CK: an open-source framework to automate, reproduce, crowdsource and reuse experiments at HPC conferences

github.com/ctuning/ck  cKnowledge.org/partners

Companies, universities, and non-profits do cool things with the help of CK

CK complements many popular tools and services

Grigori Fursin  @grigori_fursin  fursin.net/research
AI, ML, systems and quantum research is booming – 1000+ papers every year …

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

**Platforms**
- HPC
- Desktops
- IoT
- Mobile
- Cloud services
Many great tools, data sets and models to help researchers …

<table>
<thead>
<tr>
<th>Applications</th>
<th>Scientific tools</th>
<th>Build tools</th>
<th>Languages</th>
<th>Compilers</th>
<th>DevOps tools</th>
<th>Package managers</th>
<th>Libraries</th>
<th>OS</th>
<th>Shells</th>
<th>Hardwares</th>
<th>Workload managers</th>
<th>Databases / experiments</th>
<th>Knowledge sharing</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meteorology</td>
<td>MATLAB</td>
<td>Make</td>
<td>C++</td>
<td>LLVM</td>
<td>Git</td>
<td>Anaconda</td>
<td>SciPy</td>
<td>Linux</td>
<td>bash</td>
<td>CPU</td>
<td>MPI</td>
<td>MySQL</td>
<td>ArXiv</td>
<td>HPC</td>
</tr>
<tr>
<td>Health</td>
<td>Scilab</td>
<td>Cmake</td>
<td>C#</td>
<td>GCC</td>
<td>Jenkins</td>
<td>TensorFlow</td>
<td>TFLite</td>
<td>MacOS</td>
<td>sh</td>
<td>GPU</td>
<td>SSH</td>
<td>PostgreSQL</td>
<td>ACM DL</td>
<td>Deskto</td>
</tr>
<tr>
<td>Robotics</td>
<td>Simulink</td>
<td>SCons</td>
<td>C</td>
<td>Intel</td>
<td>Docker</td>
<td>OpenBLAS</td>
<td>EEMBC</td>
<td>BSD</td>
<td>csh</td>
<td>TPU / NN</td>
<td>PBS/GMT</td>
<td>MongoDB</td>
<td>IEEE DL</td>
<td>IoT</td>
</tr>
<tr>
<td>Automotive</td>
<td>LabVIEW</td>
<td>Go</td>
<td>PHP</td>
<td>PGI</td>
<td>Kubernetes</td>
<td>MAGMA</td>
<td>CUDA</td>
<td>Windows</td>
<td>ksh</td>
<td>DSP</td>
<td>Travis</td>
<td>Text files</td>
<td>GitHub</td>
<td>Mobile</td>
</tr>
<tr>
<td>Economics</td>
<td>Gnuplot</td>
<td>Fortran</td>
<td>TVM</td>
<td>TVM</td>
<td>Singularity</td>
<td>cuDNN</td>
<td>cuFFT</td>
<td>Android</td>
<td>Windows shell</td>
<td>FPGA</td>
<td>JupyterHub</td>
<td>Cloud services</td>
<td>Zenodo</td>
<td>Cloud services</td>
</tr>
<tr>
<td>Physics</td>
<td>LaTeX</td>
<td>Java</td>
<td>CUDA</td>
<td>CUDA</td>
<td></td>
<td>ArmNN</td>
<td>Boost</td>
<td>Quantum</td>
<td></td>
<td></td>
<td>CodeLabs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Astronomy</td>
<td>Ipython</td>
<td>Python</td>
<td></td>
<td></td>
<td></td>
<td>magnum</td>
<td>HDF5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>singularity</td>
<td>MPI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>tf</td>
<td>Protobuf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

