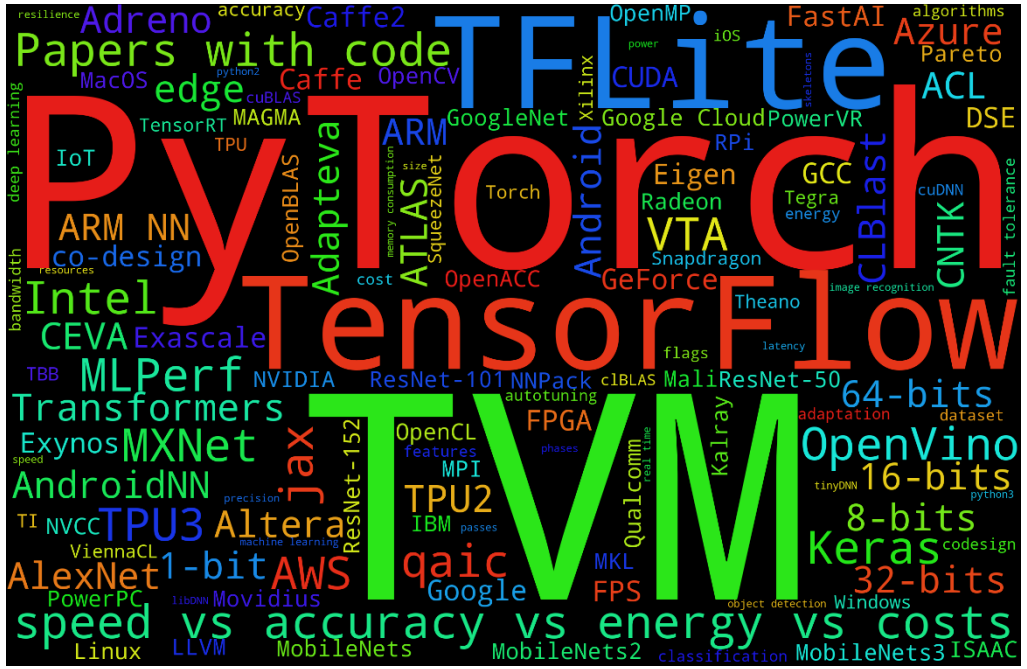
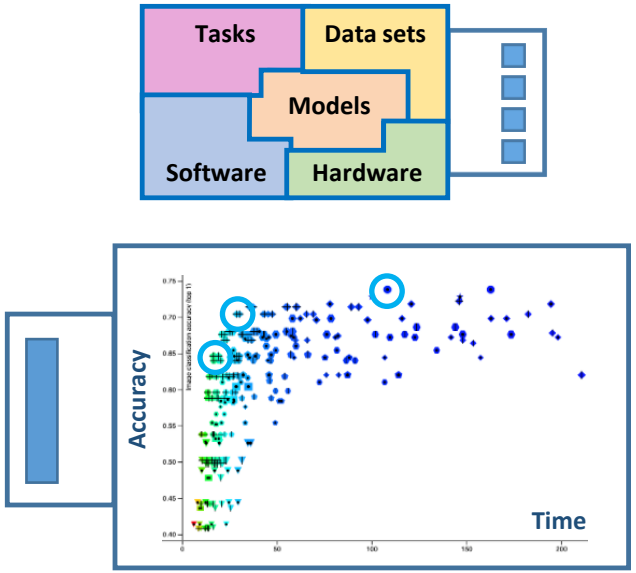
MLPerf Design Space Exploration and Production Deployment

or how our reproducibility initiatives at ML & Systems conferences, CK framework and MLPerf help to automate development and deployment of Pareto-efficient ML Systems



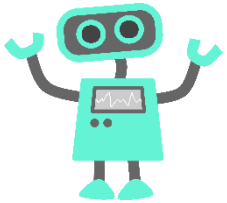


Grigori Fursin VP at OctoML.ai ; MLCommons member cKnowledge.io/@gfursin

Outline

- Personal motivation: how to make it easier to validate ML/AI-based research ideas in the real world?
- Artifact evaluation and reproducibility initiatives at ML and system conferences
- Learning from reproducing 150+ research papers at ASPLOS, CGO, PPoPP, PACT and MLSys and validating some of them in the real world
- Collective Knowledge concept (CK): bridging the growing gap between academic research and industry with reusable artifacts and automation recipes
- Using CK to automate design space exploration of ML Systems across diverse ML frameworks, models, data sets and platforms
- Automating the deployment of Pareto-efficient AI/ML systems in the real world
- Developing the new CK2 framework and discussing a new MLCommons WG on DSE – provide your feedback and join our community effort!

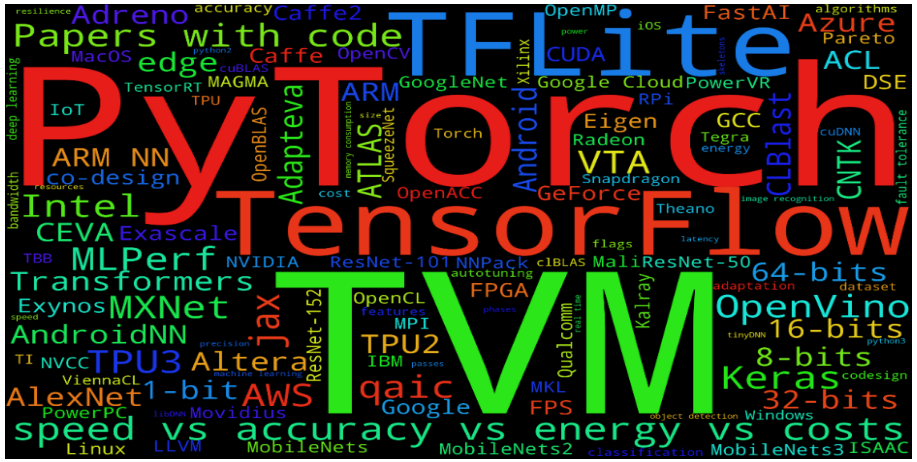
From ML/AI-based ideas to production

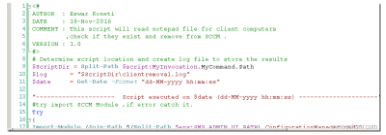


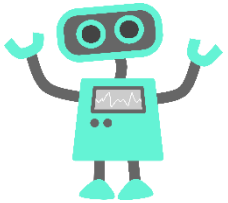
From ML/AI-based ideas to production: research

Research / Data Science / MLOps

- Ad-hoc prototyping of ideas
- Iterative experimentation
- Validation on a few use cases







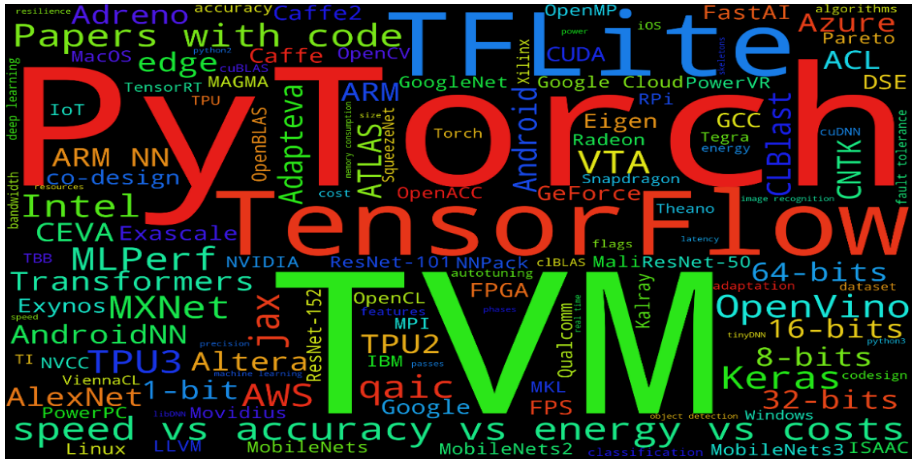
From ML/AI-based ideas to production: research + DevOps

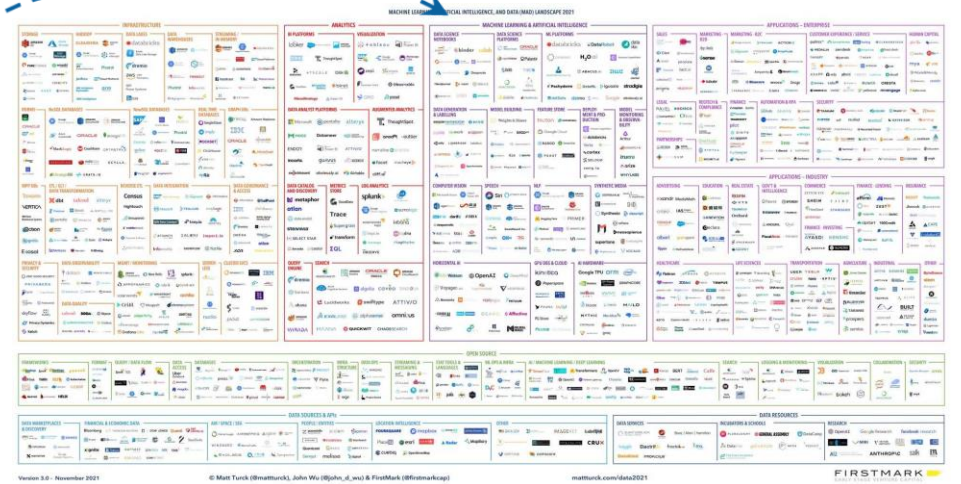
Research / Data Science / MLOps

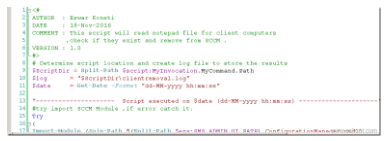
- Ad-hoc prototyping of ideas
- Iterative experimentation
- Validation on a few use cases

