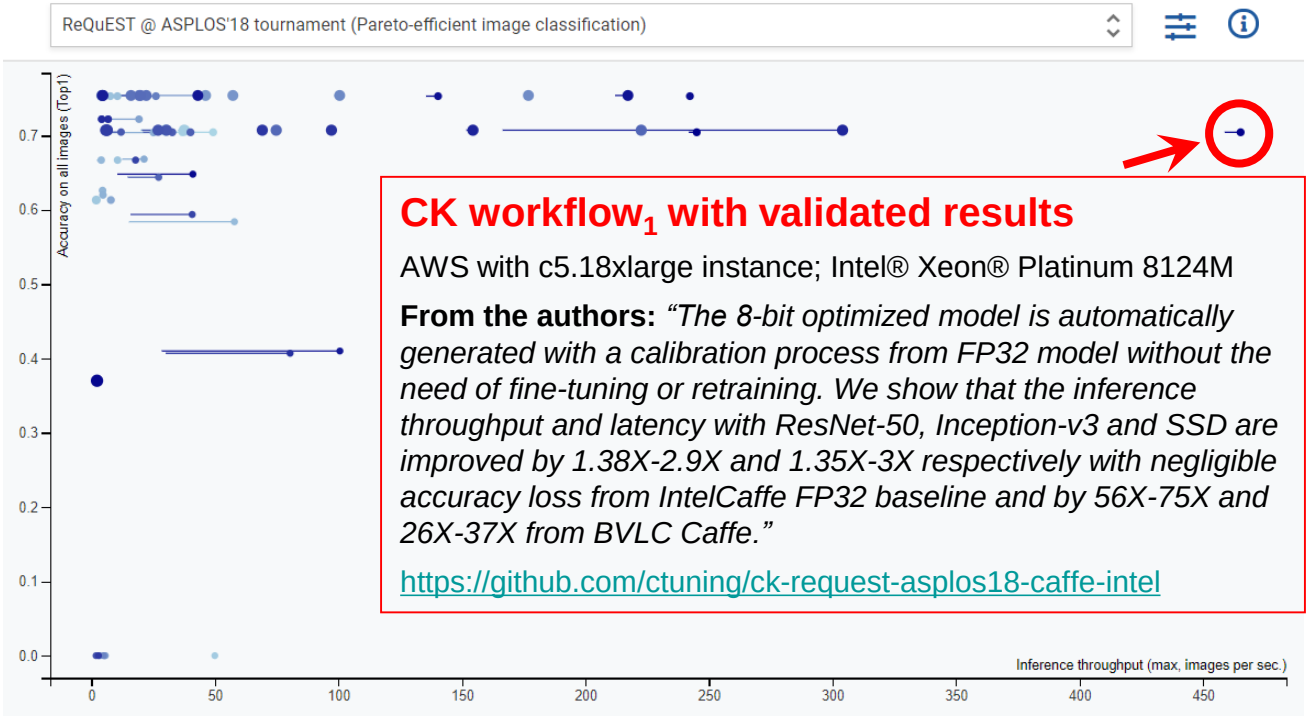
Public validation at github.com/ctuning/ck-request-asplos18-results via GitHub issues.

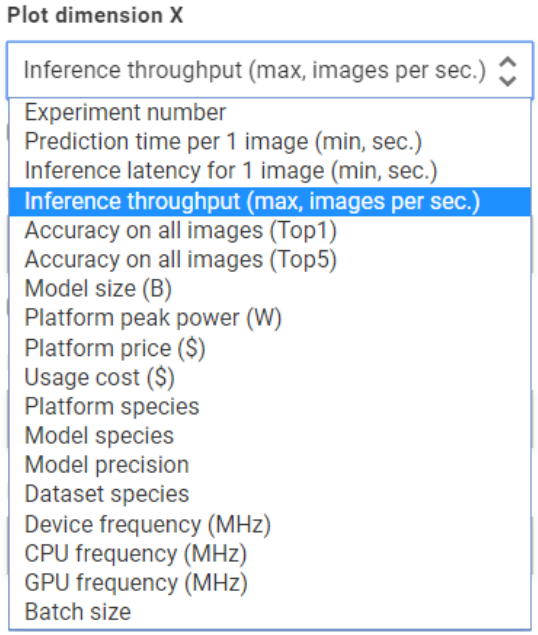
All validated papers are published in the ACM DL
with **portable, customizable and reusable CK components and workflows**:
dl.acm.org/citation.cfm?doid=3229762

