EngiBench: A Framework for Data-Driven Engineering Design Research

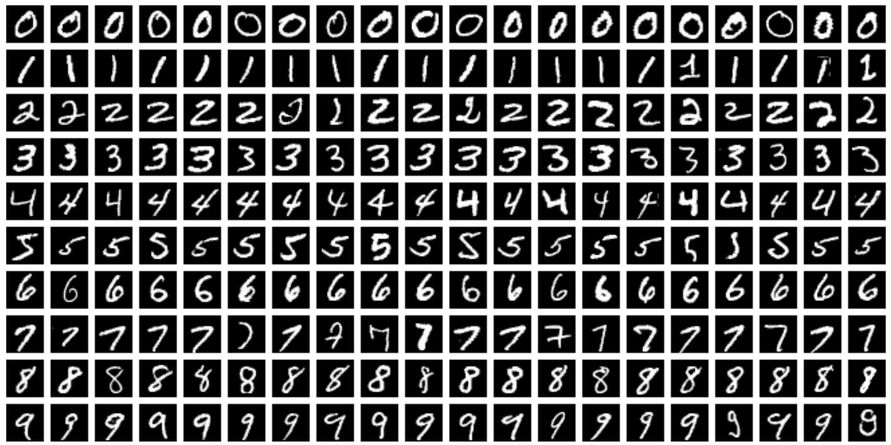
Florian Felten, Gabriel Apaza, Gerhard Bräunlich, Cashen Diniz, Xuliang Dong, Arthur Drake, Milad Habibi, Nathaniel Hoffman, Matthew Keeler, Soheyl Massoudi, Francis VanGessel, Mark Fuge