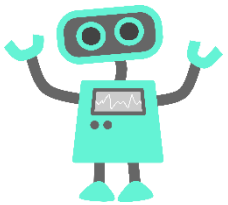
Engineering / DevOps

- Careful Planning
- Thorough testing
- Validation in the real world





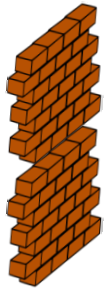




The gap between ML and Systems research and production is growing …

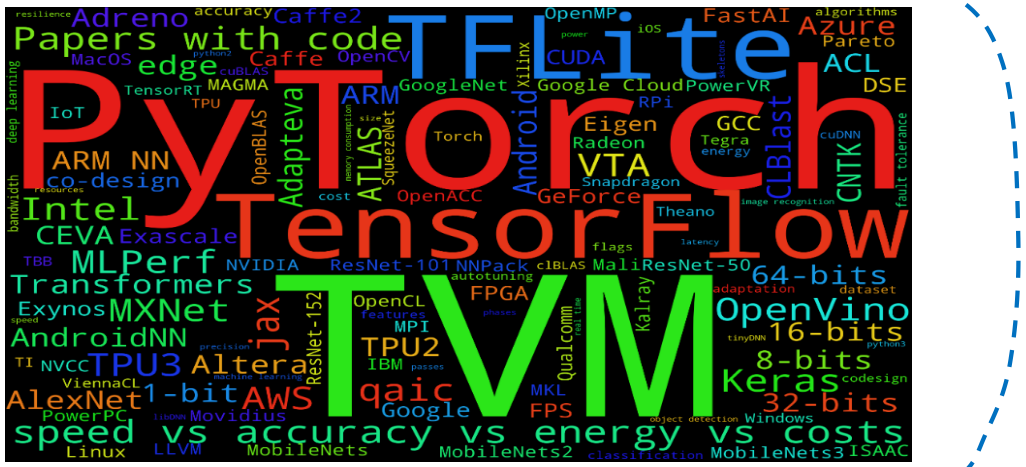
Research / Data Science / MLOps

- Ad-hoc prototyping of ideas
- Iterative experimentation
- Validation on a few use cases

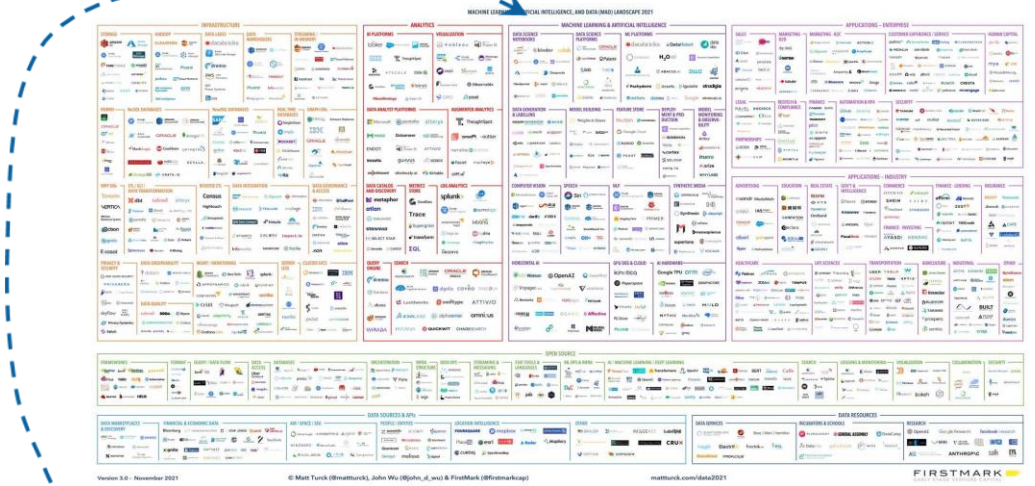


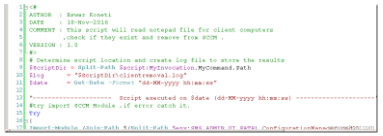
Engineering / DevOps

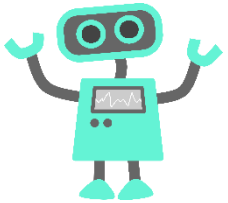
- Careful Planning
- Thorough testing
- Validation in the real world



Months
Years



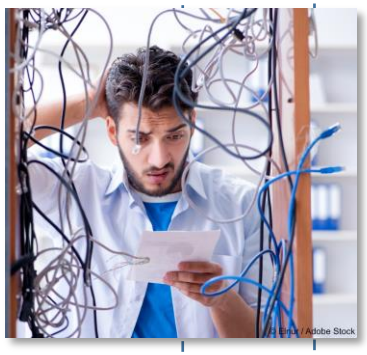




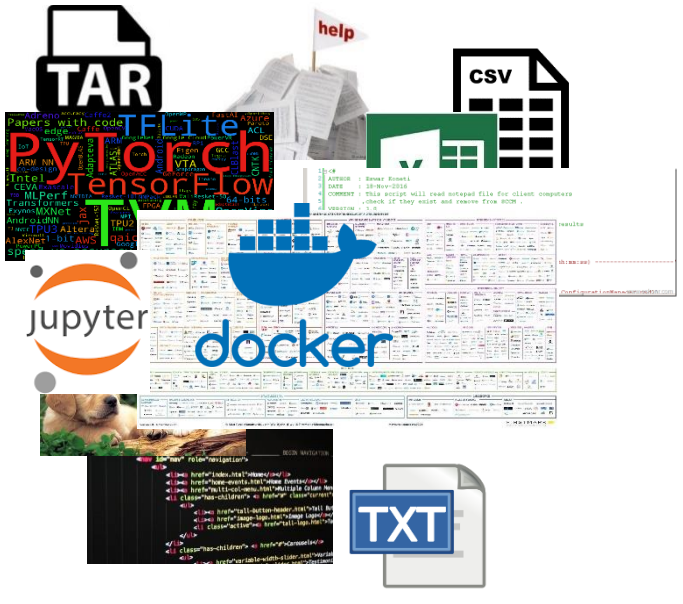
Public outcry: ML and Systems papers are difficult to reproduce; MLOps is a mess

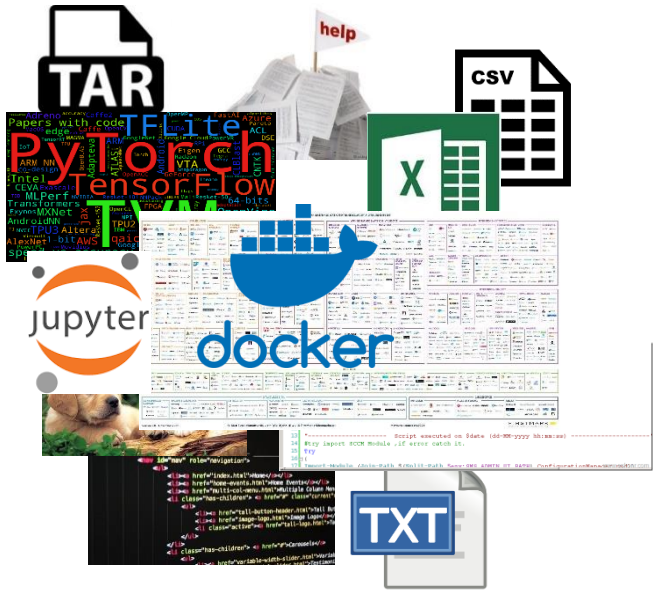
Team$_1$

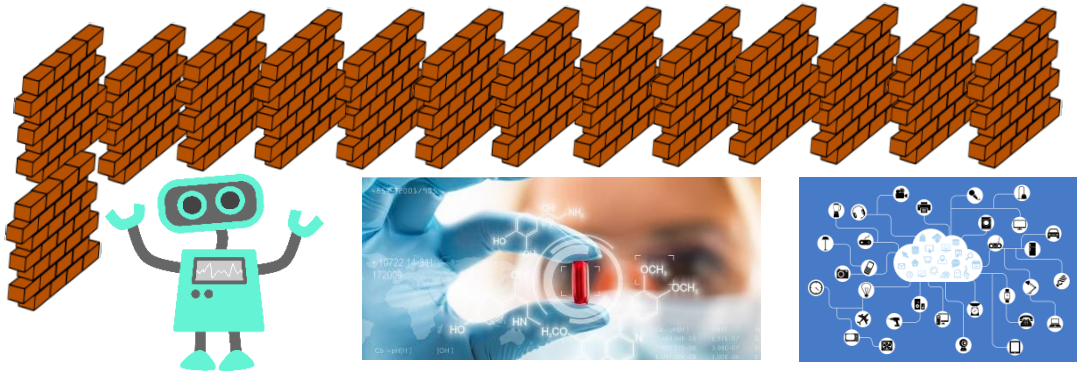
Team$_2$



Team$_3$







March 2022: www.mihaileric.com/posts/mlops-is-a-mess/

June 2020: arxiv.org/abs/2006.07161

2010-2014: Reproducibility studies and initiatives

reproducibility.cs.arizona.edu (weak reproducibility)

A comprehensive study of ~600 papers to examine if related code was shared and can be built

evaluate.inf.usi.ch/artifacts and artifact-eval.org (strong reproducibility)

The original and successful introduction of the artifact evaluation process at ACM conferencesArtifacts 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.