See ACM ReQuEST report: portalparts.acm.org/3230000/3229762/fm/frontmatter.pdf

All results are available at online scoreboards

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/reproduced-results

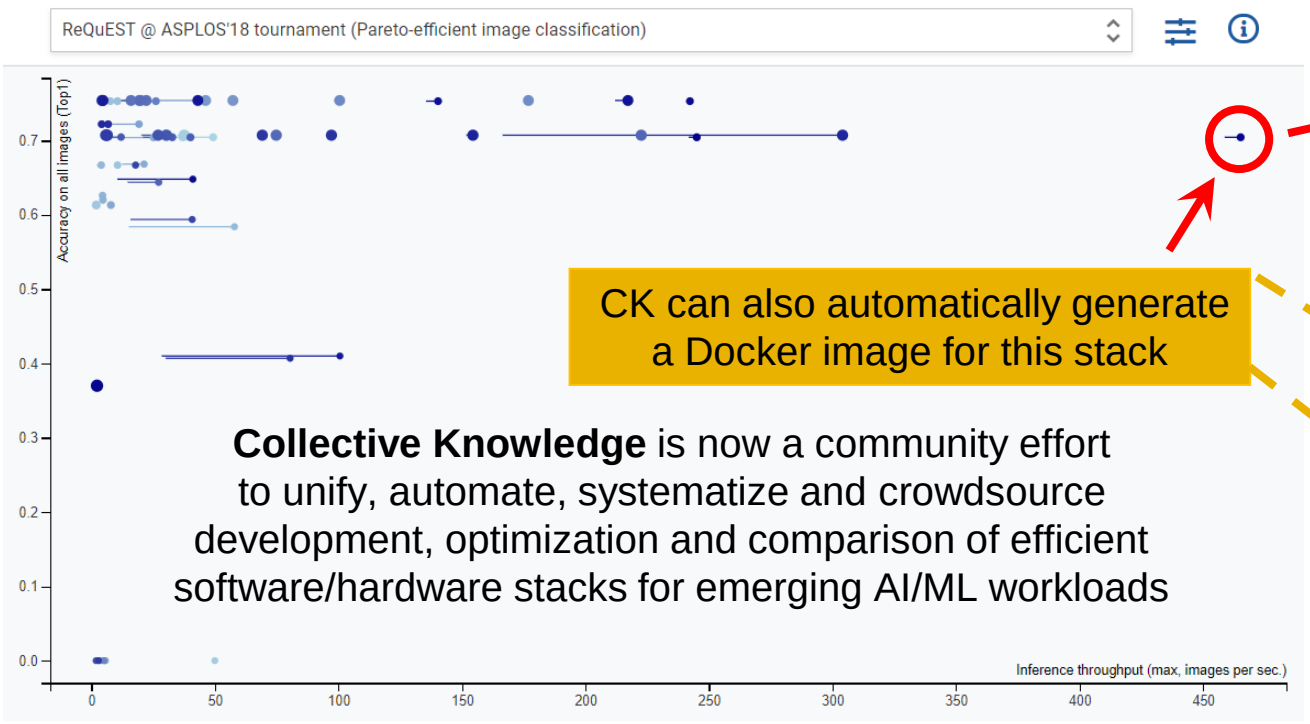




We are not announcing a single winner! We show all multi-dimensional results at cKnowledge.io/result/pareto-efficient-ai-co-design-tournament-request-acm-asplos-2018 and let the users select best ML/SW/HW stacks depending on multiple constraints for their production use!

Other companies managed to reproduce these results and started using CK

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/reproduced-results



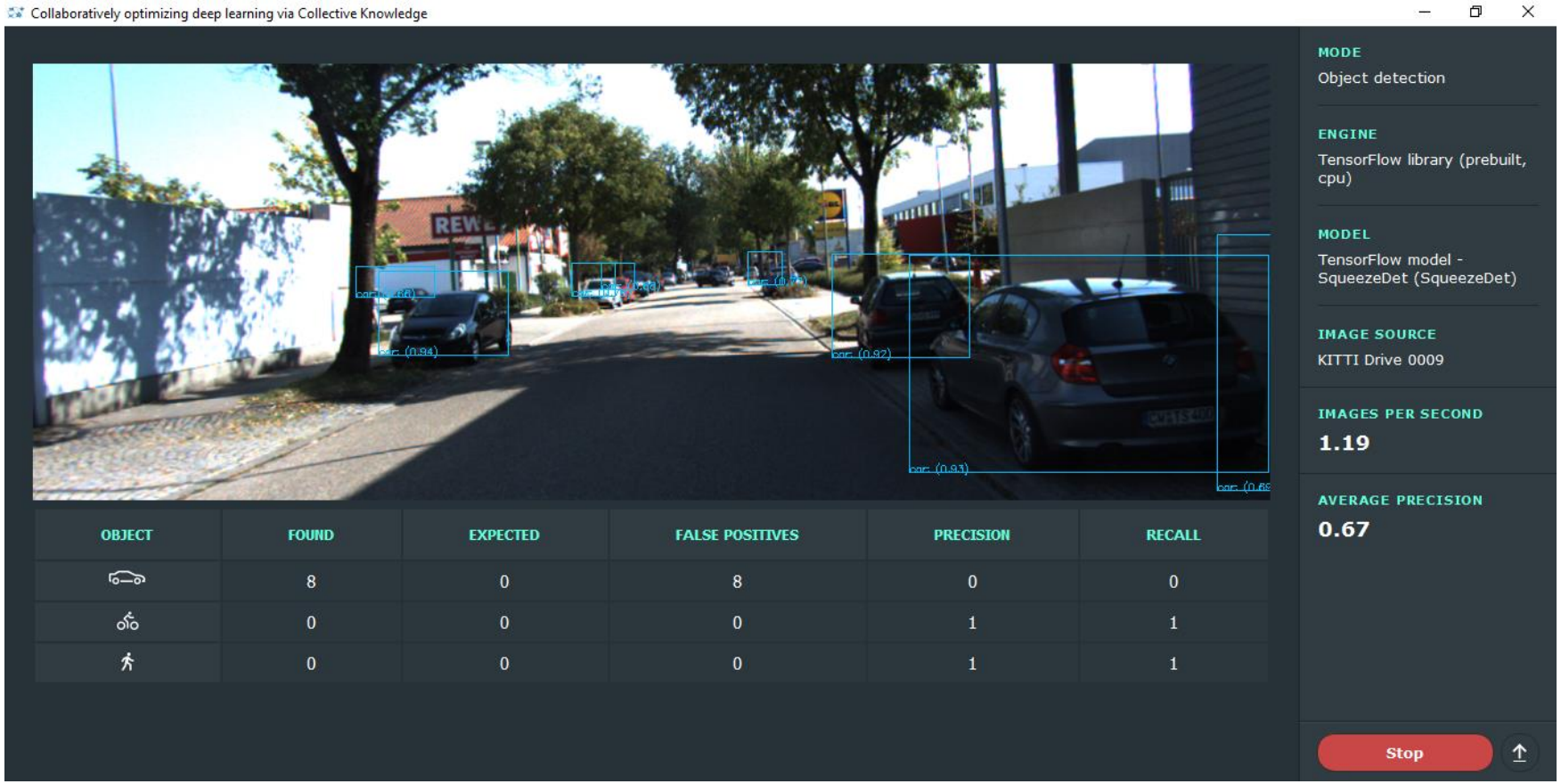
CK assists AWS market place with collaboratively optimized AI/ML stacks