ETH zürich

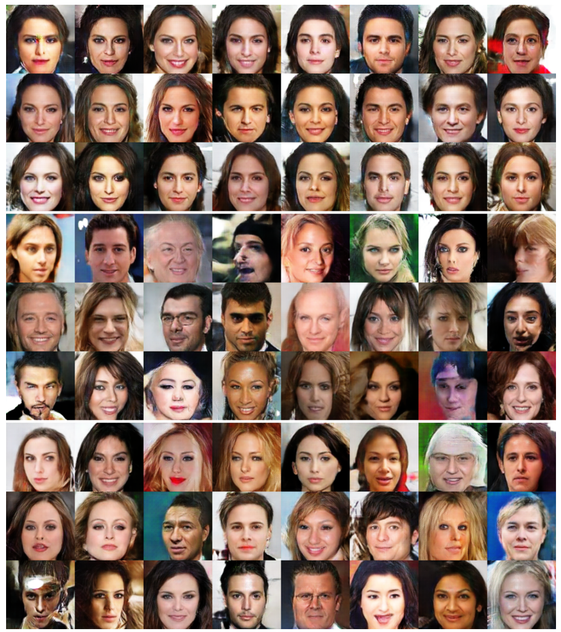
IDEAL

UNIVERSITY OF MARYLAND

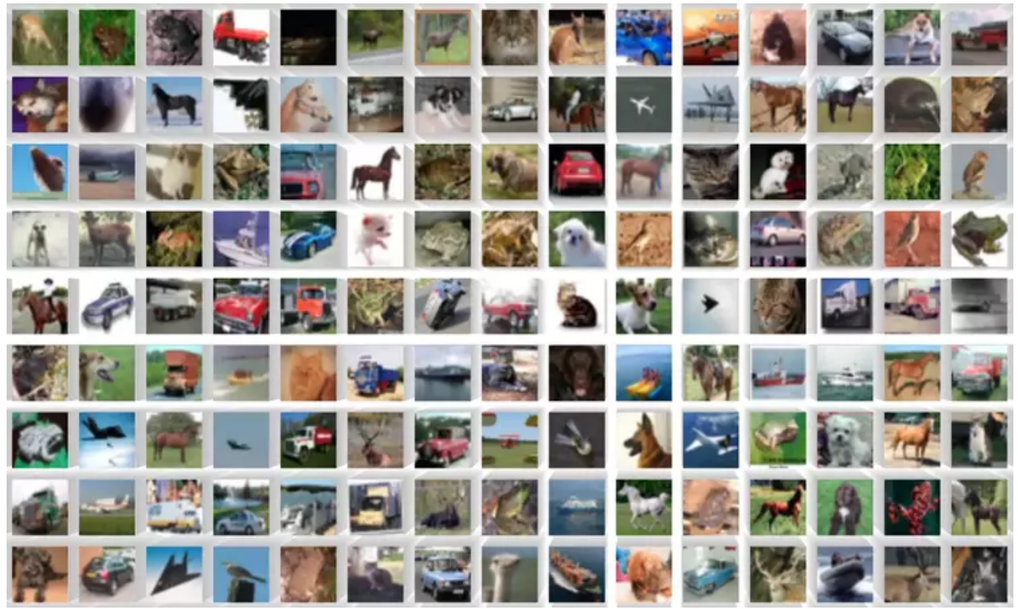
Benchmarks for Generative Models



MNIST



CelebA

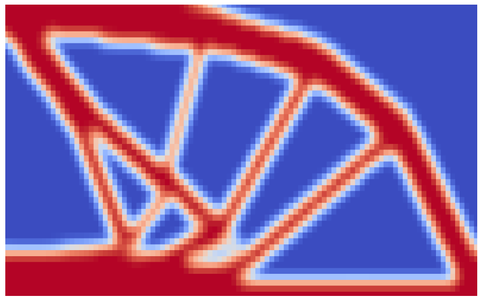


CIFAR10

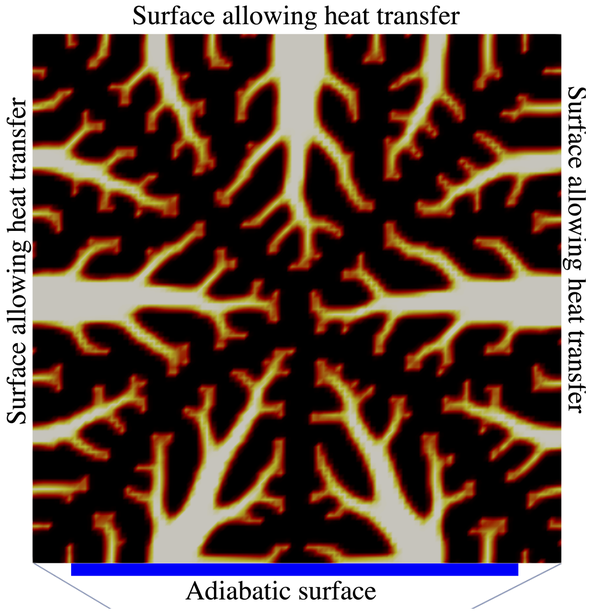
Saturated, right?

What about Engineering ?