I've established cTuning.org/ae to learn why is it so difficult to reproduce results and validate them in the real world

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)

Bruce R. Childers, Grigori Fursin, Shriram Krishnamurthi, Andreas Zeller:
Artifact Evaluation for Publications (Dagstuhl Perspectives Workshop 15452). Dagstuhl Reports 5(11): 29-35 (2015)

2015-now: ACM, SC 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-now: introduced unified appendix and reproducibility checklist

My goal is was to learn how to automate artifact evaluation and make it easier to validate results in the real world

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, tags

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

cTuning.org/ae/checklist.html

cKnowledge.io/reproduced-papers

[dl.acm.org](dl.acm.org)

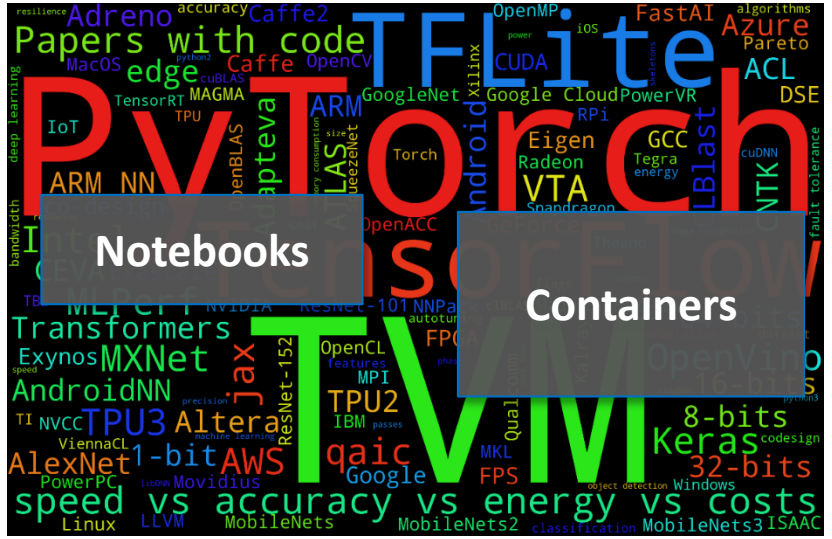
Learnings from reproducing 150+ research papers: cTuning.org/ae

- Sharing code, data, containers, PIP packages, Readme files and Jupyter notebooks is not enough to reproduce results particularly when we have to measure latency, throughput, power consumption, memory usage, accuracy and other characteristics across continuously changing systems.
    - Containers are useful to make stable snapshots but they hide the dependency hell rather than solving it and become quickly outdated.
    - Containers are often shared as a “black box” and we do not even know what is inside and how to connect them with external data and other MLOps and DevOps tools.
- **Unlike physics, there is no standard experimental methodology and evaluation criteria in computer engineering to ensure fair “apple-to-apple” comparison of different research techniques for latency, throughput, power consumption, accuracy, etc.**
- It take months to reproduce results from 1 paper and years to adopt novel techniques in production due to rapidly evolving software, hardware, APIs and data formats – many projects halt or fail when key people leave …

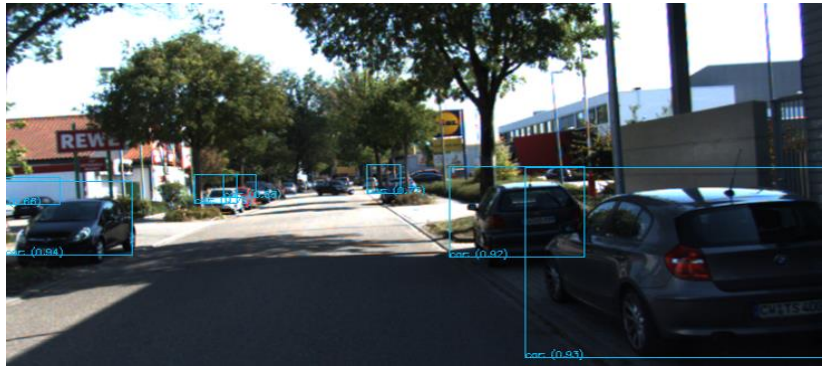
ACM TechTalk’21: www.youtube.com/watch?v=7zpeIVwICa4

The Real World

Rapidly evolving SW/HW stacks



Rapidly evolving algorithms, models and datasets



Learnings from reproducing 150+ research papers: cTuning.org/ae

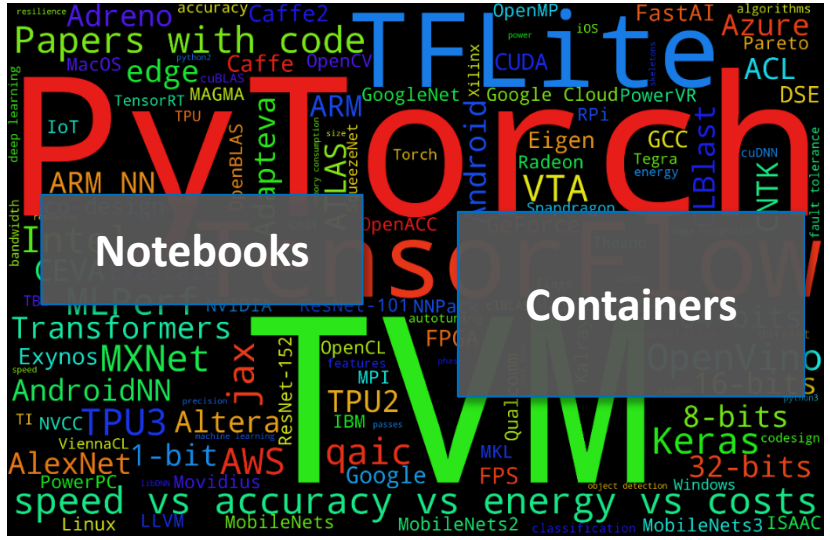
Most of the time is spent on reading ad-hoc readme files and performing the same, boring, repetitive and time-consuming tasks across continuously changing software and hardware including:

- detecting target hardware properties
- downloading various software and data
- detecting and/or installing numerous dependencies for a given host and target hardware
- substitute paths in numerous scripts, YAML/JSON files and code
- preparing or modifying numerous configuration files
- setting environment variables
- preprocessing datasets
- preparing command lines and running applications
- monitoring execution time, accuracy, energy, memory usage, etc
- post-processing results, recording them to a database, comparing with some reference ones
- visualizing and comparing results
- connecting applications with existing DevOps and MLOps tools
- packing all those ad-hoc scripts and artifacts into containers to give to other teams

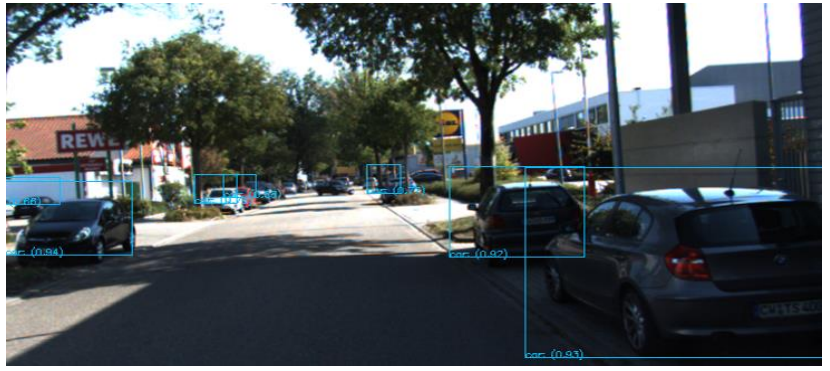
ACM TechTalk'21: www.youtube.com/watch?v=7zpeIVwICa4

The Real World

Rapidly evolving SW/HW stacks



Rapidly evolving algorithms, models and datasets



Learnings from reproducing 150+ research papers: cTuning.org/ae

Most of the time is spent on reading ad-hoc readme files and performing the same, boring, repetitive and time-consuming tasks acros

- de
- dc
- de
  ta
- su
- pr
- se
- pr
- pr
- m
- pc
  so
- vis
- co
- packing all those ad-hoc scripts and artifacts into containers to give to other teams

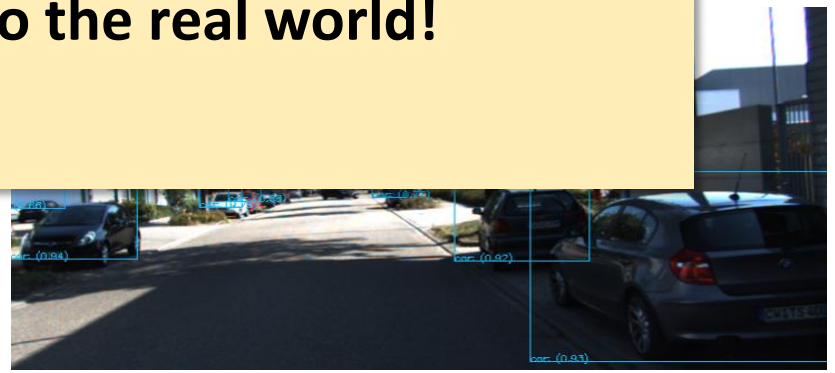
Quite challenging to read all ReadMe files and Jupyter notebooks from 50K+ ML and Systems tech. reports and papers published every year together with 100M+ GitHub repositories and 10M+ Docker containers !

Not practical and not scalable!

That's why most research ideas do not make it to the real world!

The Real World

Rapidly evolving SW/HW stacks



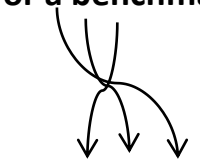
ACM TechTalk'21: www.youtube.com/watch?v=7zpeIVwICa4

2015: Artifact Appendix have helped me to notice *common* patterns across *different* R&D projects

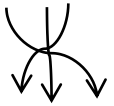
Artifact Appendix from 150+ papers

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

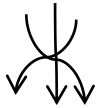
Ad-hoc scripts to compile and run a program or a benchmark



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

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

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

Have some common info: *configuration, compilation, linking and optimization flags*

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

Have some common info: *filename, size, width, height, colors, …*

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

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

Could be reused across projects | **Common automations** | **Common objects** | **Common description**

How can we automate generation and parsing of ReadMe files and Jupyter notebooks?