amazon web services

Accelerate technology transfer: companies can validate published techniques in their production environment using shared CK workflows!

We made a joint presentation with Amazon at O'Reilly AI conference (October 2018)

General Motors uses CK to select the most efficient SW/HW stacks for ML



OBJECT	FOUND	EXPECTED	FALSE POSITIVES	PRECISION	RECALL
car	8	0	8	0	0
cyclist	0	0	0	1	1
pedestrian	0	0	0	1	1

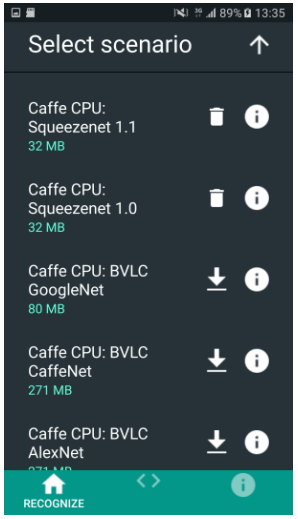
Performance, accuracy, power consumption practically never match official reports!

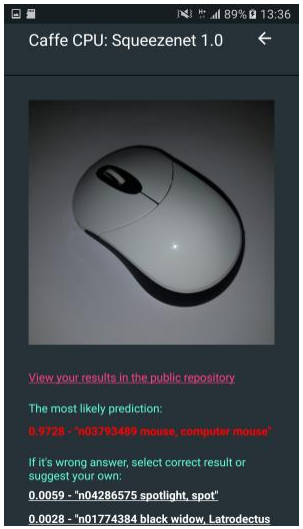
CK workflows and automation helped GM evaluate numerous models, datasets, frameworks and libraries to find the most efficient SW/HW stacks for object detection across Nvidia, AMD, ARM and Intel platforms (CUDA, OpenCL, OpenMP …)

Live presentation about how GM and partners use CK: www.youtube.com/watch?v=1ldgVZ64hEI

