Reproducibility and frugality in AI benchmarking: lessons from Benchopt

Thomas Moreau

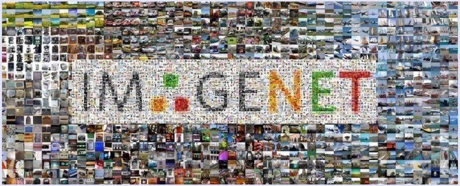
A word about me

- Researcher at Inria – MIND
- `https://tommoral.github.io`
- thomas.moreau@inria.fr
- **Research topics:** Time-series, Physiological signals, Inverse Problems, Bilevel Optimization, Unrolling, Pattern Learning, Point Processes, .
- **OSS maintainer/contributor**

The Era of Benchmarks: AI as an emprical science

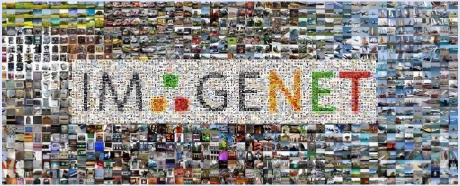
The ImageNet competition

- Annual competition since 2010
- Evaluate image classification methods with 14M labeled images among 1k categories



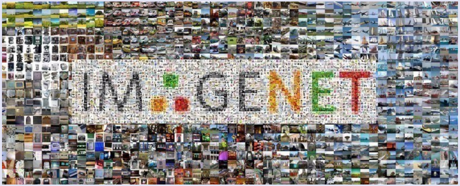
The ImageNet competition

- Annual competition since 2010
- Evaluate image classification methods with 14M labeled images among 1k categories
- **Boosted AI and Deep Learning research when Alex Krizhevsky won in 2012.**



The ImageNet competition

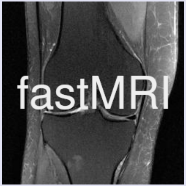
- Annual competition since 2010
- Evaluate image classification methods with 14M labeled images among 1k categories
- **Boosted AI and Deep Learning research when Alex Krizhevsky won in 2012.**



$\Rightarrow$ Demonstrates the importance of benchmarks to drive research in AI.

Many benchmarks followed ImageNet:

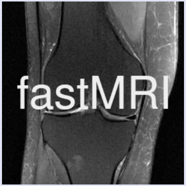
- Natural Language Processing: GLUE, SuperGLUE
- Reinforcement Learning: Atari, MuJoCo, OpenAI Gym
- Others: fastMRI, DAWNBench, MLPerf, etc.



Many benchmarks in AI

Many benchmarks followed ImageNet:

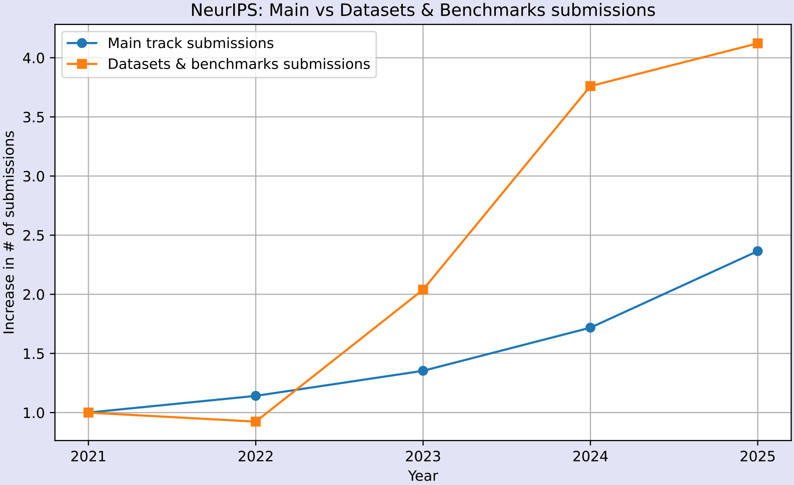
- Natural Language Processing: GLUE, SuperGLUE
- Reinforcement Learning: Atari, MuJoCo, OpenAI Gym
- Others: fastMRI, DAWNBench, MLPerf, etc.



$\Rightarrow$ Benchmarks are now ubiquitous in AI research.

Too many benchmarks in AI?

In the recent years, many benchmarks have been proposed:



Benchmark goals in AI

	Short-term progress	**Long-term evaluation**
Task-specific	Challenge/Competition $\rightarrow$ push limits quickly	SOTA tracking $\rightarrow$ measure progress
Generalizable	Research question $\rightarrow$ empirical study	Benchmark framework $\rightarrow$ stable & extensible

Benchmark goals in AI

	Short-term progress	**Long-term evaluation**
Task-specific	Challenge/Competition $\rightarrow$ push limits quickly	SOTA tracking $\rightarrow$ measure progress
Generalizable	Research question $\rightarrow$ empirical study	Benchmark framework $\rightarrow$ stable & extensible

Takeaway

Most attention goes to the top-left quadrant for fast progress, but solid science requires the bottom-right.

The three components of a benchmark

A benchmark is defined by three components:

- **Objective**: what is being measured?
- **Dataset**: on what evidence?
- **Solvers/Methods**: What are we comparing?

The three components of a benchmark

A benchmark is defined by three components:

- **Objective**: what is being measured?
- **Dataset**: on what evidence?
- **Solvers/Methods**: What are we comparing?

Different benchmark goals emphasize different components.

- **Challenge benchmarks:** Fixed dataset + single metric $\rightarrow$ stresses solver comparison. Focus on performance

The three components of a benchmark

A benchmark is defined by three components:

- **Objective**: what is being measured?
- **Dataset**: on what evidence?
- **Solvers/Methods**: What are we comparing?

Different benchmark goals emphasize different components.