Person$_1$ or Team$_1$ produces and shares some artifacts and knowledge

Some ad-hoc directory and file structure:

/ project root / **program** / mlperf-image-classification / scripts to run MLPerf IC benchmark
/ mlperf-bert / scripts to run MLPerf BERT benchmark

/ project root / **package** / mlperf-loadgen / scripts to download and install MLPerf benchmark
/ dataset-imagenet / scripts to download ImageNet

/ project root / **script** / imagenet-pre-post-processing / scripts to pre and postprocess ImageNet

/ project root / **experiment** / mlperf-inference-v1.1-octoml / some files with benchmark results

README

jupyter

Person$_2$ or Team$_2$ needs to consume others' artifacts and knowledge

?

No idea what is inside and how to use it…

2015 - cur: Collective Knowledge concept (CK)

Person$_1$ or Team$_1$ produces and shares some artifacts and knowledge

Let's treat **all** shared projects as a database of reusable artifacts and automations

/ project root / **program** / mlperf-image-classification / scripts to run MLPerf IC benchmark
/ mlperf-bert / scripts to run MLPerf BERT benchmark

/ project root / **package** / mlperf-loadgen / scripts to download and install MLPerf benchmark
/ dataset-imagenet / scripts to download ImageNet

/ project root / **script** / imagenet-pre-post-processing / scripts to pre and postprocess ImageNet

/ project root / **experiment** / mlperf-inference-v1.1-octoml / some files with benchmark results

README

jupyter

Person$_2$ or Team$_2$ needs to consume others' artifacts and knowledge

CK framework: provides Python API, CLI and web service to manage and share research projects as a database of reusable artifacts and automations

arxiv.org/abs/2011.01149

2015 - 2021: CK proof-of-concept (2022: new CK2 framework in development)

Collective Knowledge Framework:
github.com/ctuning/ck

A simple Python library and CLI
with minimal dependencies
to manage research projects
as a database of reusable components

ck {action} {module}:{component} @input.json

```
pip install ck
ck pull repo:mlcommons@ck-mlops
ck add ck-mlops:dataset:imagenet –tags=2012
ck ls dataset:*imagenet*
ck search dataset --tags=2012
ck find package:imagenet-2012-train
ck find 1dc07ee0f4742028:b4f26f2ca41539d9
  ~CK/ck-ml/package/imagenet-2012-train
ck search package --tags=compiler,tvm
ck rm dataset:imagenet
```

Artifact automated and reusable

Collective Knowledge COMPATIBLE

CK-compatible research project (all objects has Unique IDs)!

/ 1st level dirs	/ 2nd level dirs	/ 3rd level dirs	
program	image corner detection	meta.json	info.json
program	matmul OpenCL	meta.json	info.json
program	object detection CUDA	meta.json	info.json
soft	GCC V9.3	meta.json	info.json
soft	LLVM V11.1	meta.json	info.json
soft	Intel Compilers 2021	meta.json	info.json
dataset	image-jpeg-0001	meta.json	info.json
dataset	bzip2-0006	meta.json	info.json
dataset	txt-0012	meta.json	info.json
dataset	video-raw-1280x1024	meta.json	info.json
experiment	cvs speedups	meta.json	info.json
experiment	txt hardware counters	meta.json	info.json
experiment	xls table with graphs	meta.json	info.json

2015 – cur: started encoding repetitive tasks from ML and Systems papers as CK automation recipes and objects

1) **Describe different operating systems**

```
ck pull repo:ck-ml
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,gcc
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
```

5) Run portable program workflow

```
ck run program:mlperf-inference-image-classification
```

cKnowledge.io/modules **cKnowlege.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": {...}
}
```

Find module:

```
ck find module:platform
```

Add new module with an action:

```
ck add module:octomizer
ck add_action module:octomizer --func=run
ck run octomizer:model-mlperf-ssd-mobilenet
```

CK can be used to generate Docker containers and can be used inside containers to automate MLOps

github.com/mlcommons/ck-mlops/tree/main/docker

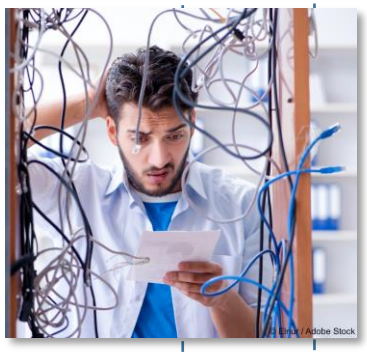
github.com/mlcommons/ck-mlops/blob/main/docker/ck-mlperf-inference-dev-image-classification-onnx-tvm/Dockerfile.ubuntu-20.04

```
FROM ubuntu:20.04
…
# Install CK
RUN ck pip3 install ck

# Clone private CK repo
RUN ck pull repo:mlcommons@ck-mlops

# Install packages to CK env entries
RUN ck setup kernel --var.install_to_env=yes

RUN ck detect platform.os --platform_init_uoa=generic-linux-dummy
RUN ck detect soft:compiler.python --full_path=/usr/bin/python3
RUN ck detect soft:compiler.gcc --full_path=`which gcc`
RUN python3 -m pip install protobuf

RUN ck install package --tags=mlperf,inference,src,octoml.dev
RUN ck install package --tags=lib,python-package,mlperf,loadgen
RUN ck install package --tags=imagenet,2012,val,min,non-resized
RUN ck install package --tags=imagenet,2012,aux,from.berkeley
RUN ck install package --tags=lib,python-package,onnxruntime-cpu,1.7.0
RUN ck install package --tags=lib,python-package,onnx,1.9.0

RUN ck install package --tags=model,mlperf,onnx,resnet50,v1.5-opset-11
RUN ck install package --tags=lib,python-package,scipy
RUN ck install package --tags=tool,cmake,prebuilt,v3.18.2
RUN ck install package --tags=compiler,llvm,prebuilt,v12.0.0

RUN ck install package --tags=compiler,tvm,dev --
env.CK_HOST_CPU_NUMBER_OF_PROCESSORS=4

# Install MLPerf task requirements
RUN ck run program:mlperf-inference-bench-image-classification-tvm-onnx-cpu --
cmd_key=install-python-requirements