Beam

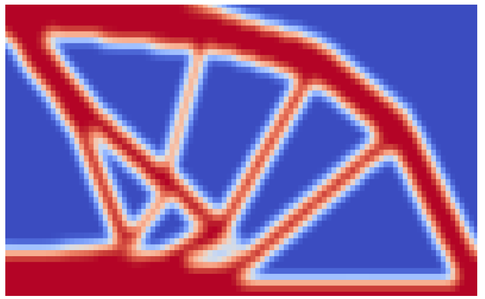


Heat conduction

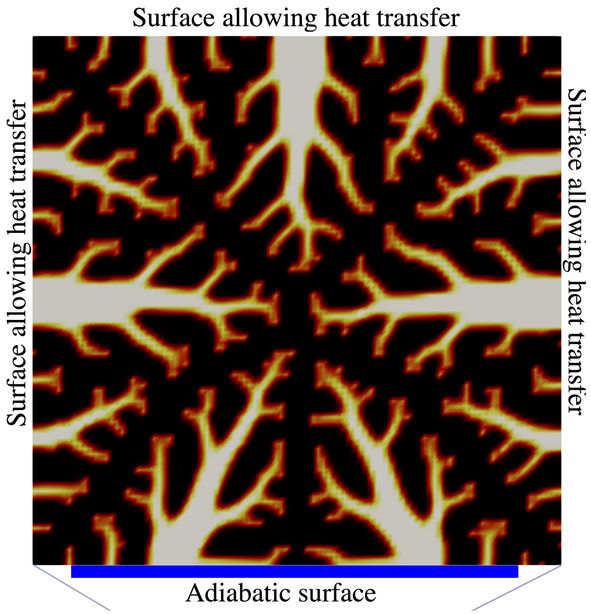


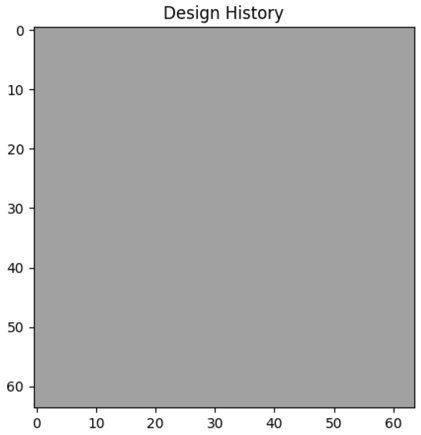
What about Engineering ?

Beam



Heat conduction

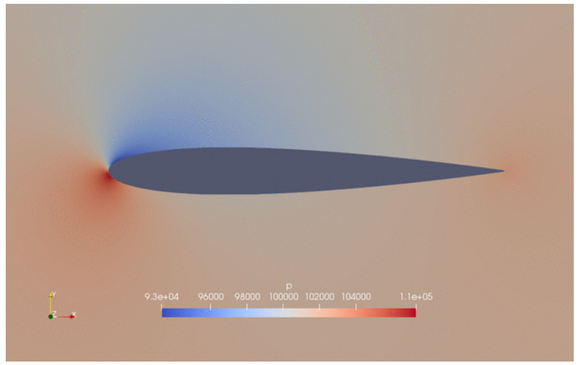




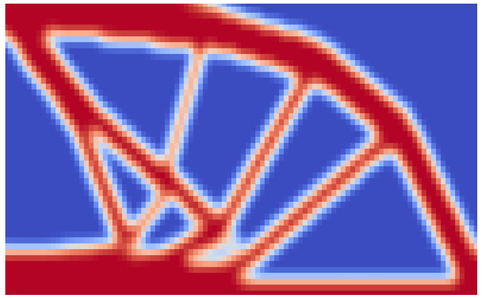
Thermo-elastic Beam

What about Engineering ?