- **Challenge benchmarks:** Fixed dataset + single metric $\rightarrow$ stresses solver comparison. Focus on performance
- **SOTA tracking benchmarks:** Multiple metrics and datasets, compare solvers to measure progress over time.

  Risk: test-set overfitting / cherry-picked metrics.

The three components of a benchmark

A benchmark is defined by three components:

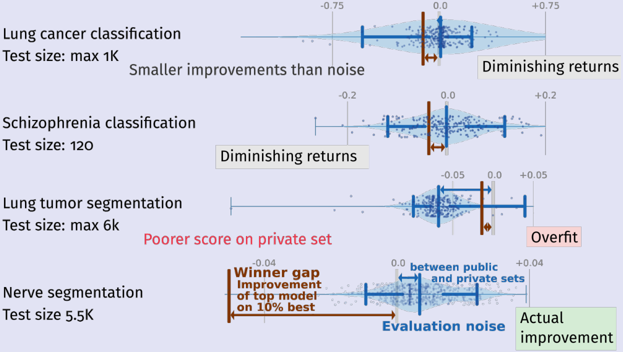
- **Objective**: what is being measured?
- **Dataset**: on what evidence?
- **Solvers/Methods**: What are we comparing?

Different benchmark goals emphasize different components.

- **Challenge benchmarks:** Fixed dataset + single metric $\rightarrow$ stresses solver comparison. Focus on performance
- **SOTA tracking benchmarks:** Multiple metrics and datasets, compare solvers to measure progress over time. Risk: test-set overfitting / cherry-picked metrics.
- **Research benchmarks:** Fixed set of methods evaluated, with broad range of metrics. Risk: incomplete / quickly outdated

[Varoquaux and Cheplygina 2022]

- Futile benchmarks



Challenges in AI benchmarking

- Futile benchmarks
- Lack of proper baselines hinders scientific progress.

Do we really need Foundation Models for multi-step-ahead Epidemic Forecasting?

Position: Quo Vadis, Unsupervised Time Series Anomaly Detection?

M. Saquib Sarfraz[1 2] Mei-Yen Chen[1] Lukas Layer[1] Kunyu Peng[2] Marios Koulakis[2]

PNAS RESEARCH ARTICLE | COMPUTER SCIENCES OPEN ACCESS

Implicit data crimes: Machine learning bias arising from misuse of public data

Efrat Shimron[a,1], Jonathan I. Tamir[b,c,d], Ke Wang[a], and Michael Lustig[a]

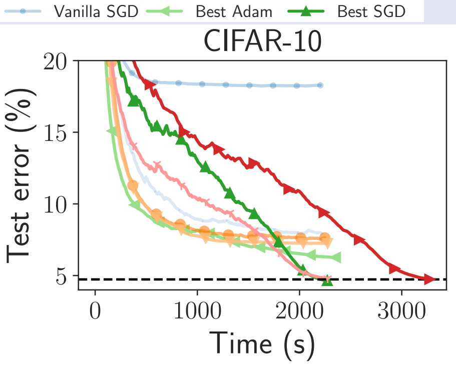
Descending through a Crowded Valley — Benchmarking Deep Learning Optimizers

Robin M. Schmidt[* 1] Frank Schneider[* 1] Philipp Hennig[1 2]

Unclear improvement!

[Moreau et al. 2022]

- Futile benchmarks
- Lack of proper baselines hinders scientific progress.
- Reproducing benchmarks is hard.



Challenges in AI benchmarking

- Futile benchmarks
- Lack of proper baselines hinders scientific progress.
- Reproducing benchmarks is hard.
- Statistical validity is often missing.

[Christodoulou et al. 2024]

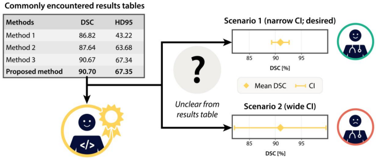
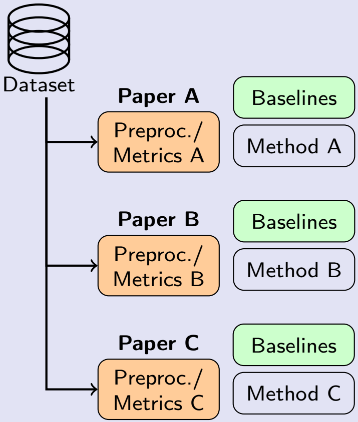


Fig. 1: Common practice in medical imaging algorithm performance reporting leaves many open questions.

Challenges in AI benchmarking

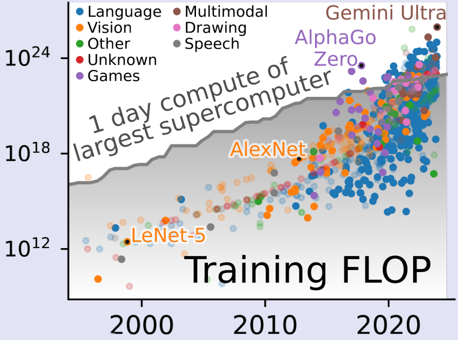
- Futile benchmarks
- Lack of proper baselines hinders scientific progress.
- Reproducing benchmarks is hard.
- Statistical validity is often missing.
- Benchmarking cost is duplicated across groups.



Each paper independently rebuilds preprocessing, baselines, and evaluation $\rightarrow$ duplicated cost.

[Varoquaux et al. 2025]

- Futile benchmarks
- Lack of proper baselines hinders scientific progress.
- Reproducing benchmarks is hard.
- Statistical validity is often missing.
- Benchmarking cost is duplicated across groups.



Reproducible method comparison with Benchopt

. . .