# Run TVM-based MLPerf inference benchmark (Offline;Accuracy)
CMD ck run program:mlperf-inference-bench-image-classification-tvm-onnx-cpu \
        --cmd_key=accuracy-offline \
        --env.EXTRA_OPS="--thread 1 --max-batchsize 1"
```

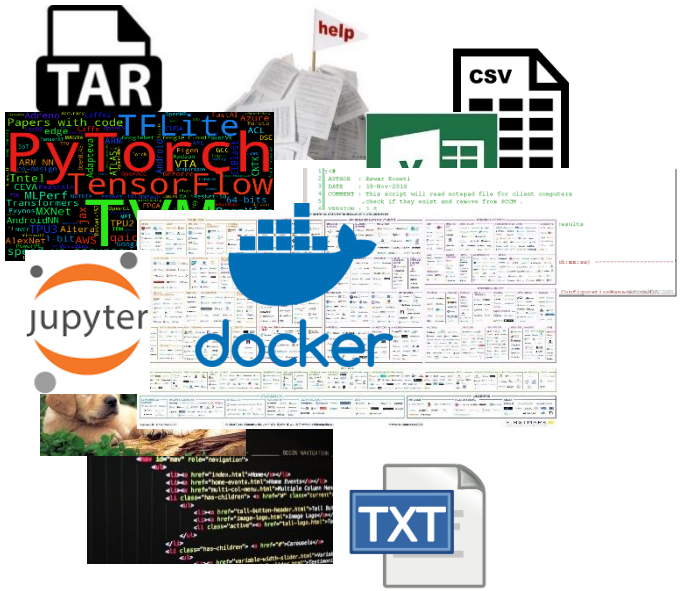
Can we make it easier to reproduce research techniques and bring them to production?

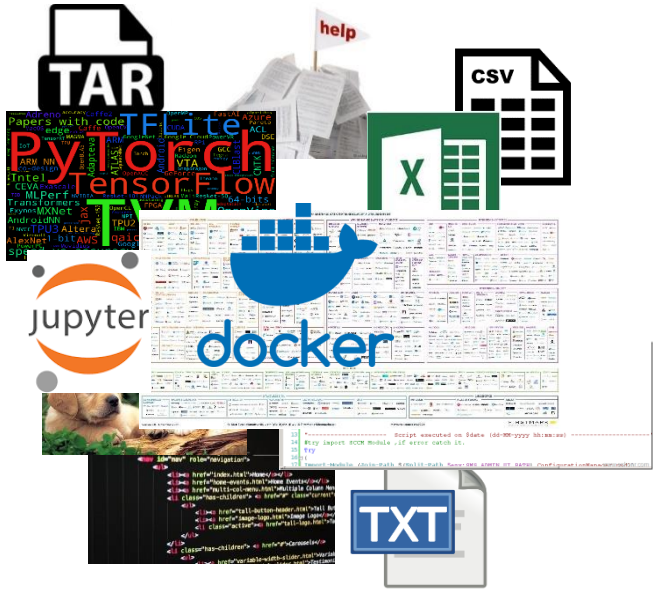
Team$_1$

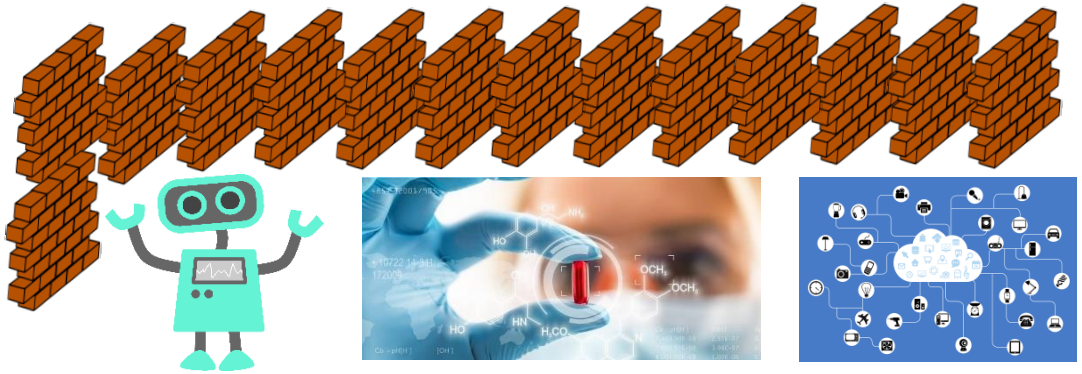
Team$_2$



Team$_3$







March 2022: www.mihaileric.com/posts/mlops-is-a-mess/

June 2020: arxiv.org/abs/2006.07161

Sharing artifacts and knowledge in a unified and reusable way automate and simplifies this process!

Team$_1$

Artifact automated and reusable Collective Knowledge COMPATIBLE

CK-compatible research project (all objects has Unique IDs)!

/ 1st level dirs	/ 2nd level dirs	/ 3rd level dirs	
program	image corner detection	meta.json	info.json
program	matmul OpenCL	meta.json	info.json
program	object detection CUDA	meta.json	info.json
soft	GCC V9.3	meta.json	info.json
soft	LLVM V11.1	meta.json	info.json
soft	Intel Compilers 2021	meta.json	info.json
dataset	image-jpeg-0001	meta.json	info.json
dataset	bzip2-0006	meta.json	info.json
dataset	txt-0012	meta.json	info.json
dataset	video-raw-1280x1024	meta.json	info.json
experiment	cvs speedups	meta.json	info.json
experiment	txt hardware counters	meta.json	info.json
experiment	xls table with graphs	meta.json	info.json

CK
API / CLI

Team$_2$

Artifact automated and reusable Collective Knowledge COMPATIBLE

CK-compatible research project (all objects has Unique IDs)!

/ 1st level dirs	/ 2nd level dirs	/ 3rd level dirs	
program	image corner detection	meta.json	info.json
program	matmul OpenCL	meta.json	info.json
program	object detection CUDA	meta.json	info.json
soft	GCC V9.3	meta.json	info.json
soft	LLVM V11.1	meta.json	info.json
soft	Intel Compilers 2021	meta.json	info.json
dataset	image-jpeg-0001	meta.json	info.json
dataset	bzip2-0006	meta.json	info.json
dataset	txt-0012	meta.json	info.json
dataset	video-raw-1280x1024	meta.json	info.json
experiment	cvs speedups	meta.json	info.json
experiment	txt hardware counters	meta.json	info.json
experiment	xls table with graphs	meta.json	info.json

CK
API / CLI

Team$_3$

Artifact automated and reusable Collective Knowledge COMPATIBLE

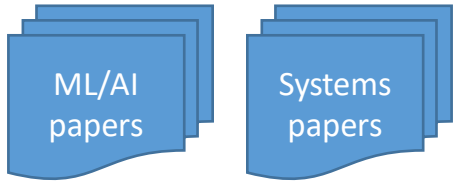
CK-compatible research project (all objects has Unique IDs)!

/ 1st level dirs	/ 2nd level dirs	/ 3rd level dirs	