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

**Web services**
- GitHub
- GitLab
- BitBucket
- Travis
- JupyterHub
- CodeLabs
- SageMaker

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

**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
- Npm
- Pip
- Sbt
dpkg
- Spack
- EasyBuild

**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

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

**Workload managers**
- MPI
- SLURM
- PBS
- FLUX

**Databases**
- MySQL
- PostgreSQL
- MongoDB
- CouchDB
- Text files
- JSON files
- XLS files

**Platforms**
- HPC
- Desktops
- IoT
- Mobile
- Cloud services
Let’s innovate …

<table>
<thead>
<tr>
<th>Applications</th>
<th>Scientific tools</th>
<th>Build tools</th>
<th>Languages</th>
<th>Compilers</th>
<th>DevOps tools</th>
<th>Package managers</th>
<th>Libraries</th>
<th>OS</th>
<th>Shells</th>
<th>Hardward</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meteorology</td>
<td>MATLAB</td>
<td>Make</td>
<td>C++</td>
<td>LLVM</td>
<td>Git</td>
<td>Anaconda</td>
<td>SciPy</td>
<td>Linux</td>
<td>bash</td>
<td>CPU</td>
<td>HPC</td>
</tr>
<tr>
<td>Health</td>
<td>Scilab</td>
<td>Cmake</td>
<td>C#</td>
<td>GCC</td>
<td>Jenkins</td>
<td>TF-Lite</td>
<td>MacOS</td>
<td>MacOS</td>
<td>sh</td>
<td>GPU</td>
<td>Desktops</td>
</tr>
<tr>
<td>Robotics</td>
<td>Simulink</td>
<td>SCons</td>
<td>C</td>
<td>Intel</td>
<td>Docker</td>
<td>OpenBLAS</td>
<td>BSD</td>
<td>BSD</td>
<td>csh</td>
<td>TPU/NN</td>
<td>Mobile</td>
</tr>
<tr>
<td>Automotive</td>
<td>LabVIEW</td>
<td></td>
<td>Go</td>
<td>PGI</td>
<td></td>
<td>MAGMA</td>
<td>Windows</td>
<td>Windows</td>
<td>ksh</td>
<td>DSP</td>
<td>Cloud</td>
</tr>
<tr>
<td>Economics</td>
<td>GnuPlot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cuDNN</td>
<td>Android</td>
<td>Android</td>
<td>Windows shell</td>
<td>FPGA</td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td>LaTeX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cuFFT</td>
<td></td>
<td></td>
<td></td>
<td>Quantum</td>
<td></td>
</tr>
<tr>
<td>Astronomy</td>
<td>Ipython</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ArmNN</td>
<td></td>
<td></td>
<td></td>
<td>Simulators</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLBoost</td>
<td></td>
<td></td>
<td></td>
<td>Interconnects</td>
<td></td>
</tr>
</tbody>
</table>

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

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

Build tools
- Make
- Cmake
- SCons

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

Models
- GoogleNet
- AlexNet
- VGG
- ResNet
- MobileNets
- SSD
- SqueezeNet
- DeepSpeech
- MiDataSets
- Human Cell Atlas
- 1000 Genomes
- Earth models
- OpenStreetMap
- BitBucket
- Travis
- JupyterHub
- Codelabs
- SageMaker
- MPI
- SLURM
- PBS
- FLUX
- 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

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

Platforms
- HPC
- Desktops
- IoT
- Mobile
- Cloud services

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

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

OS
- Linux
- MacOS
- BSD
- Windows
- Android
Reproducing results at HPC conferences (artifact evaluation): the good, the bad, and the ugly

Creating ad-hoc project with many add-hoc scripts and README files, and sharing it via GitHub, GitLab, ... doesn’t make it easily portable, reusable and customizable!

Algorithm

Program

Compiler

Binary and libraries

State of the system

Data set

Run-time environment

Architecture

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

image corner detection

matmul OpenCL

compression

neural network CUDA

GCC V8.1

LLVM V7.0

Intel Compilers 2017

image-jpeg-0001

bzip2-0006

txt-0012

video-raw-1280x1024

cvs speedups

txt hardware counters

xls table with graph

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

Have some (common) meta: compilation, linking and optimization flags

Have some common meta: features, characteristics, optimizations

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

Very often software from published papers die when students leave or projects finish!
Creating ad-hoc project with many
add-hoc scripts and README files,
and sharing it via GitHub, GitLab
doesn’t make it easily portable,
reusable and customizable!

Very often software from published papers die when students leave or projects finish!
Collective Knowledge concept (CK): add, share and reuse common APIs!