References

- **TM**, Massias, M., Gramfort, A., Ablin, P., Bannier, P.-A., Charlier, B., Dagréou, M., la Tour, T. D., Durif, G., Dantas, C. F., Klopfenstein, Q., Larsson, J., Lai, E., Lefort, T., Malézieux, B., Moufad, B., Nguyen, B. T., Rakotomamonjy, A., Ramzi, Z., Salmon, J., and Vaiter, S. (2022). Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In *NeurIPS*

Making runnable benchmarks with `benchopt`

Benchopt

`benchopt` provides a framework to organize and run benchmarks

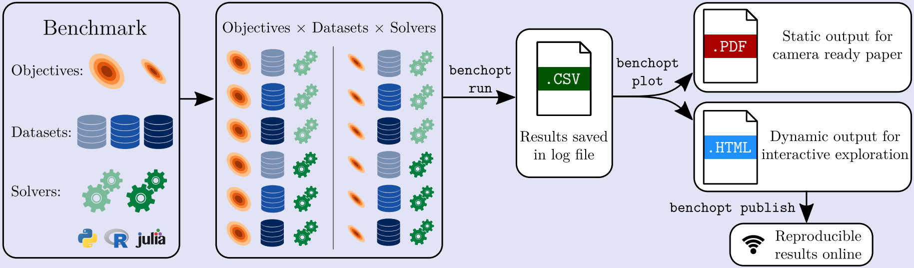
Examples of existing benchmarks:

- **Image Classification (resnet)**
- **Logistic regression**
- **Lasso**
- **ICA**
- **Unsup. Domain Adaptation**
- **Bilevel Optimization**
- **Brain Computer Interface**
- ...

Contributors from...

Components of a benchmark

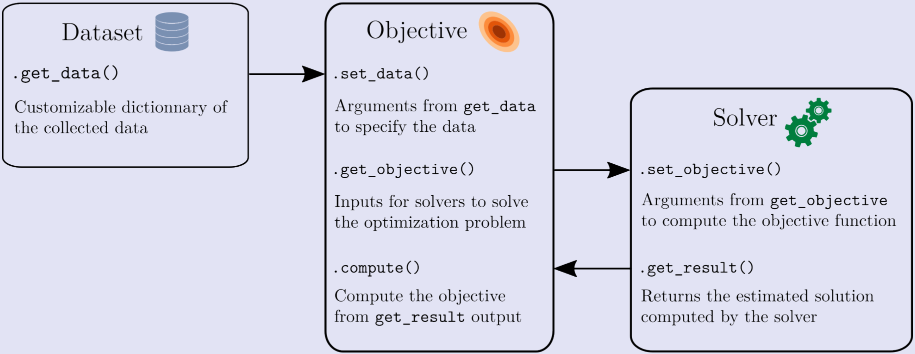
3 components: Objective, Dataset, Solver



A modular framework to create benchmarks

3 components: Objective, Dataset, Solver

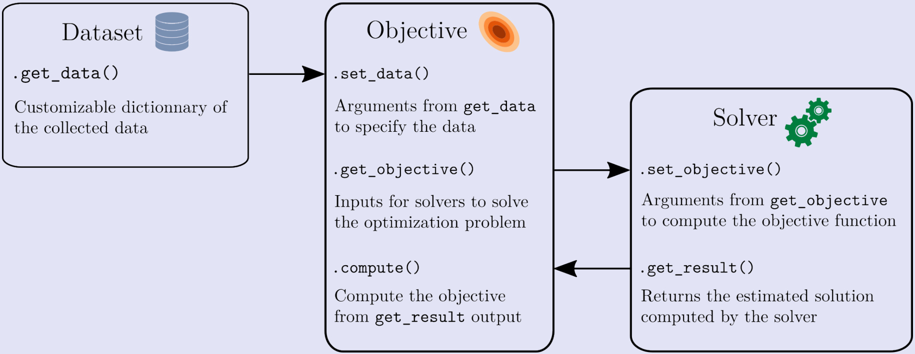
Dependency relation between Dataset - Objective - Solver



A modular framework to create benchmarks

3 components: Objective, Dataset, Solver

Dependency relation between Dataset - Objective - Solver



$\Rightarrow$ Benchopt defines the interface between components.

Explicit requirements and parameters

```python
from benchopt import BaseSolver
from benchmark_utils import grad, init_func


class Solver(BaseSolver):
    name = "GD"
    requirements = ["numpy"]
    sampling_strategy = "callback"
    parameters = {"lr": [1, 1.9], "init": [0, 0.1]}

    def set_objective(self, X):
        self.X = X

    def run(self, cb):
        self.w = init_func(self.X, self.init)
        while cb():
            self.w -= self.lr * grad(self.X, self.w)

    def get_result(self):
        return dict(w=self.w)
```

Eval multiple metrics at once

```python
from benchopt import BaseObjective
from benchmark_utils import split, error


class Objective(BaseObjective):
    name = "Least Squares"
    url = "https://github.com/#ORG/#BENCHMARK_NAME"

    def set_data(self, X):
        self.X_train, self.X_test = split(X)

    def evaluate_result(self, w):
        return dict(
            train_error=error(w, self.X_train),
            test_error=error(w, self.X_test),
        )

    def get_objective(self):
        return dict(X=self.X_train)
```

Explicit preprocessing

```python
from benchopt import BaseDataset
from sklearn.datasets import load_digits


class Dataset(BaseDataset):
    name = "Digits"
    requirements = ["scikit-learn"]

    def get_data(self):
        X = load_digits(return_X_y=True)[0]
        X /= X.std()
        return dict(X=X)
```

Explicit preprocessing

```python
from benchopt import BaseDataset
from sklearn.datasets import load_digits


class Dataset(BaseDataset):
    name = "Digits"
    requirements = ["scikit-learn"]

    def get_data(self):
        X = load_digits(return_X_y=True)[0]
        X /= X.std()
        return dict(X=X)
```

$\Rightarrow$ Reproducible benchmark by design!