program	image corner detection	meta.json	info.json
program	matmul OpenCL	meta.json	info.json
program	object detection CUDA	meta.json	info.json
soft	GCC V9.3	meta.json	info.json
soft	LLVM V11.1	meta.json	info.json
soft	Intel Compilers 2021	meta.json	info.json
dataset	image-jpeg-0001	meta.json	info.json
dataset	bzip2-0006	meta.json	info.json
dataset	txt-0012	meta.json	info.json
dataset	video-raw-1280x1024	meta.json	info.json
experiment	cvs speedups	meta.json	info.json
experiment	txt hardware counters	meta.json	info.json
experiment	xls table with graphs	meta.json	info.json

CK
API / CLI



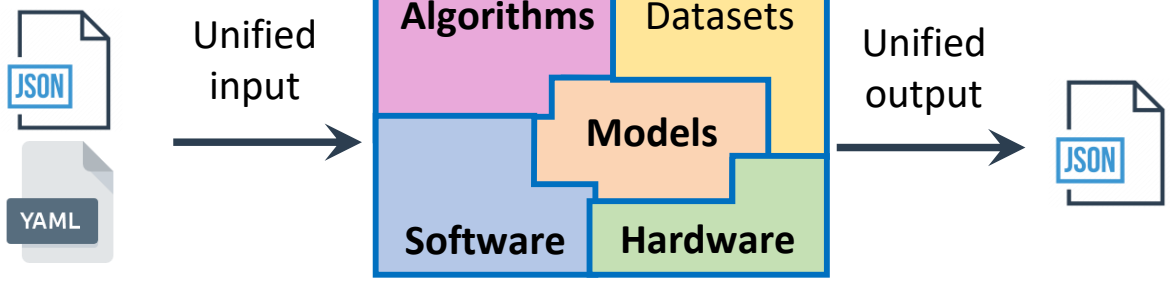

cKnowledge.io: aggregated 1000+ open-source CK components to automate ML & Systems R&D



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

CK JSON API	CK JSON API	CK JSON API	CK JSON API	CK JSON API
Algorithms	Models	Software	Data sets	Hardware
Object detection	MobileNets	TVM	ImageNet	CPU
Object classification	ResNet	PyTorch	KITTI	GPU
Speech recognition	VGG	TensorFlow	VOC	TPU
...	...	...	...	...
Training/inference	SqueezeDet	TFLite	Real data sets	NN accelerators
...	...	...	...	...

GUI to https://github.com/mlcommons/ck-mlops

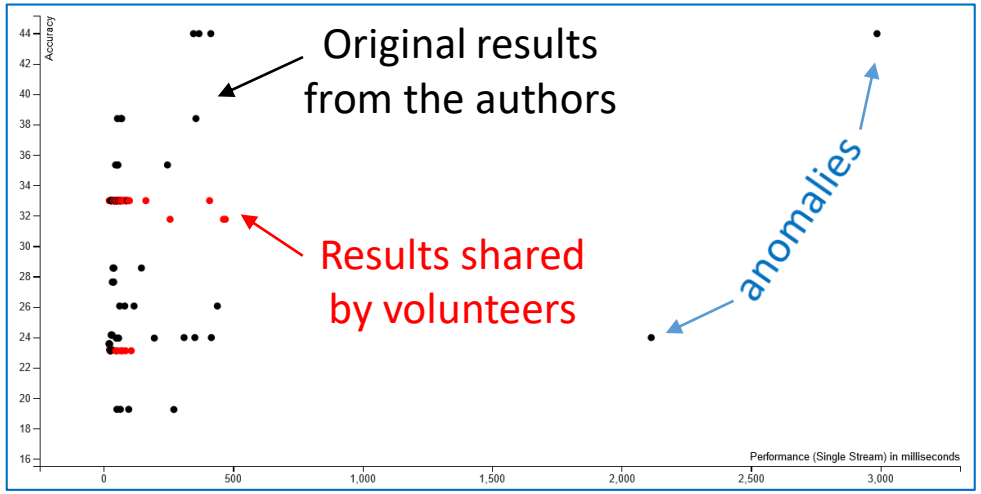


Adaptive containers and portable workflows with plug&play components, portable workflows, common APIs, and unified I/O

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

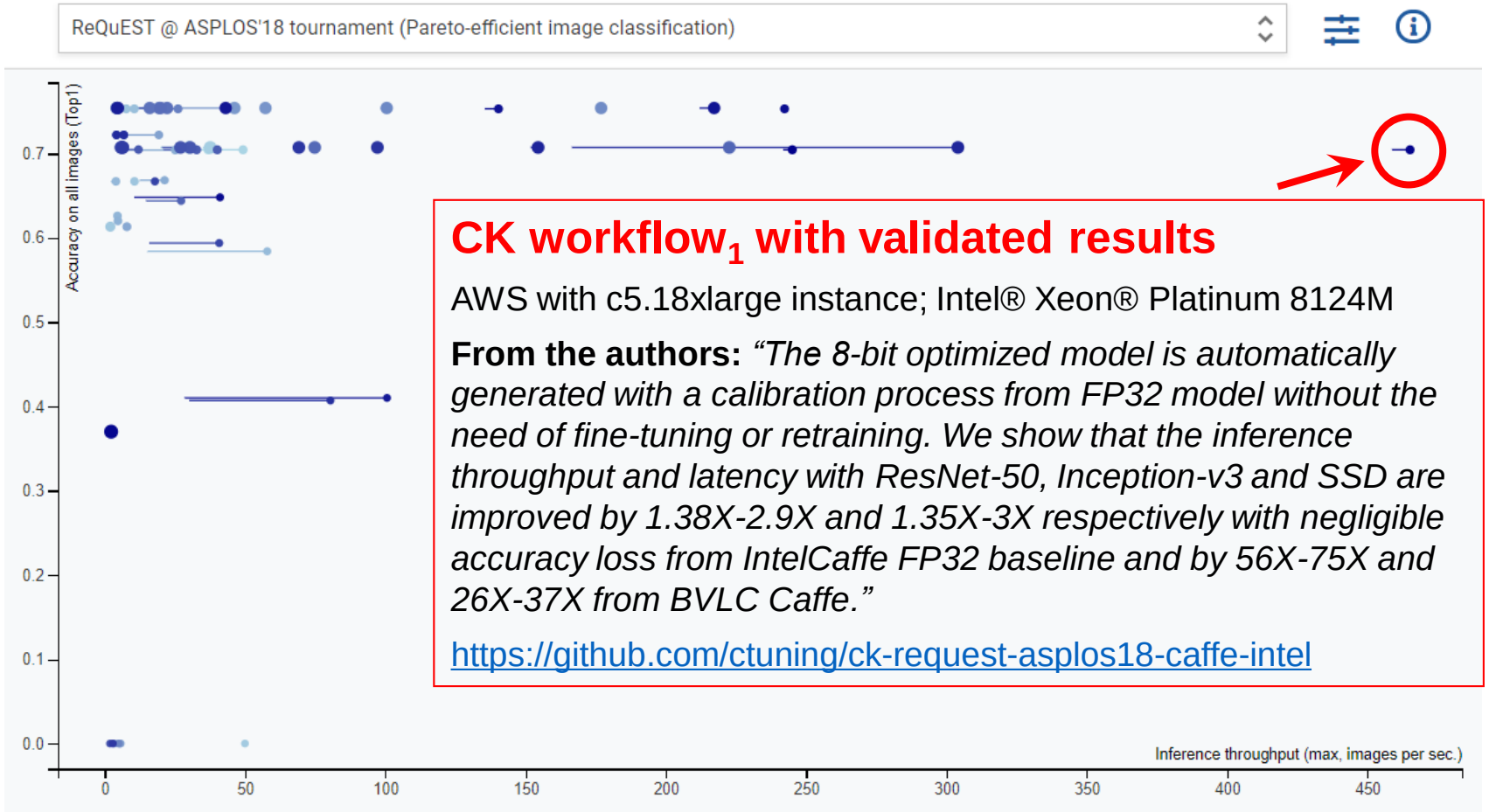


CK-based Deep learning optimization tournament at ASPLOS'18

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

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

cKnowledge.io/results

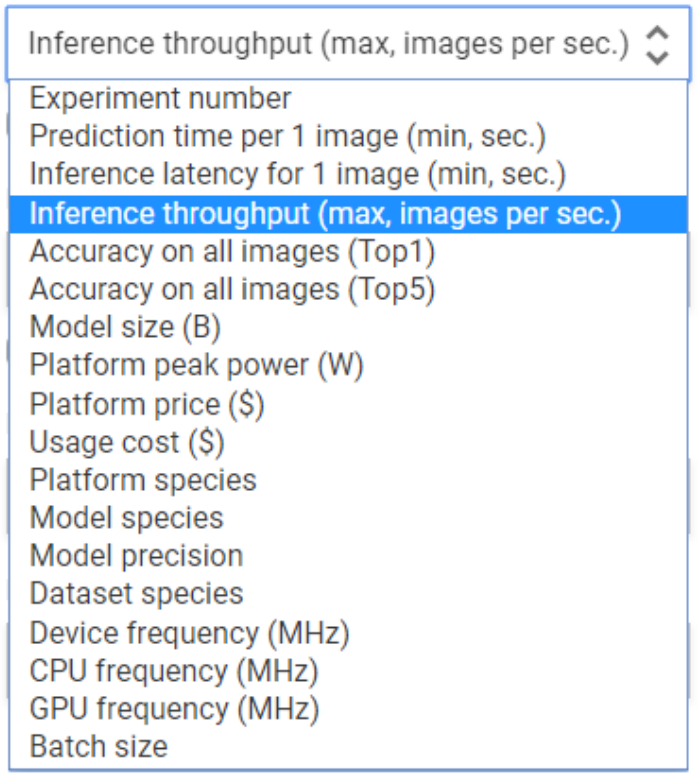


CK workflow$_1$ with validated results

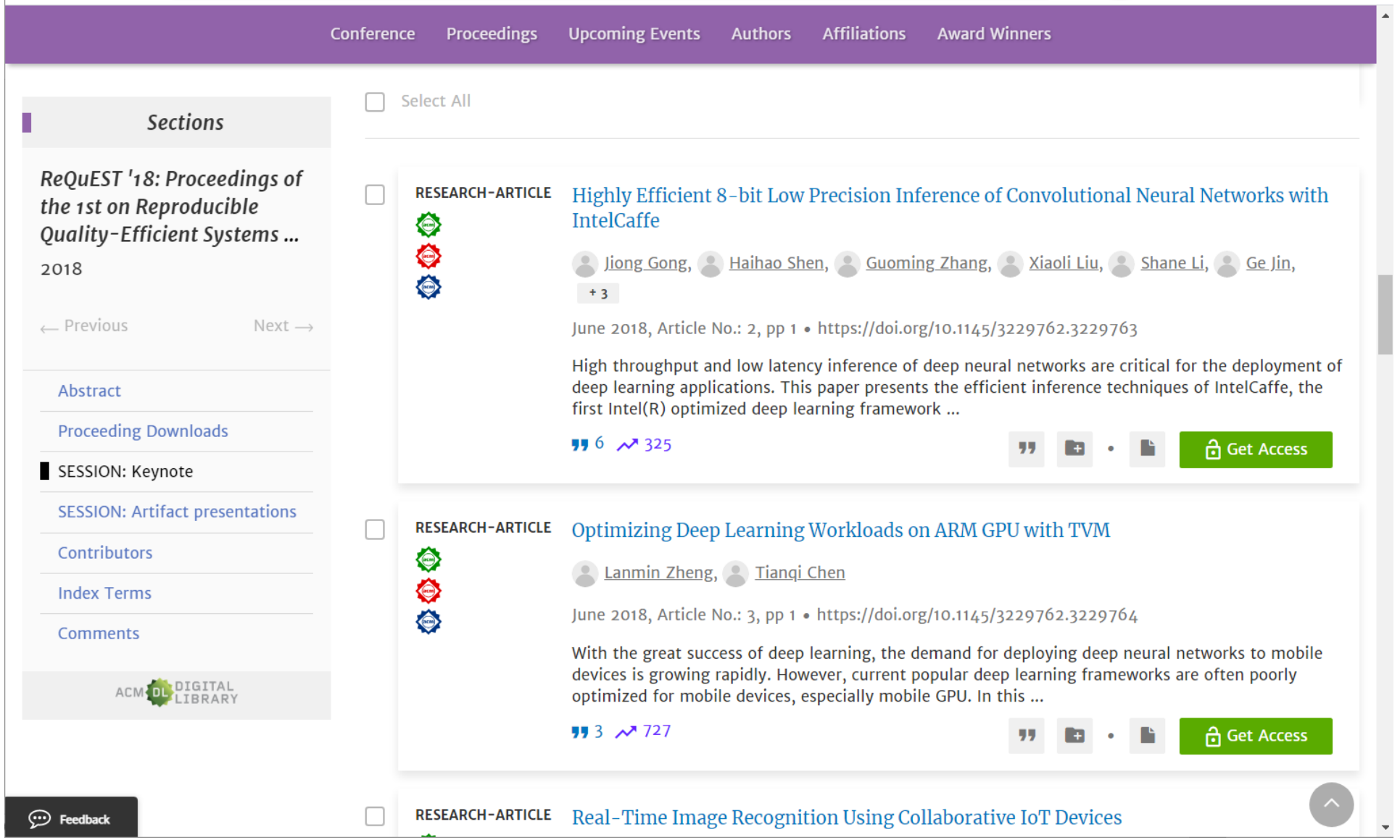
AWS with c5.18xlarge instance; Intel® Xeon® Platinum 8124M

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

https://github.com/ctuning/ck-request-asplos18-caffe-intel



Published validated papers with reusable workflows in the ACM DL

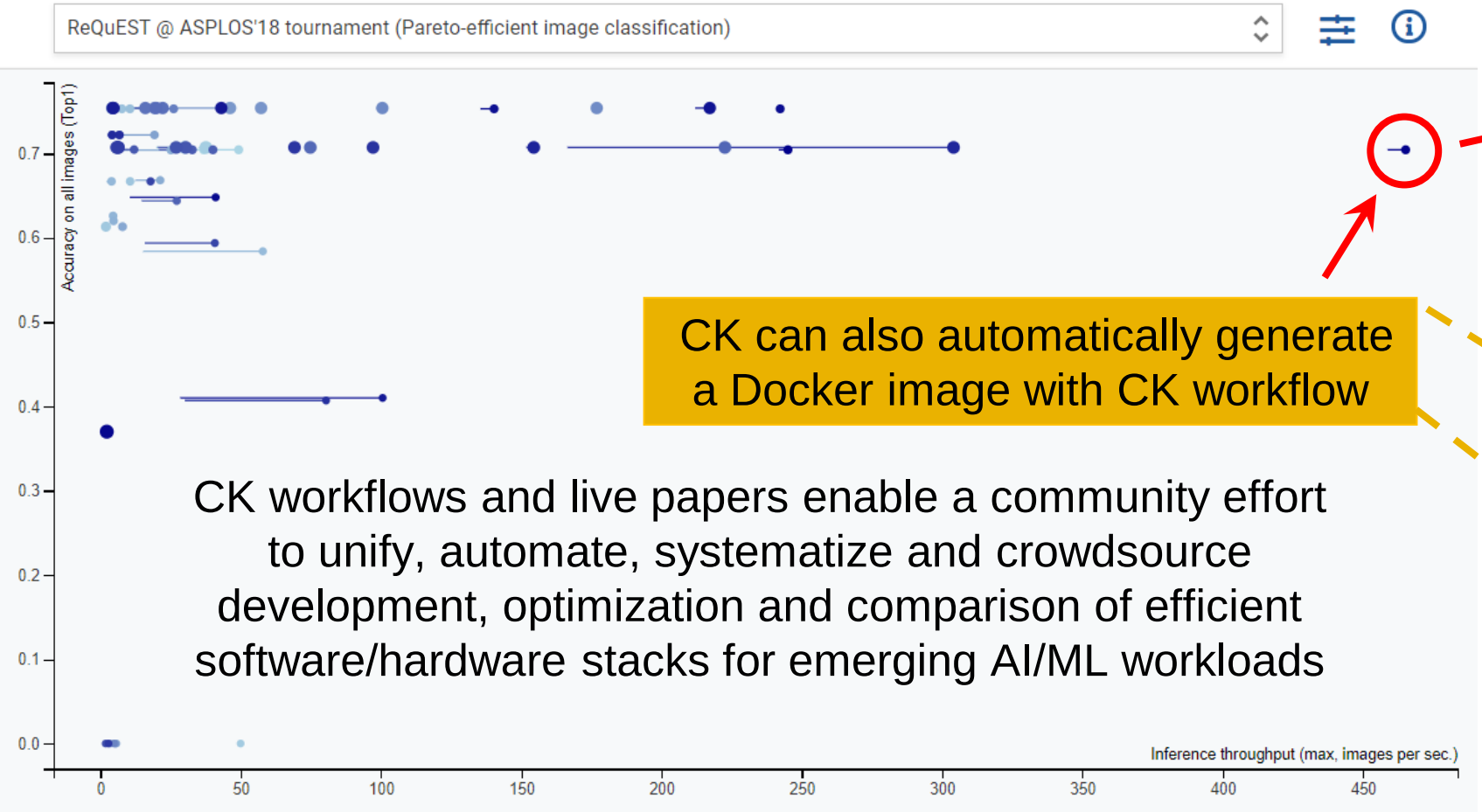


dl.acm.org/doi/proceedings/10.1145/3229762

CK workflows + Docker containers made it easier to bring research ideas 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



CK can also automatically generate a Docker image with CK workflow

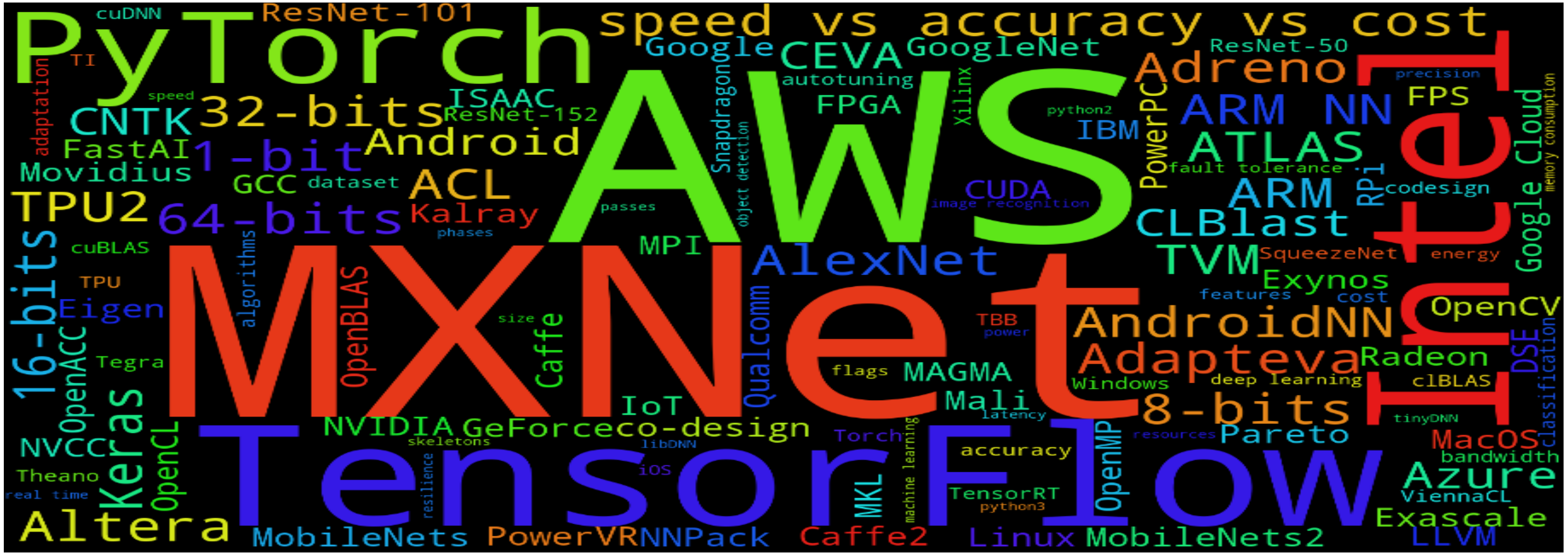
CK workflows and live papers enable a community effort to unify, automate, systematize and crowdsource development, optimization and comparison of efficient software/hardware stacks for emerging AI/ML workloads