CK workflows were used to crowdsource AI/ML benchmarking across Android devices

Continuously collect statistics, bugs and misclassifications

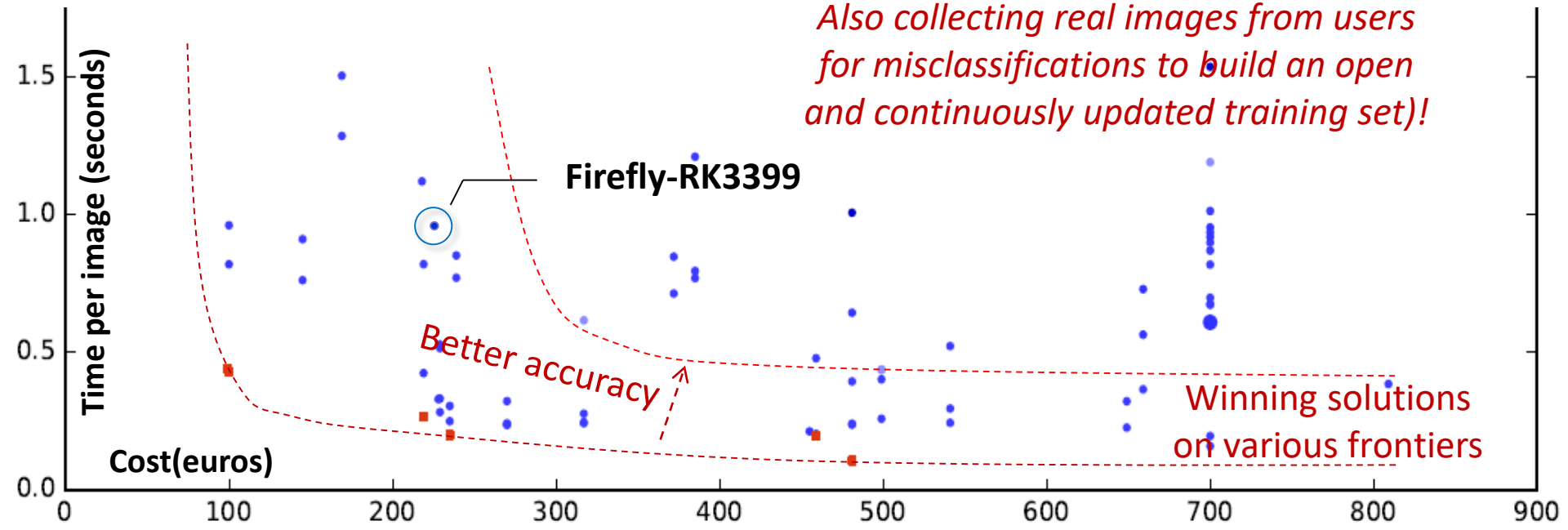




The number of distinct participated platforms:**800+**
The number of distinct CPUs: **260+**
The number of distinct GPUs: **110+**
The number of distinct OS: **280+**
Power range: **1-10W**

No need for a dedicated and expensive cloud – volunteers help us validate research ideas similar to SETI@HOME

Also collecting real images from users for misclassifications to build an open and continuously updated training set)!



If performance is bad (Caffe on Firefly-RK3399) we can continue crowd-tuning it

Tunable parameters of OpenCL-based BLAS (**github.com/CNugteren/CLBlast**)
For now only two data sets (small & large)

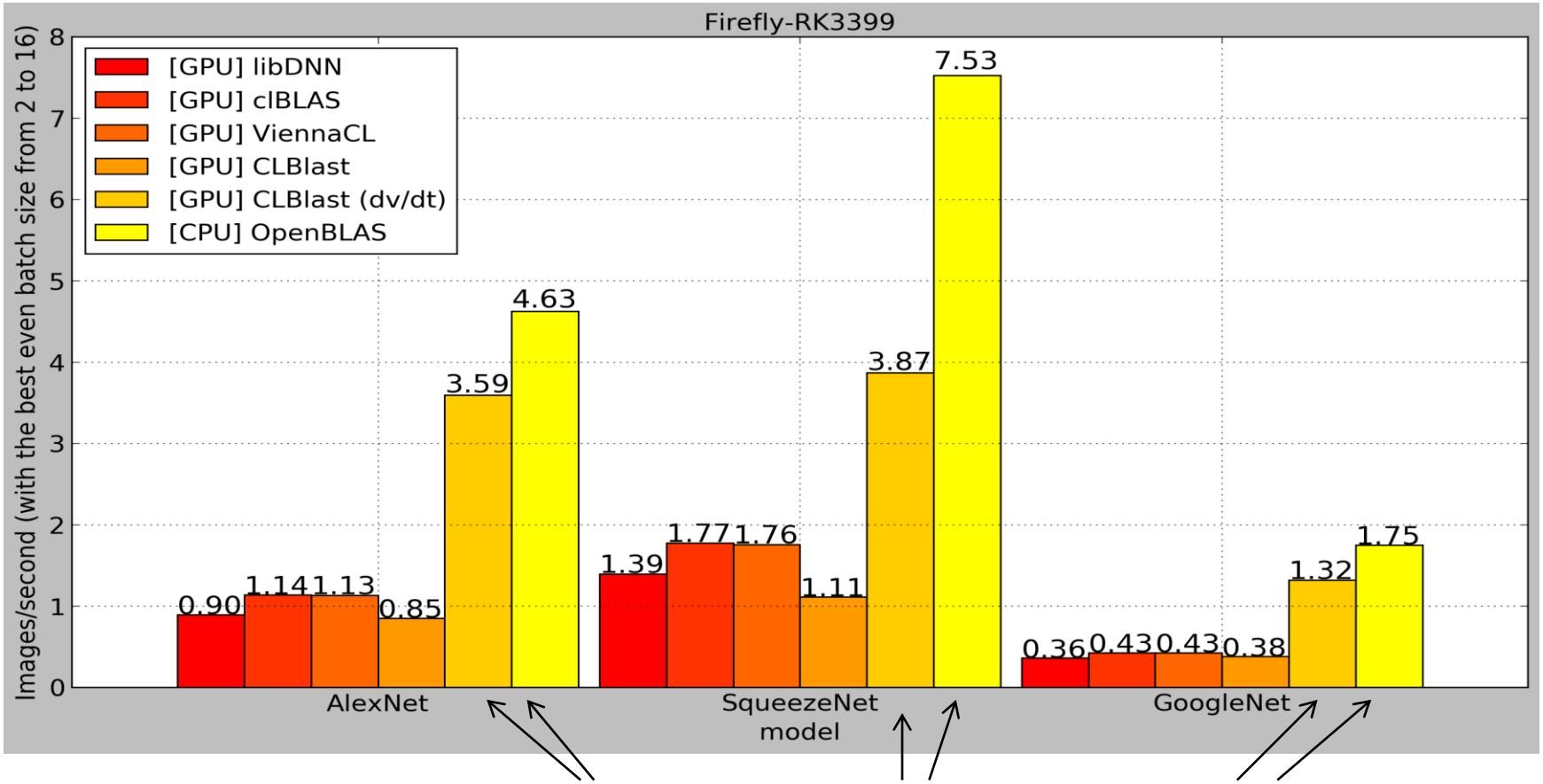
Name	Description	Ranges
KWG	2D tiling at workgroup level	{32,64}
KWI	KWG kernel-loop can be unrolled by a factor KWI	{1}
MDIMA	Local Memory Re-shape	{4,8}
MDIMC	Local Memory Re-shape	{8, 16, 32}
MWG	2D tiling at workgroup level	{32, 64, 128}
NDIMB	Local Memory Re-shape	{8, 16, 32}
NDIMC	Local Memory Re-shape	{8, 16, 32}
NWG	2D tiling at workgroup level	{16, 32}
SA	manual caching using the local memory	{0, 1}
SB	manual caching using the local memory	{0, 1}
STRM	Striding within single thread for matrix A and C	{0,1}
STRN	Striding within single thread for matrix B	{0,1}
VWM	Vector width for loading A and C	{8,16}
VWN	Vector width for loading B	{0,1}