Goal: if you use the same setup, you don't need to re-run the baseline methods!

Structure of a benchmark

```
benchmark/
├── objective.py
├── datasets/
│   ├── dataset1.py
│   └── dataset2.py
└── solvers/
    ├── solver1.py
    └── solver2.py
```

Modular & extendable

- New metric? modify objective

Structure of a benchmark

```
benchmark/
├── objective.py
├── datasets/
│   ├── dataset1.py
│   └── dataset2.py
└── solvers/
    ├── solver1.py
    └── solver2.py
```

Modular & extendable

- New metric? modify objective
- New dataset? add a file

```
benchmark/
├── objective.py
├── datasets/
│   ├── dataset1.py
│   └── dataset2.py
└── solvers/
    ├── solver1.py
    └── solver2.py
```

Modular & extendable

- New metric? modify objective
- New dataset? add a file
- New solver? add a file

Structure of a benchmark

```
benchmark/
├── objective.py
├── datasets/
│   ├── dataset1.py
│   └── dataset2.py
└── solvers/
    ├── solver1.py
    └── solver2.py
```

Modular & extendable

- New metric? modify objective
- New dataset? add a file
- New solver? add a file

Structure of a benchmark

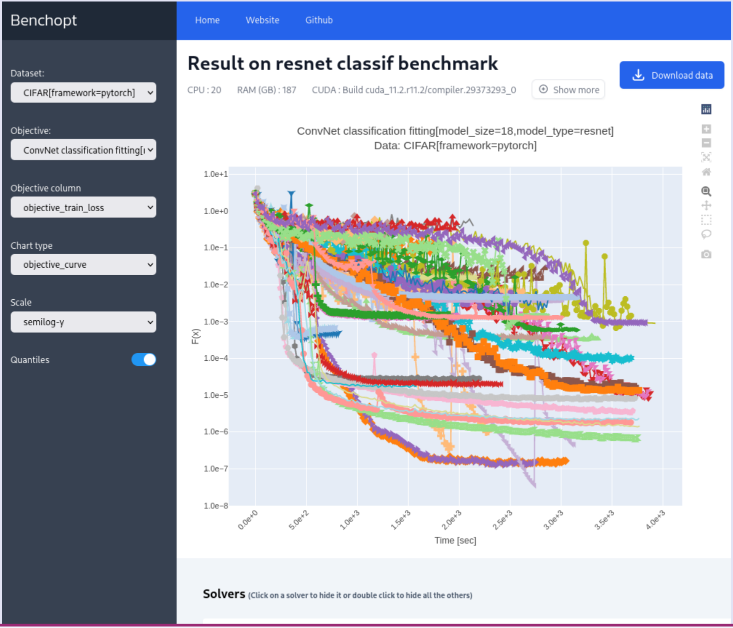
```
benchmark/
├── objective.py
├── datasets/
│   ├── dataset1.py
│   └── dataset2.py
└── solvers/
    ├── solver1.py
    └── solver2.py
```

Modular & extendable

- New metric? modify objective
- New dataset? add a file
- New solver? add a file

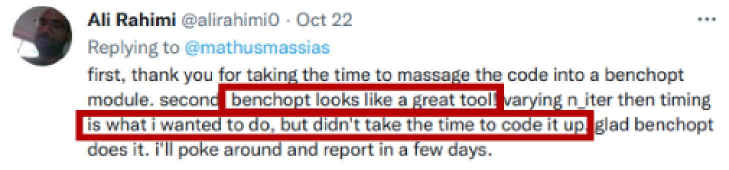
Template to create a new benchmark:
`https://github.com/benchopt/template_benchmark`

Interactive results exploration



Benchopt makes your life easy

- Build on previous benchmarks
- Use solvers in Python, R, Julia, binaries...
- Monitor any metric you want altogether (test/train loss, ...)
- Add parameters to solvers and use config files
- Share and publish HTML results
- Run all benchmarks in parallel, on HPC clusters...
- Cache runs and results
- and much more!



Typical case: deep learning optimization

A. List of optimizers and schedules considered

Table 2: List of optimizers considered for our benchmark. This is only a subset of all existing methods for deep learning.

Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KF-QN-CNN	(Ren & Goldfarb, 2021)
AdaBatch	(Devarakonda et al., 2017)	KFAC	(Martens & Grosse, 2015)
AdaBayes/AdaBayes-SS	(Aitchison, 2020)	KFLR/KFRA	(Botev et al., 2017)
AdaBelief	(Zhuang et al., 2020)	L4Adam/L4Momentum	(Rolínek & Martius, 2018)
AdaBlock	(Yun et al., 2019)	LAMB	(You et al., 2020)
AdaBound	(Luo et al., 2019)	LaProp	(Ziyin et al., 2020)
AdaComp	(Chen et al., 2018)	LARS	(You et al., 2017)
Adadelta	(Zeiler, 2012)	LHOPT	(Almeida et al., 2021)
Adafactor	(Shazeer & Stern, 2018)	LookAhead	(Zhang et al., 2019)
AdaFix	(Bae et al., 2019)	M-SVAG	(Balles & Hennig, 2018)
AdaFom	(Chen et al., 2019a)	MADGRAD	(Defazio & Jelassi, 2021)
AdaFTRL	(Orabona & Pál, 2015)	MAS	(Landro et al., 2020)
Adagrad	(Duchi et al., 2011)	MEKA	(Chen et al., 2020b)
ADAHESSIAN	(Yao et al., 2020)	MTAdam	(Malkiel & Wolf, 2020)
Adai	(Xie et al., 2020)	MVRC-1/MVRC-2	(Chen & Zhou, 2020)
AdaLoss	(Teixeira et al., 2019)	Nadam	(Dozat, 2016)
Adam	(Kingma & Ba, 2015)	NAMSB/NAMSG	(Chen et al., 2019b)
Adam+	(Liu et al., 2020b)	ND-Adam	(Zhang et al., 2017a)
AdamAL	(Tao et al., 2019)	Nero	(Liu et al., 2021b)
AdaMax	(Kingma & Ba, 2015)	Nesterov	(Nesterov, 1983)
AdamBS	(Liu et al., 2020c)	Noisy Adam/Noisy K-FAC	(Zhang et al., 2018)
AdamNC	(Reddi et al., 2018)	NosAdam	(Huang et al., 2019)
AdaMod	(Ding et al., 2019)	Novograd	(Ginsburg et al., 2019)
AdamP/SGDP	(Heo et al., 2021)	NT-SGD	(Zhou et al., 2021b)
AdamT	(Zhou et al., 2020)	Padam	(Chen et al., 2020a)
AdamW	(Loshchilov & Hutter, 2019)	PAGE	(Li et al., 2020b)
AdamX	(Tran & Phong, 2019)	PAL	(Mutschler & Zell, 2020)
ADAS	(Eliyahu, 2020)	PolyAdam	(Orvieto et al., 2019)
AdaS	(Hosseini & Plataniotis, 2020)	Polyak	(Polyak, 1964)
AdaScale	(Johnson et al., 2020)	PowerSGD/PowerSGDM	(Vogels et al., 2019)
AdaSGD	(Wang & Wiens, 2020)	Probabilistic Polyak	(de Roos et al., 2021)
AdaShift	(Zhou et al., 2019)	ProbLS	(Mahsereci & Hennig, 2017)
AdaSqrt	(Hu et al., 2019)	PStorm	(Xu, 2020)
Adathm	(Sun et al., 2019)	QHAdam/QHM	(Ma & Yarats, 2019)
AdaX/AdaX-W	(Li et al., 2020a)	RAdam	(Liu et al., 2020a)
AEGD	(Liu & Tian, 2020)	Ranger	(Wright, 2020b)
ALI-G	(Berrada et al., 2020)	RangerLars	(Grankin, 2020)
AMSBound	(Luo et al., 2019)	RMSProp	(Tieleman & Hinton, 2012)
AMSGrad	(Reddi et al., 2018)	RMSterov	(Choi et al., 2019)
AngularGrad	(Roy et al., 2021)	S-SGD	(Sung et al., 2020)
ArmijoLS	(Vaswani et al., 2019)	SAdam	(Wang et al., 2020b)
ARSG	(Chen et al., 2019b)	Sadam/SAMSGrad	(Tong et al., 2019)
ASAM	(Kwon et al., 2021)	SALR	(Yue et al., 2020)
AutoLRS	(Jin et al., 2021)	SAM	(Foret et al., 2021)
AvaGrad	(Savarese et al., 2019)	SC-Adagrad/SC-RMSProp	(Mukkamala & Hein, 2017)
BAdam	(Salas et al., 2018)	SDProp	(Ida et al., 2017)
BGAdam	(Bai & Zhang, 2019)	SGD	(Robbins & Monro, 1951)
BPGrad	(Zhang et al., 2017b)	SGD-BB	(Tan et al., 2016)
BRMSProp	(Aitchison, 2020)	SGD-G2	(Ayadi & Turinici, 2020)
BSGD	(Hu et al., 2020)	SGDEM	(Ramezani-Kebrya et al., 2021)
C-ADAM	(Tutunov et al., 2020)	SGDHess	(Tran & Cutkosky, 2021)
CADA	(Chen et al., 2021)	SGDM	(Liu & Luo, 2020)
Cool Momentum	(Borysenko & Byshkin, 2020)	SGDR	(Loshchilov & Hutter, 2017)
CProp	(Preechakul & Kijsirikul, 2019)	SHAdagrad	(Huang et al., 2020)
Curveball	(Henriques et al., 2019)	Shampoo	(Anil et al., 2020; Gupta et al., 2018)
Dadam	(Nazari et al., 2019)	SignAdam++	(Wang et al., 2019a)
DeepMemory	(Wright, 2020a)	SignSGD	(Bernstein et al., 2018)
DGNOpt	(Liu et al., 2021a)	SKQN/S4QN	(Yang et al., 2020)
DiffGrad	(Dubey et al., 2020)	SM3	(Anil et al., 2019)
EAdam	(Yuan & Gao, 2020)	SMG	(Tran et al., 2020)

Frank Schneider
@frankstefansch1

📣 #ICML 2021 Paper 📣
Overwhelmed by the flood of optimizers for deep learning? We felt the same and performed an extensive benchmark. Joint work with @robinschmidt_ & @PhilippHennig5.
Paper: arxiv.org/abs/2007.01547
Results: github.com/SirRob1997/Cro…
Video: youtu.be/cz9RzlstFdE

Frank Schneider
@frankstefansch1

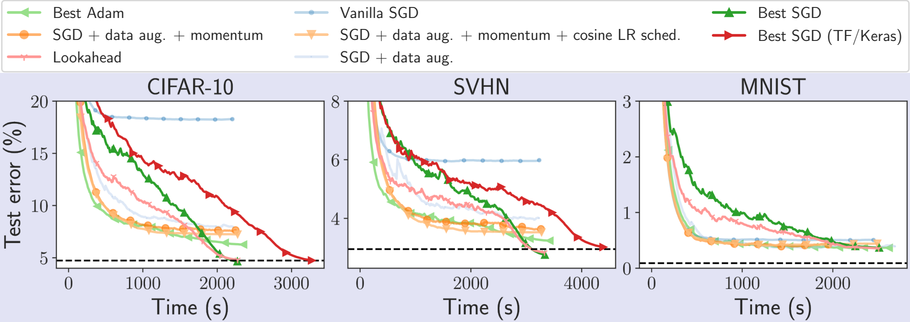
Our results? There is no winner consistently outperforming the competition. Instead, Adam remains a strong contender for many problems.
In some cases, just trying out a few optimizers with their default hyperparameters can work as well as tuning one specific method.