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

amazon web services

2019-cur: Using CK to modularize MLPerf inference benchmark

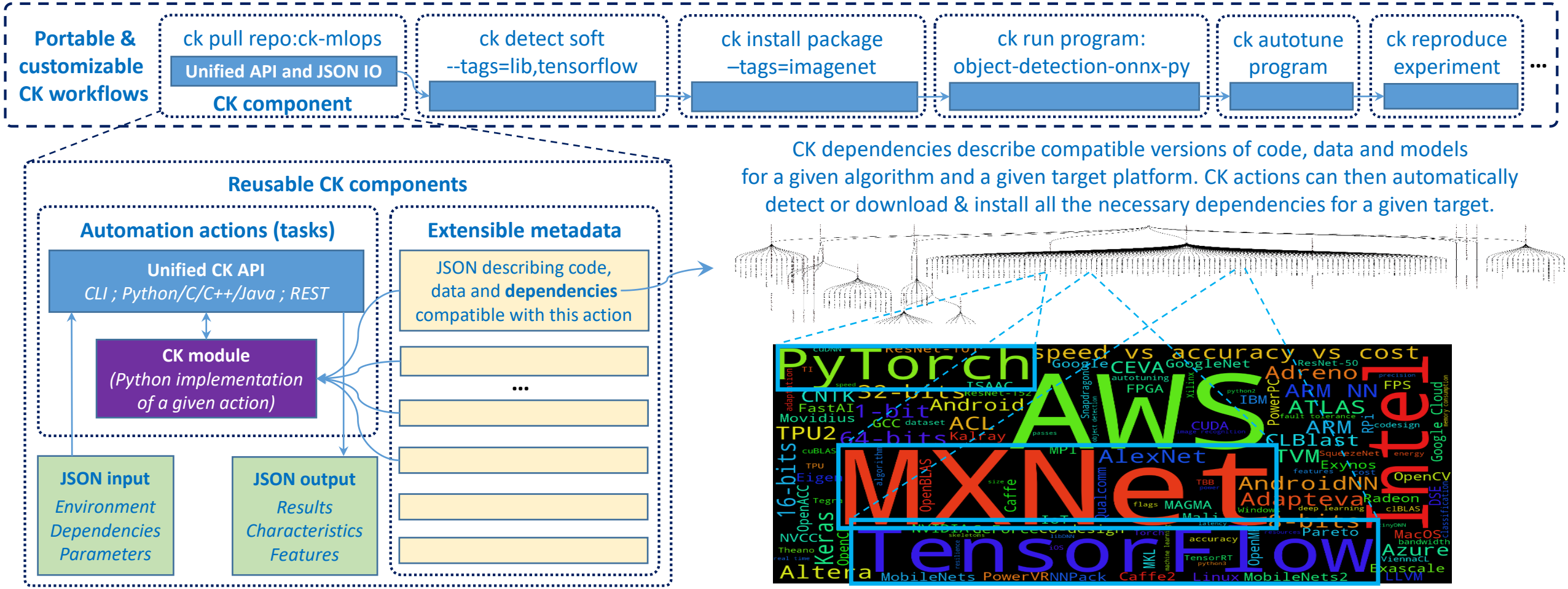
Very complex and time-consuming process to prepare, submit and reproduce results
across rapidly evolving Ml/SW/HW stacks – must be simplified to attract more submitters!



2019-cur: Using CK to modularize MLPerf inference benchmark

We've prototyped CK-based MLPerf workflows
demonstrating that it was possible to automate ML/SW/HW DSE and MLPerf inference benchmark submission

github.com/mlcommons/ck/tree/master/docs/mlperf-automation

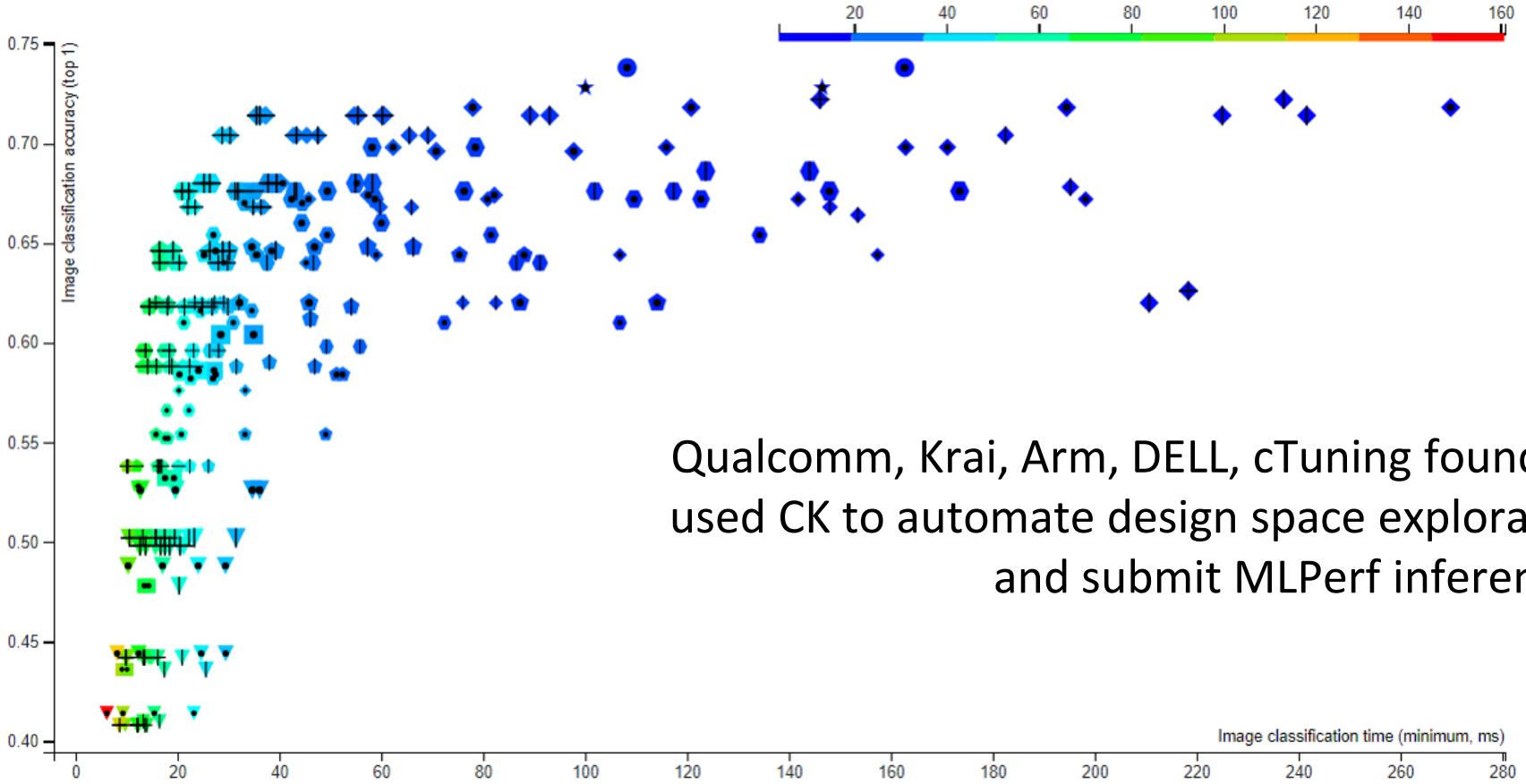


CK dependencies describe compatible versions of code, data and models for a given algorithm and a given target platform. CK actions can then automatically detect or download & install all the necessary dependencies for a given target.

CK is successfully used to prepare MLPerf inference submissions from cloud to edge

"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.



Qualcomm, Krai, Arm, DELL, cTuning foundation, OctoML and others used CK to automate design space exploration of ML/SW/HW stacks and submit MLPerf inference results

Continue CK-related developments within MLCommons as a community effort

OctoML.ai and the cTuning foundation have joined MLCommons
to help modularize MLPerf inference benchmark and automate
submissions across diverse ML frameworks, models, datasets
and platforms from cloud to edge

OctoML.ai and the cTuning foundation have donated the CK framework
and CK-based MLPerf workflows to MLCommons
to help the community modularize MLPerf benchmark,
automate design space exploration, and share results in a reproducible and deployable format:

github.com/mlcommons/ck github.com/mlcommons/ck-mlops

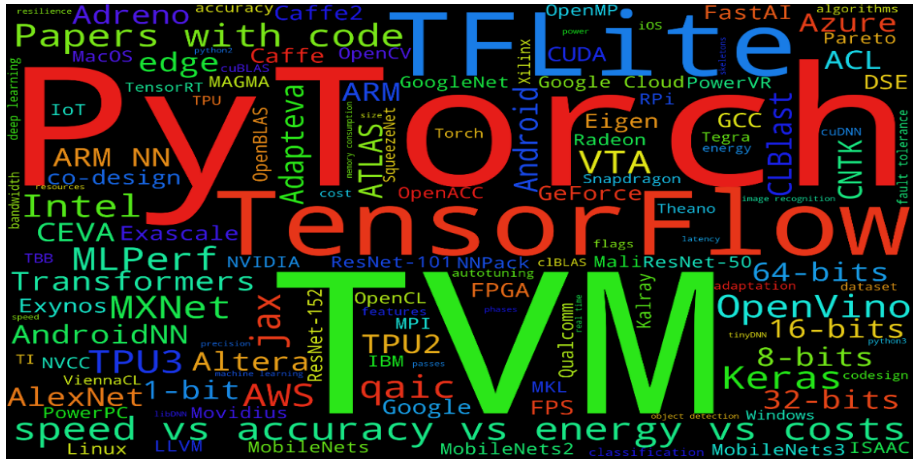
CK2: a new community project to learn how to bridge the gap between ML research and production

Research / Data Science / MLOps

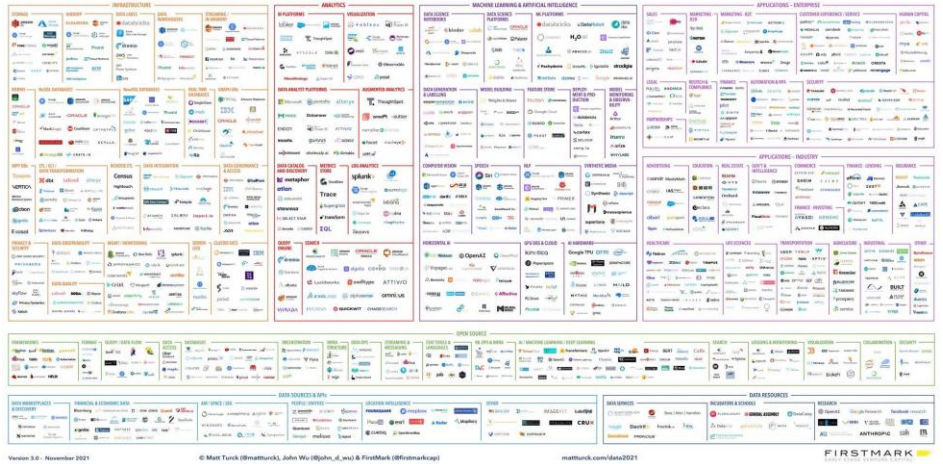
- Quick prototyping of ideas
- Quick experimentation
- Validation on a few use cases

Engineering / DevOps

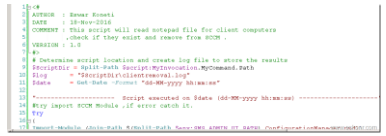
- Careful Planning
- Thorough testing
- Validation in the real world



CK2 aka Collective Mind



github.com/mlcommons/ck/tree/master/ck2



Join our community effort to develop the CK2 framework and modularize AI and ML

Short term goal