Provide unified Python APIs and JSON meta descriptions for similar code and data objects – enables DevOps!

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
- GCC V8.1
- LLVM V7.0
- Intel Compilers 2017
- image-jpeg-0001
- bzip2-0006
- txt-0012
- video-raw-1280x1024
- cvs speedups
- txt hardware counters
- xls table with graph

**meta.json**
CK framework: just CLI to create and access APIs (very portable - minimal dependencies)

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

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

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

meta.json

JSON input → JSON output

image corner detection
matmul OpenCL
compression
neural network CUDA

GCC V8.1
LLVM V7.0
Intel Compilers 2017

bzip2-0006
txt-0012
video-raw-1280x1024

cvs speedups
txt hardware counters
xls table with graph

meta.json

JSON input → JSON output

meta.json

JSON input → JSON output

JSON input → JSON output

meta.json

JSON input → JSON output

JSON input → JSON output

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

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

cKnowledge.org/shared-modules.html
CK framework: the community can implement complex workflows via CK APIs

- **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
- image-jpeg-0001
- bzip2-0006
- txt-0012
- video-raw-1280x1024
- cvs speedups
- txt hardware counters
- xls table with graph

CK: small python module (~200Kb); no extra dependencies; Linux; Win; MacOS

Assemble workflows from shared components

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

CK framework: the community can implement complex workflows via CK APIs

cKnowledge.org/shared-programs.html
CK framework: provide simple and unified directory structure for research projects

cKnowledge.org/shared-repos.html

/ 1st level directory – CK modules

- setup
- find
- extract features
- compile
- run
- autotune
- add
- replay

/ 2nd level dir - CK entries

- TensorFlow
- PyTorch
- ARM compute lib
- ImageNet
- Car video stream
- Real surveillance camera
- image classification
- object detection
- GEMM OpenCL
- convolution CPU
- performance results
- training / accuracy
- bugs

/ CK meta info

- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file
- JSON file

JSON API
Python module
holder for original artifact
We started collaboratively abstract all components with our partners since 2017 ...

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

**Scientific tools**
- MATLAB
- Scilab
- Simulink
- LabVIEW
- Gnuplot
- LaTeX
- Ipypthon

**Build tools**
- Make
- Cmake
- SCons
- Bazal
- 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
- TF
- OpenBLAS
- MAGMA
- cuDNN
- cuFFT
- ArmNN
- CLBlast
- Boost
- HDF5
- MPI
- OpenCV
- Protobuf

**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

**Tools**
- Cmake
- Build
- SLURM
- MPI
- TVM
- GCC
- dpkg
- Anaconda
- Jenkins
- JupyterHub
- Jenkins
- C
- C#
- Go
- Python

**Platforms**
- HPC
- Desktops
- IoT
- Mobile
- Cloud services

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

**Collective Knowledge**
- Simple Python APIs with JSON (dictionary) I/O
- Simple JSON meta-description of all components
- Simple access from command line, different languages and web
- Simple sharing of all components via GitHub, Zenodo, etc ...

**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

**OS**
- Linux
- MacOS
- BSD
- Windows
- Android
Started converting all my own past R&D on performance autotuning and machine learning to CK …

cKnowledge.org/shared-repos.html  cKnowledge.org/shared-modules.html

Artifact automated and reusable Collective Knowledge COMPATIBLE Workflow CK

1) Describing different operating systems (github.com/ctuning/ck-env)

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

2) Detecting and unifying information about platforms

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

3) Detecting installed “software” (both code and data):

   $ ck search soft --tags=dataset  cKnowledge.org/shared-soft-detection-plugins.html
   $ ck detect soft:compiler.llvm

   $ ck show env --tags=llvm

4) Installing missing packages (both code and data): front-end to EasyBuild, Spack, scons …

   $ ck search package --tags=model  cKnowledge.org/shared-packages.html
   $ ck install compiler:compiler-llvm-7.0.0-universal
   $ ck show env --tags=llvm
   $ ck virtual env --tags=llvm,v7.0.0

I want to work in a native environment and use containers to make stable snapshots
Enabling customizable and portable workflows by connecting CK components