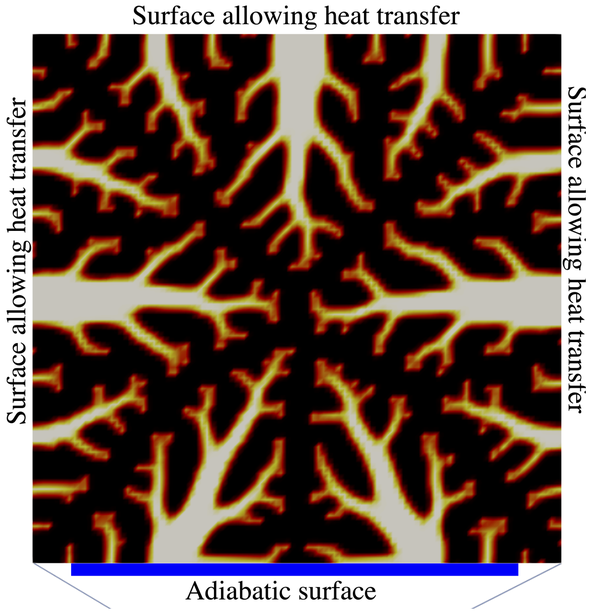
Airfoil

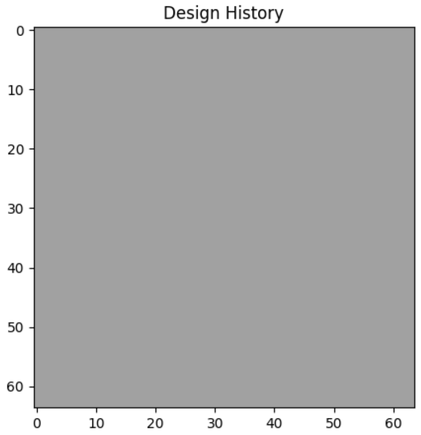


Beam



Heat conduction

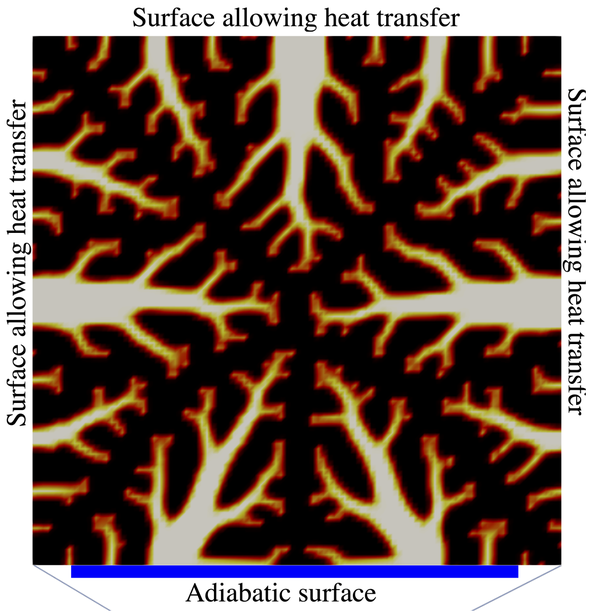




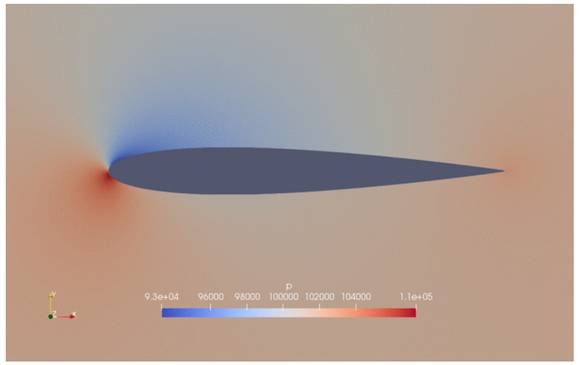
Thermo-elastic Beam

What about Engineering ?