- Develop the 2nd version of the CK framework (CM) with the community to modularize AI and ML based on 5 years of practical CK experience.
  - Community prototype: https://github.com/mlcommons/ck/tree/master/ck2
- Use CM to modularize MLPerf inference and generate MLCube containers
- Develop CM-based version of MLPerf inference reference models, initially as proof of concept

Long term goals

- Make it easier to customize, run, test and reproduce MLPerf inference benchmarks across continuously evolving software, hardware, models and datasets.
- Automate Design Space Exploration of ML/SW/HW stacks to trade off performance, accuracy, energy, size and costs
- Automate submission of Pareto-efficient ML Systems to MLPerf inference open division
- Develop an open database of allowed MLPerf benchmark configurations, benchmarking results and provenance information with an UI for visualization and analysis.

Cross-org collaborations

- Promote MLCommons activities and technology in ACM/IEEE/NeurIPS reproducibility initiatives

We are considering creating a new MLPerf WG on DSE and production deployment:

Contact grigori@octoml.ai if you are interested to participate and co-lead…

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 Hetherignton, 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, Nadine Mendelek, Thierry Moreau, Dewey Murdick, Mircea Namolaru, Luigi Nardi, Cedric Nugteren, Michael O'Boyle, Ivan Ospiov, 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, Sergey Serebryakov, Aaron Smith, Jim Spohrer, Michel Steuwer, Victoria Stodden, Robert Stojnic, Arjun Suresh, 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, Thomas Zhu

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

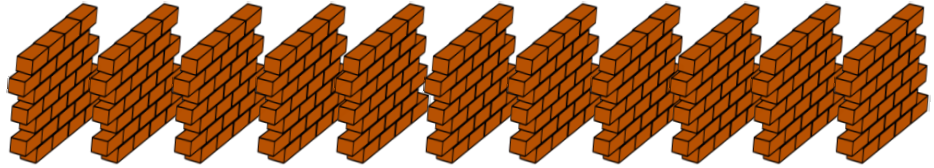
CK collaborators: cKnowledge.io/partners

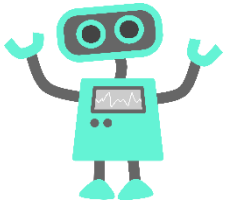
OctoML.ai for supporting MLCommons and the development of the CK2 framework

Let’s learn together how to bridge the growing gap between research and production

Let’s learn how to share and reuse our knowledge, experience and best practices about co-designing, benchmarking, optimizing and deploying Pareto-efficient ML Systems in production

Having a common optimization and deployment infrastructure for ML Systems that unifies and interconnects existing technologies rather than substituting or rewriting them will help accelerate innovation and reduce the time to market for intelligent and Pareto-efficient systems





github.com/mlcommons/ck/tree/master/ck2

grigori@octoml.ai cKnowledge.io/@gfursin