Universal program workflow to compile, run, profile and autotune diverse benchmarks across diverse data sets and platforms, validate results, record experiments, share and reproduce them, and report discrepancies

cKnowledge.org/repo

$ 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

We can even automatically generate reproducible and interactive articles (collaboration with Raspberry Pi foundation): cKnowledge.org/rpi-crowd-tuning
Repositories of customizable, portable and reusable research components with CK API

cKnowledge.org/shared-repos.html

Crowdsource experiments with the help of volunteers across diverse models, data sets and platforms

AI frameworks
- TensorFlow
- Caffe
- Caffe2
- CNTK
- Torch
- MXNet

Models
- AlexNet
- GoogleNet
- VGG
- ResNet
- SqueezeDet
- SSD
- MobileNets

Data sets
- KITTI
- COCO
- VOC
- ImageNet

Real life objects from the community

Libraries
- cuDNN
- ArmCL
- OpenCL
- ViennaCL
- CLBlast
- cuBLAS
- TVM

OS
- Linux
- MacOS
- Windows
- Android

Hardware
- CPU
- DSP
- FPGA
- GPU
- NN accelerators
- Simulators

Customizable CK workflows for real-world user tasks

Assemble scenarios such as image classification as LEGO™

Share complete workflows along with published papers to automate artifact evaluation and help the community build upon prior work

Present best results, workflows and components on a live scoreboard for fair comparison and reuse

cKnowledge.org/repo

Help students learn multidisciplinary techniques, quickly prototype new ones, validate them in practice with companies, and even contribute back new research components

Help companies select the most appropriate workflows, save R&D costs, accelerate adoption of new techniques!
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
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!

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

We are not announcing a single winner! We show all multi-dimensional results at cKnowledge.org/dashboard/request.asplos18 and let users select best ML/SW/HW stacks depending on multiple constraints!
Amazon colleagues managed to reproduce 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!

Collective Knowledge is now a community effort to unify, automate, systematize and crowdsource development, optimization and comparison of efficient software/hardware stacks for emerging AI/ML workloads.

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

See Amazon presentation at O’Reilly AI conference:
conferences.oreilly.com/artificial-intelligence/ai-eu/public/schedule/detail/71549
CK helped General Motors to select the most efficient SW/HW stacks (performance, accuracy, power consumption practically never match official reports!)

CK allows to select the most efficient SW/HW stacks on a Pareto frontier (performance, accuracy, energy, memory usage, costs) for object detection, image classification and other tasks: www.youtube.com/watch?v=1ldgVZ64hEi
CK helps to automate Student Cluster competitions

[link: github.com/ctuning/ck-scc18/wiki] - proof-of-concept CK workflow to automate installation, execution and customization of SeisSol application from the SC18 SCC Reproducibility Challenge across different platforms, environments and datasets

- Support automatic detection of already installed tools and data sets
- Can install missing dependencies via EasyBuild and Spack
- 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.org/dashboard/hackathon.20190127

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>
</table>
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.org/dashboard/hackathon.20190127

Results from the Quantum Machine Learning Hackathon in Paris

The most efficient design

\[
\frac{d\vec{v}}{dt}
\]
cKnowledge.org: future plans to enable open science

Websites:
• [github.com/ctuning/ck](https://github.com/ctuning/ck)
• [http://cKnowledge.org](http://cKnowledge.org)
• [cKnowledge.org/shared-repos.html](http://cKnowledge.org/shared-repos.html)

Huge thanks to all partners and contributors:
[http://cKnowledge.org/partners](http://cKnowledge.org/partners)

From prototype to production quality (beginning of a long journey)
• Collaboratively standardize APIs and meta descriptions
• Improve installation and documentation; add GUIs
• Add more CK components and workflows for real-world tasks
• Create online index (everyone could describe their components)

Open to collaboration
• Joint R&D projects, hackathons, and tournaments (AI, ML, quantum)
• Automation and sharing of experiments
• Reproducible articles with reusable workflows

Contact: [Grigori.Fursin@cTuning.org](mailto:Grigori.Fursin@cTuning.org) or [grigori@dividiti.com](mailto:grigori@dividiti.com)