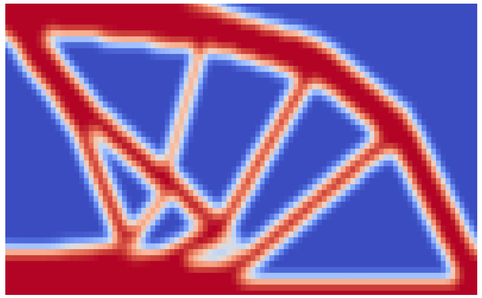
Heat conduction

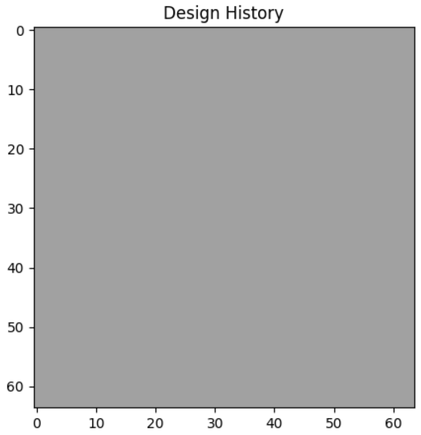


Airfoil



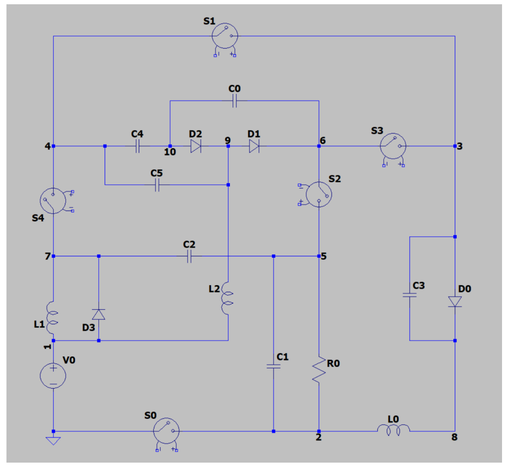
Beam



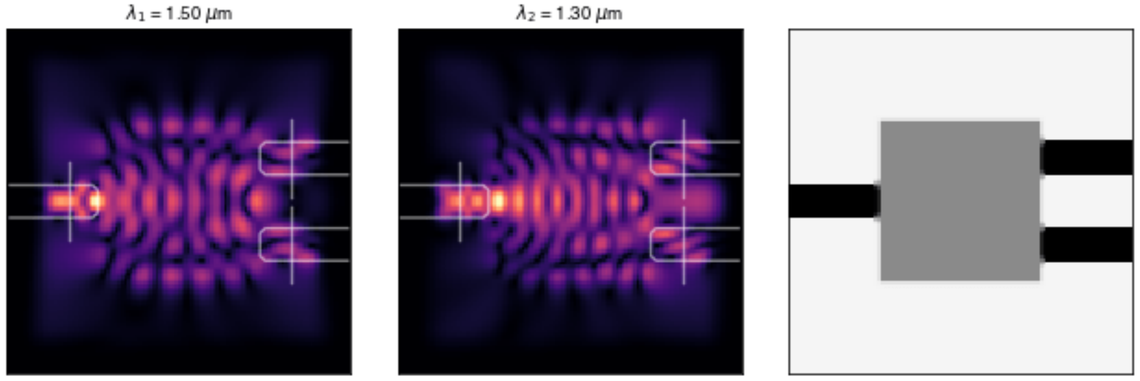


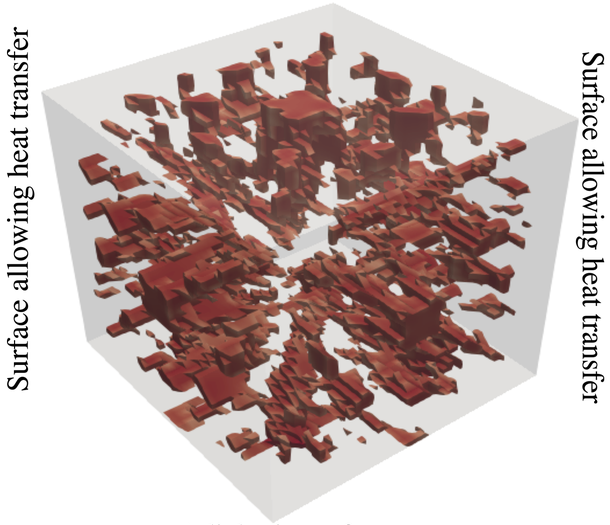
Thermo-elastic Beam

Power electronics

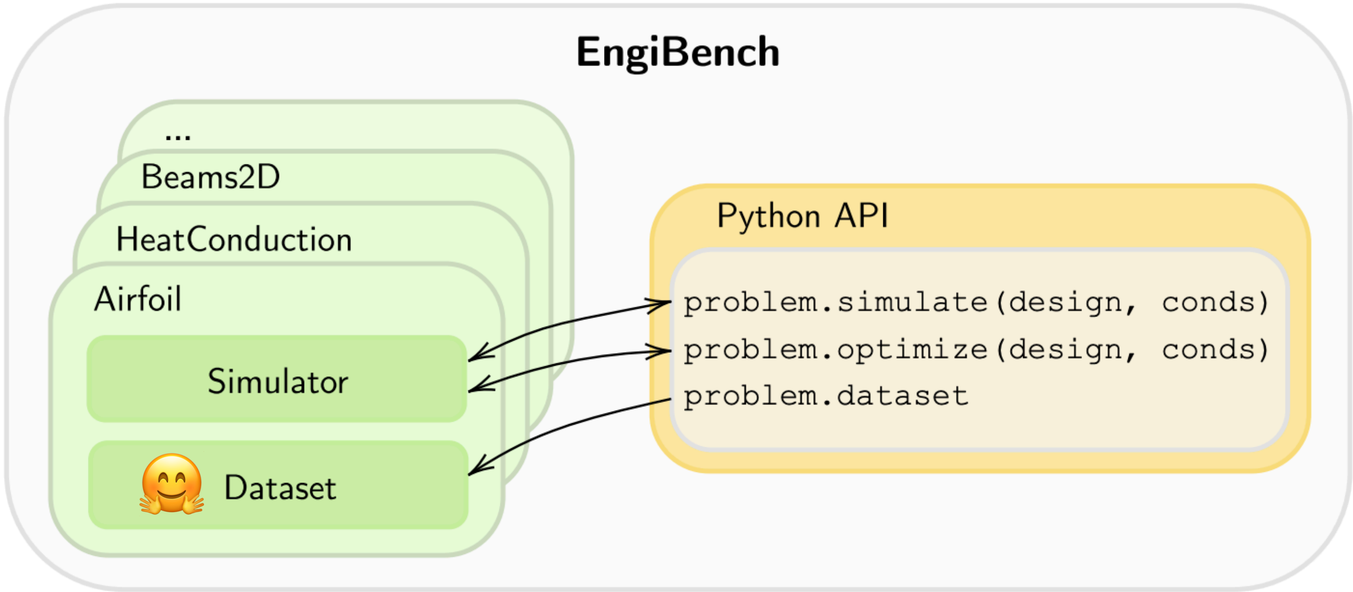


Photonics





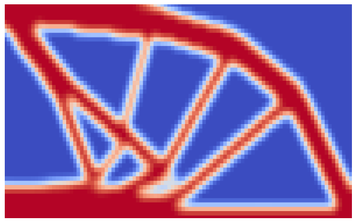
Contribution: EngiBench



A **set of engineering design problems under a unified Python API**.

Problem = Dataset + Simulator

~Gym + Datasets for Engineering Design



An intuitive API

```python
from engibench.problems.beams2d.v0 import Beams2D

problem = Beams2D()
problem.reset(seed=42)


problem.design_space # Box(0.0, 1.0, (50, 100), float64)
problem.objectives # (("compliance", "MINIMIZE"),)
problem.conditions # (("volfrac", 0.35), ("forcedist", 0.0),…)


problem.dataset
inverse_model = train_inverse(problem.dataset)

desired_conds = {"volfrac": 0.7, "forcedist": 0.3}
generated_design = inverse_model.predict(desired_conds)

objs = problem.simulate(generated_design, desired_conds)
opt_design, history = problem.optimize(generated_design, desired_conds)
problem.render(opt_design)
```

Versioned problems to explicitly state breaking changes when updating