$\Rightarrow$ Many novel methods but unclear improvements.

$\Rightarrow$ But this benchmark cannot be easily reproduced!

Example: Optimization for ResNet on image classication

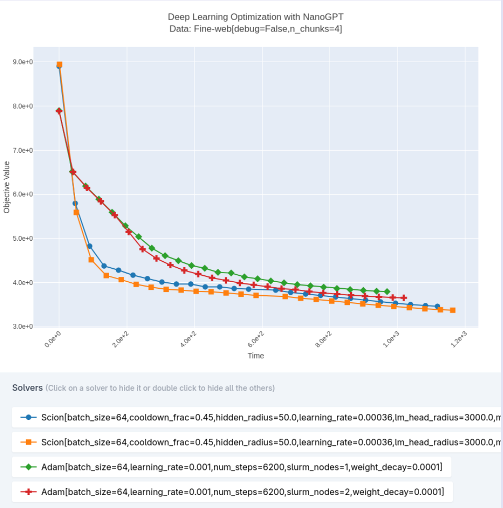
- Image classification with resnet18
- Various optimization strategies
- Compare `pytorch` and `tensorflow`
- Publish reproducible SOTA for baselines



https://github.com/benchopt/benchmark_resnet_classif/

Example: Large scale-optimization for Deep learning (WIP)

- Use modern large-scale datasets and models
- Classical optimization tricks
- Distributed training and mixed precision



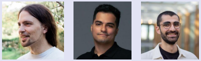
`https://github.com/tommoral/benchmark_nanogpt/`

Develop reference benchmarks

skada-bench Public
generated from benchopt/template_benchmark

. . .

benchmark_bci Public
generated from benchopt/template_benchmark



. . .

benchmark_tsad Public
generated from benchopt/template_benchmark

. . .

Improve the benchmarking methodology

How to ensure that a method is better than another?

Investigating the ranking confidence depending on the number of samples and the CV strategy.

Reproducibility in AI

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility

- Provide tools for other to use on their use case?

  Reusability

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility

- Provide tools for other to use on their use case?

  Reusability

- Run with new parameters to evaluate robust results?

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility
- Provide tools for other to use on their use case?

  Reusability
- Run with new parameters to evaluate robust results?
- Run with a new dataset to extend the validity?

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility
- Provide tools for other to use on their use case?

  Reusability
- Run with new parameters to evaluate robust results?
- Run with a new dataset to extend the validity?
- Extend the results with a new method, to show improvement?

  Replicability | External reproducibility

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility
- Provide tools for other to use on their use case?

  Reusability
- Run with new parameters to evaluate robust results?
- Run with a new dataset to extend the validity?
- Extend the results with a new method, to show improvement?

  Replicability | External reproducibility

Reproducible research

Different goals:

- Reproduce the exact same results?

  Repeatability | Bitwise reproducibility

- Provide tools for other to use on their use case?

  Reusability

- Run with new parameters to evaluate robust results?
- Run with a new dataset to extend the validity?
- Extend the results with a new method, to show improvement?

  Replicability | External reproducibility

The last point becomes complicated with AI

Reproducible research in AI

Technical challenges for repeatability:

- Multiple frameworks: `PyTorch`, `Tensorflow`, `jax`,...
- Hardware dependence: CPU, GPU, TPUs,...
- Large scale datasets: hard to share, pre-process,...

Technical challenges for repeatability:

- Multiple frameworks: `PyTorch`, `Tensorflow`, `jax`,...
- Hardware dependence: CPU, GPU, TPUs,...
- Large scale datasets: hard to share, pre-process,...

Challenges for reuse:

- Usability: API, modularity, documentation, tests,...
- Maintenance: dependencies, code rot, long term support, ...

Reproducible research in AI

Technical challenges for repeatability:

- Multiple frameworks: `PyTorch`, `Tensorflow`, `jax`,...
- Hardware dependence: CPU, GPU, TPUs,...
- Large scale datasets: hard to share, pre-process,...

Challenges for reuse:

- Usability: API, modularity, documentation, tests,...
- Maintenance: dependencies, code rot, long term support, ...

Statistical challenges for replicability:

- Stochastic algorithms: random initialization, data shuffling,...
- Data handling: splitting, data preprocessing, model selection,...

Good practices to share code

Minimal requirement for a research project: (in `Python`)

- Write clean code:
  - Consistent style: *Use `black` or `flake8`.*
  - Use readable names.
  - No notebooks!
- Document your code: *docstring and `README.md`. optinaly `sphinx`.*
- Determinist output: *Control the random seeds.*

Optional but advised

- Document dependencies.
- Proper package organization
- Add some tests: *Use `pytest`.*

Going further: creating a package

If you really want to make your research *reproductible* by other in different contexts, you need to properly package it.