Some extra constraints to avoid illegal combinations

Use different autotuners and ML to speed up design space exploration based on probabilistic focused search, generic algorithms, deep learning, SVM, KNN, MARS, decision trees …

CK also helps to automatically benchmark combinations of different libraries and compilers

Universal software and hardware benchmarking



- Caffe with autotuned OpenBLAS (threads and batches) is the fastest
- Caffe with autotuned CLBlast is 6..7x faster than default version and competitive with OpenBLAS-based version– now worth making adaptive selection at run-time.

CK helped to automate MLPerf.org inference submissions

"**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.

cKnowledge.io/?q="reproduced-results"+AND+MLPerf

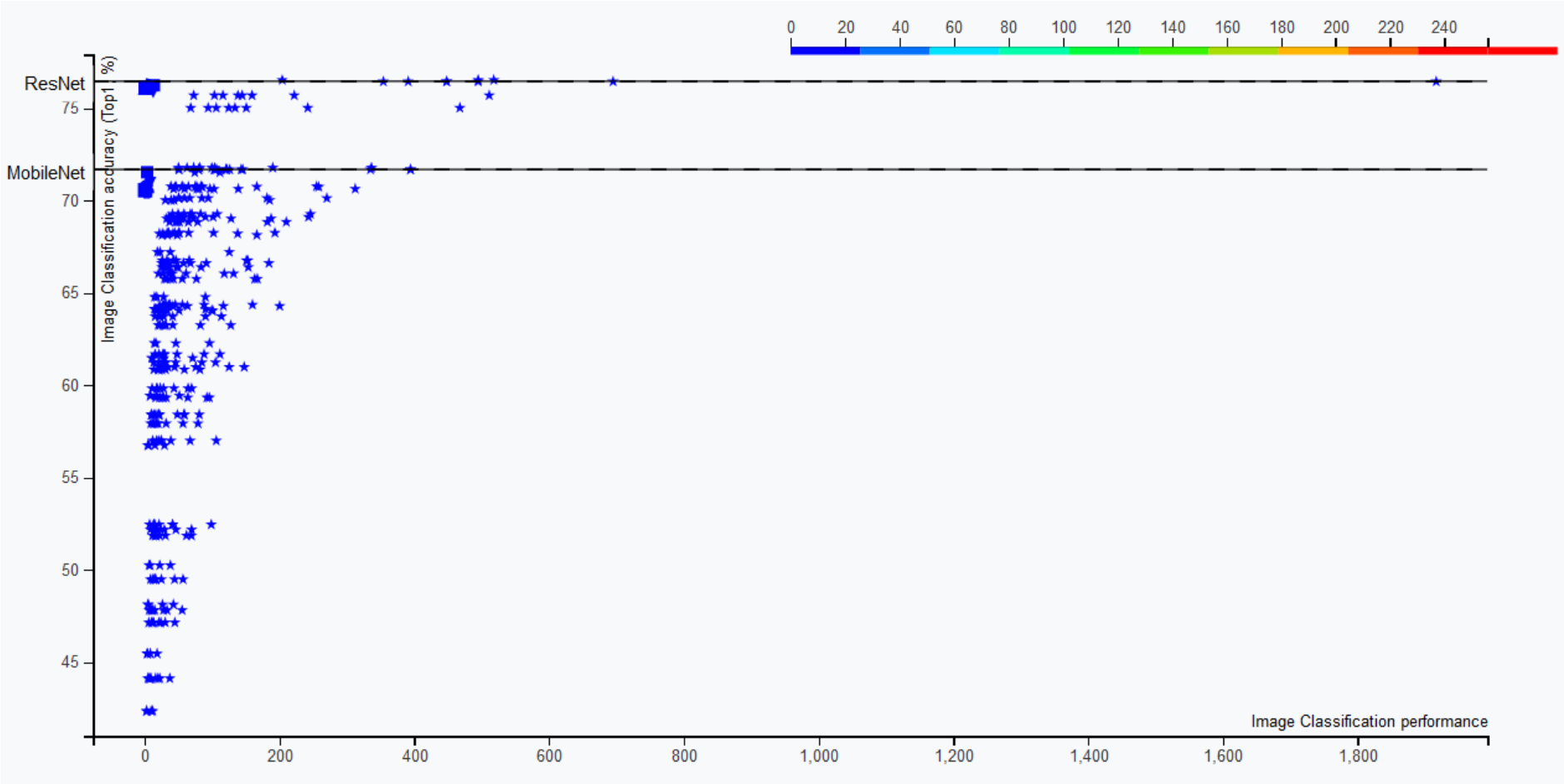
A broad ML benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms.