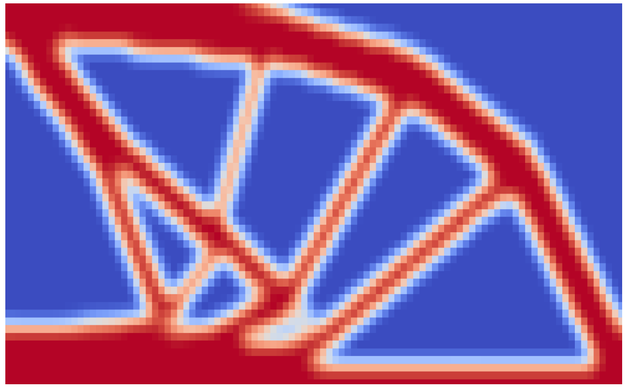
Automated metadata extraction, e.g, to size your neural networks, or to get the dataset columns

Direct access to the dataset

Call the simulator

Want to try another problem?

```
from engibench.problems.airfoil.v0 import Airfoil

problem = Airfoil()
problem.reset(seed=42)


problem.design_space # Dict("angle_of_attack": Box(0.0, 10.0), "coords":…)
problem.objectives # (("drag_coefficient", "MINIMIZE"))
problem.conditions #  (("mach", 0.8), ("reynolds",1e6),      …))


problem.dataset
inverse_model = train_inverse(problem.dataset)



desired_conds = {"mach": 0.45, "reynolds": 4e6}
generated_design = inverse_model.predict(desired_conds)
random_design, _ = problem.random_design()

objs = problem.simulate(random_design, desired_conds)
opt_design, history = problem.optimize(random_design, desired_conds)
problem.render(opt_design)
```



Take home: EngiBench and EngiOpt

✅ One Python API, several engineering design problems.

✅ Highly reproducible, open source and
open to contributions!

✅ We also have algorithms implementations as baselines:
GAN, Diffusion, VAE, Surrogate models.
Can you beat them?

🤗 Come see us on Wednesday at Poster Session 1: 11AM

⏭️ What's preventing you from trying it?
`pip install engibench`

Paper

Website