- Documentation: *Sphinx*.
- Test on multiple platforms: *Continuous Integration*.
- Release on pypi/conda-forge
- Talk about it ! :)

Example of package: `https://github.com/tomMoral/test_package`

An overlooked aspect: longer term maintenance

Many research results are not maintained after publication:

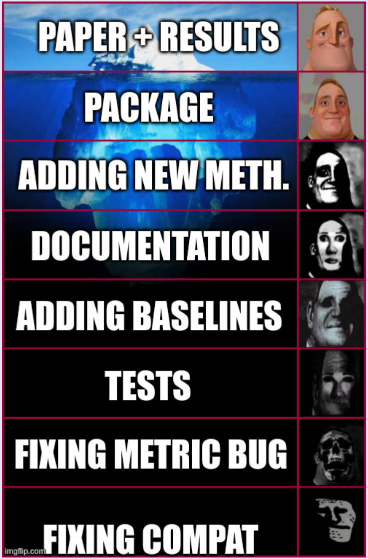
- Every PhD creates a package;
- Every post-doc abandons one;
- The ecosystem grows horizontally, not vertically

An overlooked aspect: longer term maintenance

Many research results are not maintained after publication:

- Every PhD creates a package;
- Every post-doc abandons one;
- The ecosystem grows horizontally, not vertically

$\Rightarrow$ Limited incentives in AI to maintain a codebase



Crossing the validation crisis in AI benchmarking: the challenge of statistical validity

Evaluating decision functions & learning algorithms

In AI, we produce decision function g, that can be evaluated with:

$$S_{\mathcal{D}}(g) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\sim\mathcal{D}} m(g(X), y)$$

The oracle score is:

$$S^*(g) = \mathbb{E}_{\mathcal{D}\sim d}\left[S_{\mathcal{D}}(g)\right]$$

Evaluating decision functions & learning algorithms

In AI, we produce decision function g, that can be evaluated with:

$$S_{\mathcal{D}}(g) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\sim\mathcal{D}} m(g(X), y)$$

The oracle score is:

$$S^*(g) = \mathbb{E}_{\mathcal{D}\sim d}\left[S_{\mathcal{D}}(g)\right]$$

For learning algorithms F_λ producing $g = F_\lambda(\mathcal{D}_{train}, \xi)$, the oracle score becomes:

$$\begin{aligned}\mathcal{S}^*(F_\lambda) =& \mathbb{E}_{\mathcal{D}_{train}\sim d}\mathbb{E}_{\xi}[S^*(F_\lambda(\mathcal{D}_{train}, \xi))]\\ =& \mathbb{E}_{\mathcal{D}_{test}\sim d}\mathbb{E}_{\mathcal{D}_{train}\sim d}\mathbb{E}_{\xi}[S_{\mathcal{D}_{test}}(F_\lambda(\mathcal{D}_{train}, \xi))]\end{aligned}$$

Benchmarking goal: compare and rank methods

Given two learning algorithms, we want to compare them according to their oracle score:

$$\mathcal{S}^*(F_\lambda) \overset{?}{>} \mathcal{S}^*(F_{\lambda'})$$

Benchmarking goal: compare and rank methods

Given two learning algorithms, we want to compare them according to their oracle score:

$$\mathcal{S}^*(F_\lambda) \overset{?}{>} \mathcal{S}^*(F_{\lambda'})$$

In practice, we only have access to empirical estimates of the oracle score!

$\Rightarrow$ How to ensure that our benchmark results reflect the true oracle ranking?

DE L' Importance DE la validation croisée

OU DU RAPPORT QUE LES LOIX DOIVENT AVOIR AVEC LA CONSTITUTION DE CHAQUE GOUVERNEMENT, LES MOEURS, LE CLIMAT, LA RELIGION, LE COMMERCE, &c.

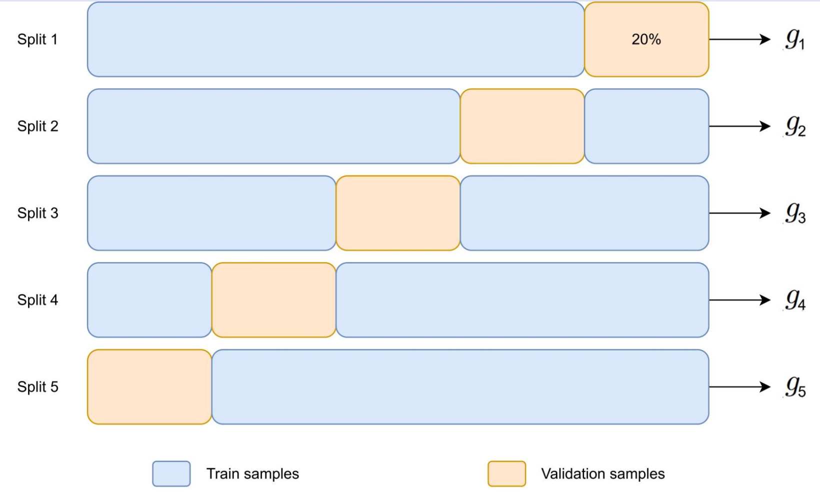
à quoi l'Auteur a ajouté

Des recherches nouvelles ſur les Loix Romaines touchant les Succeſſions, ſur les Loix Françoiſes, & ſur les Loix Féodales.

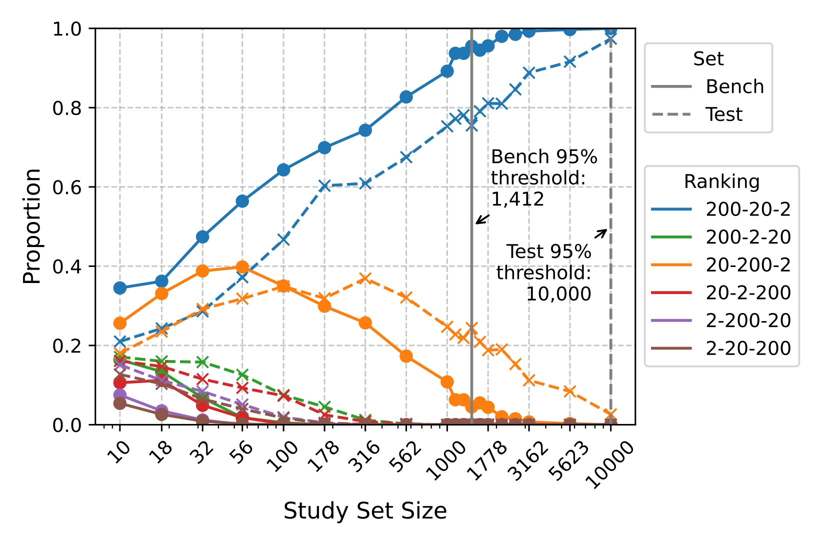
TOME PREMIER.