It is now possible to test how object detection from MLPerf works live:
cKnowledge.io/solution/demo-obj-detection-coco-tf-cpu-webcam-linux-azure

CK was used to autotune MobileNets across diverse devices for MLPerf submissions

The MLPerf consortium has released over 500 inference benchmarking v0.5 results from 14 organizations (including DellEMC, Nvidia, Google, Intel, Alibaba, Habana) measuring how fast and how well a pre-trained computer system can classify images, detect objects, and translate sentences.

Over 400 of these results were automated with the CK framework.



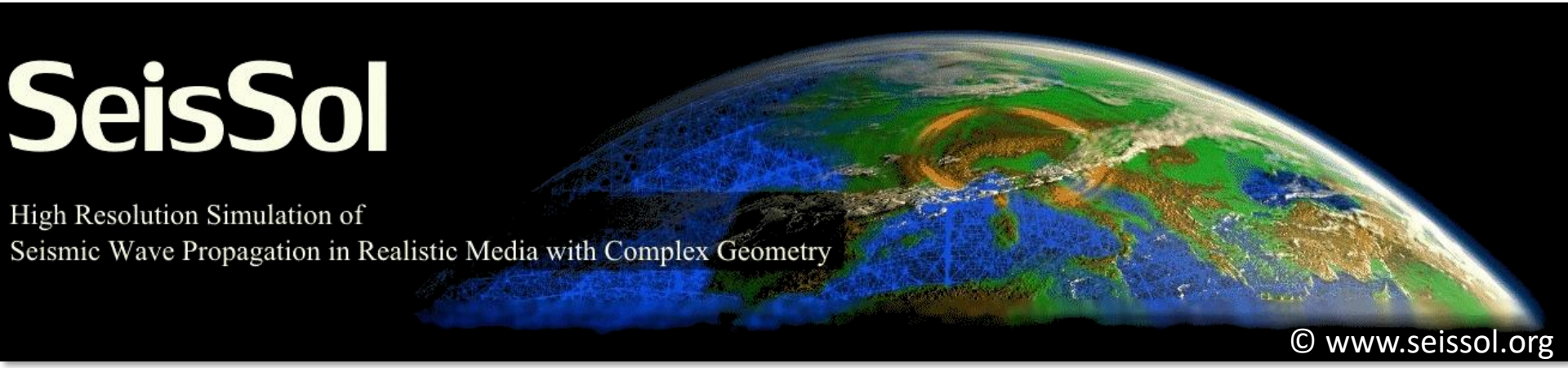
cKnowledge.io/reproduced-results mlperf.org

CK helps to automate, customize and reproduce HPC workloads

We collaborate with the Student Cluster Competition at ACM/IEEE Supercomputing to automate installation, execution and customization of HPC applications: across different platforms, environments and datasets:

github.com/ctuning/ck-scc18

github.com/reproindex/ck-scc

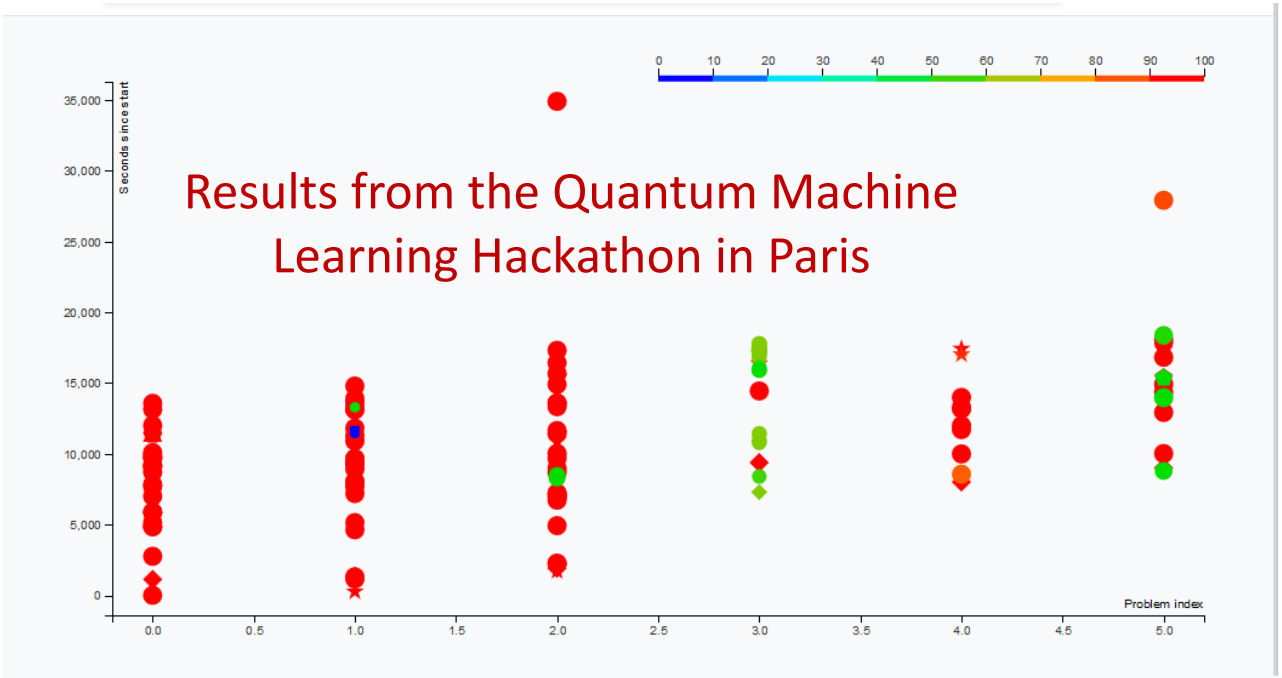


- Support automatic detection of already installed tools and data sets
- Can install missing dependencies via Spack and EasyBuild
- Can deploy application on different supercomputers with different job managers
- Can automatically validate the correctness of results (output, performance)

CK is used to collaboratively advance quantum computing

cKnowledge.org/quantum - Quantum Collective Knowledge workflows (QCK) to support reproducible hackathons, and help researchers share, compare and optimize different algorithms across conventional and quantum platforms