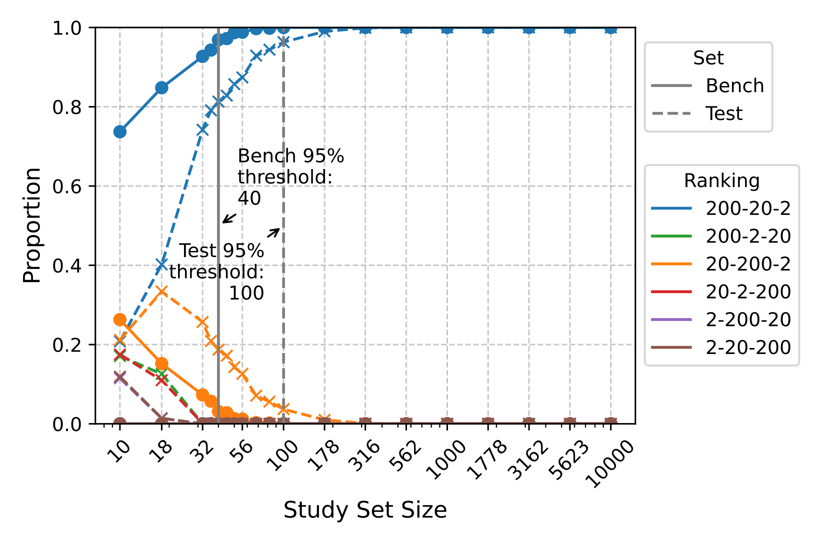
A Grenoble,
Chez BARRILLOT & FILS.

The cross-validation setting



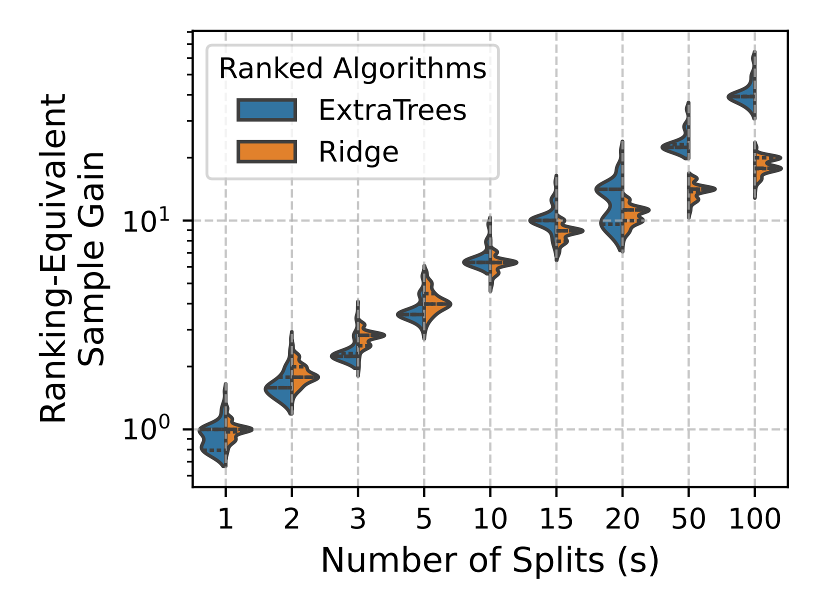
The benefit of CV over a single train/test split





Proportion
1.0
0.8
0.6
0.4
0.2
0.0
10
18
32
56
100
178
316
562
1000
1778
3162
5623
10000
Study Set Size
Bench 95% threshold: 40
Test 95% threshold: 100
Set
Bench
Test
Ranking
200-20-2
200-2-20
20-200-2
20-2-200
2-200-20
2-20-200

Sample gain of CV over a single split



Reproducibility in AI: statistical reproducibility

Stochasticity is intrinsic to AI applications.

$\Rightarrow$ How to ensure that benchmark results are statistically reproducible?

Stochasticity is intrinsic to AI applications.

$\Rightarrow$ How to ensure that benchmark results are statistically reproducible?

One answer is to use proper statistical tools to assess significance of the results.

As this is costly (need many repetitions or large datasets), we need to have a way to easily share and reuse past results!

Conclusion

Reproducible research needs more than just releasing code:

- Reusable $\rightarrow$ Clean and Documented.
- Extendable $\rightarrow$ Proper packaging and maintenance.
- Statistically valid $\rightarrow$ Proper evaluation protocols.

Conclusion

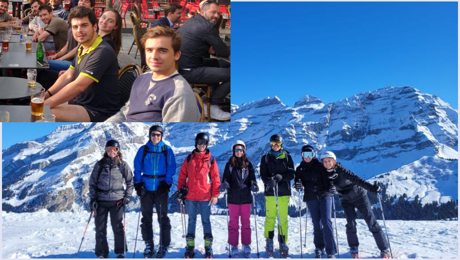
Reproducible research needs more than just releasing code:

- Reusable $\rightarrow$ Clean and Documented.
- Extendable $\rightarrow$ Proper packaging and maintenance.
- Statistically valid $\rightarrow$ Proper evaluation protocols.

Need proper tools to make it possible!

Also need to build a community around these tools, to share the maintenance and the fun!

Thank you for your attention!



Don't hesitate to star the benchopt repo!