cKnowledge.io/reproduced-results



#	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

IBM

rigetti

RIVERLANE

ThoughtWorks®

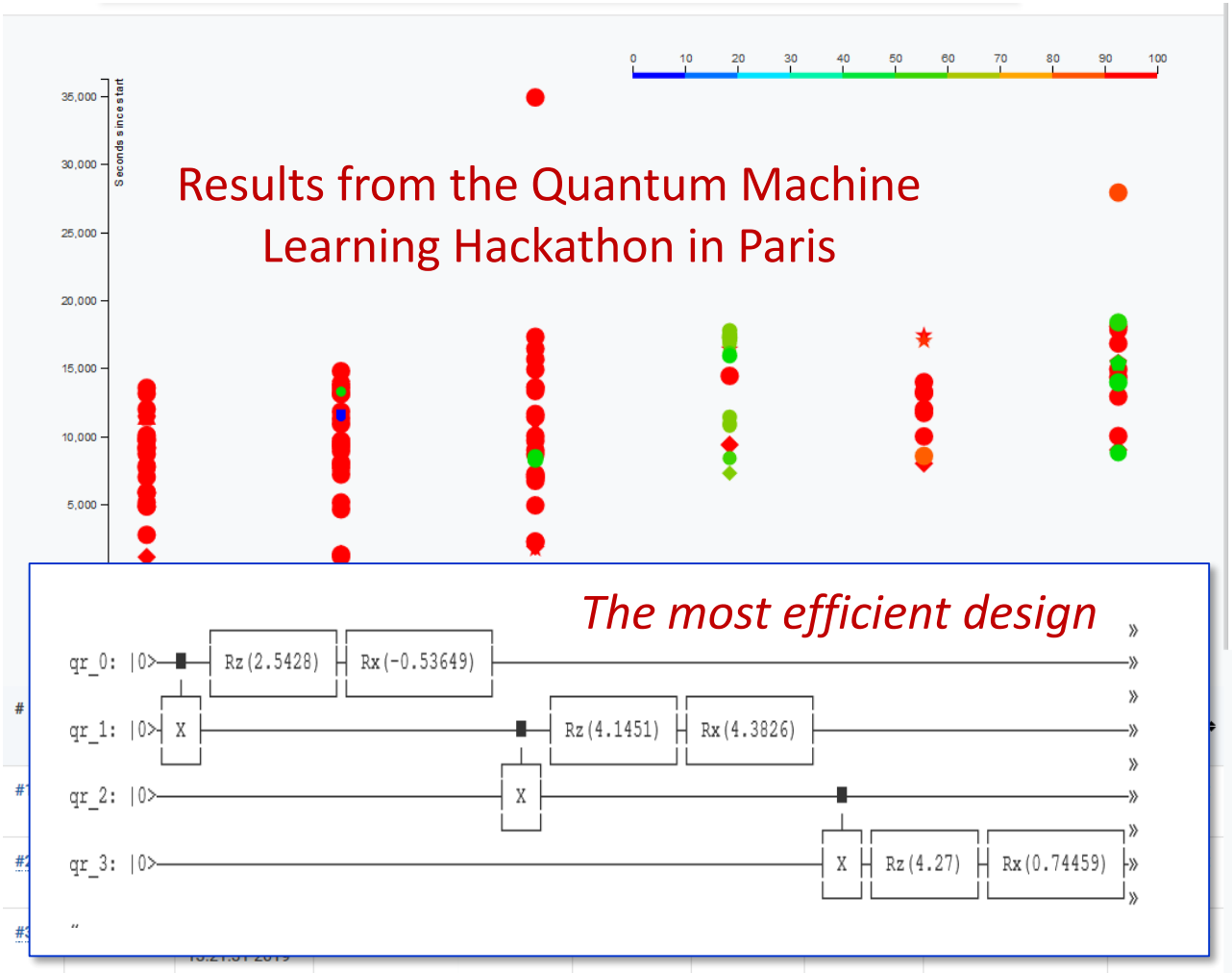
QUANTO NATION

Innovate UK

CK is used to collaboratively advance quantum computing

cKnowledge.org/quantum - Quantum Collective Knowledge workflows (QCK) to support reproducible hackathons, and help researchers share, compare and optimize different algorithms across conventional and quantum platforms

cKnowledge.io/reproduced-results



IBM

rigetti

RIVERLANE

ThoughtWorks®

QUANTO NATION

Innovate UK

ACM evaluates CK to package research workflows and results along with published papers

ACM Pilot Demo - Collective Knowledge: Packaging and Sharing

Presentation about ACM pilot to improve research reproducibility and reusability:
youtu.be/DIkZxraTmGM

cKnowledge.io portal to close the gap between research and practice

Our goal is to enable “live” papers, i.e. to share CK workflows and components along with research papers to make it easier for the community to reproduce the results and try the algorithms with the latest/different components: cKnowledge.io/solutions

CK JSON API

- Tasks
  - Object detection
  - Object classification
  - Speech recognition
  - ...
  - Training/inference
  - ...

CK JSON API

- Models
  - MobileNets
  - ResNet
  - VGG
  - ...
  - SqueezeDet
  - ...

CK JSON API

- Software
  - TensorFlow
  - Caffe
  - MXNet
  - ...
  - PyTorch/Caffe2
  - ...

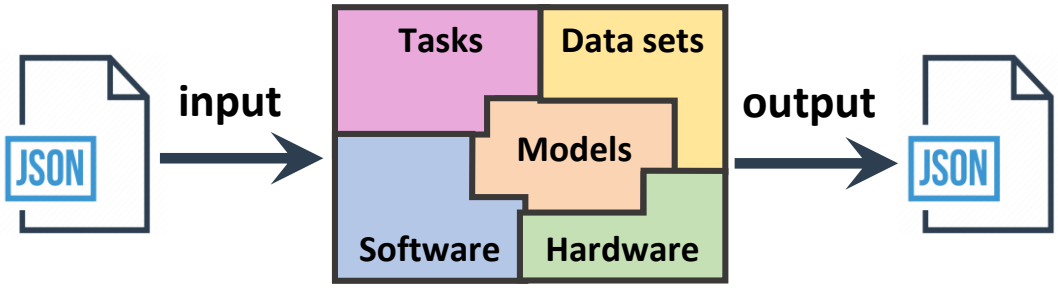
CK JSON API

- Data sets
  - ImageNet
  - KITTI
  - VOC
  - ...
  - Real data sets
  - ...

CK JSON API

- Hardware
  - CPU
  - GPU
  - TPU
  - ...
  - NN accelerators
  - ...

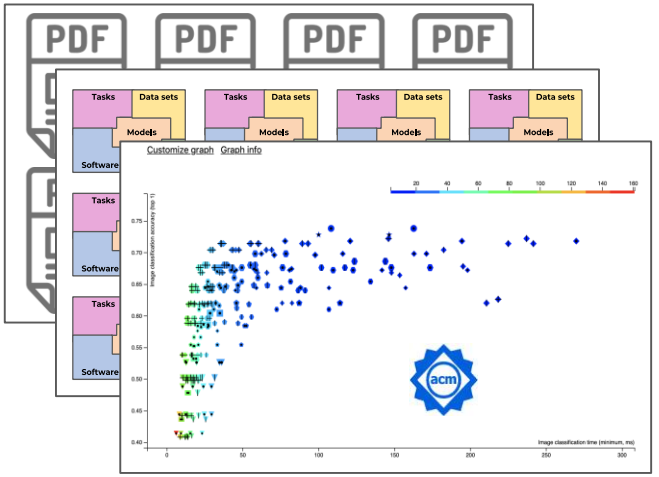
CK solutions with portable workflows and unified API



CK client (cBench)

API → Build → Run → Validate

Customizable dashboards for crowd-benchmarking



The current state of the Collective Knowledge technology

- The CK platform is a complete functional prototype: cKnowledge.io

- CK is used in production by companies and universities
  but there is still a lot to be improved to make it a user-friendly product!

- Current major issues preventing further adoption:
  - CLI and JSON meta is not user friendly (similar to Git)
  - Distributed nature of CK makes it difficult to understand who is using CK and ensure the stability/testing of workflows
  - Lack of an open portal to exchange stable components (similar to PyPi)
  - Lack of automatic testing of all components and workflows

- Currently supported by my non-profit cTuning foundation (cTuning.org)
  but our resources are very limited - we want to attract more organizations and companies to improve our open-source technology together

Don’t hesitate to get in touch if you are interested to help with this effort:
gfursin@cKnowledge.io