How do you unit test an ML model?

Wahab Kawafi

Best Practices in AI Afternoon

University of BRISTOL

The University Of Sheffield.

Contents

- Introduction & Motivation
- Unit testing during training
- Unit testing during deployment
    1. Dataset design
        - Limited
        - Unlimited
    2. IO
        - Input sensitivity
        - Output uncertainty
- Conclusion

Background

- BSc Biomedical Science
- MSc Space Physiology
- PhD Computational Biology & Machine Learning
- ML Engineer - CFMS
- AI Infrastructure Engineer - Isambard-AI

The University Of Sheffield.

KING'S College LONDON

cfms

University of BRISTOL

✈️ https://www.youtube.com/watch?v=WRf395ioJRY

ML **Engineering**

- AI is very popular at the moment.
- There is a huge focus on novelty in publication. There is likely already a model out there that does what you want to do!
- The simpler the better.
- Application and implementation!

- Two hats:
  1. ML is code! 🧢
     "Just write Python unit tests!"
  2. ML is more! 🎩

Inspiration: ISO-2768

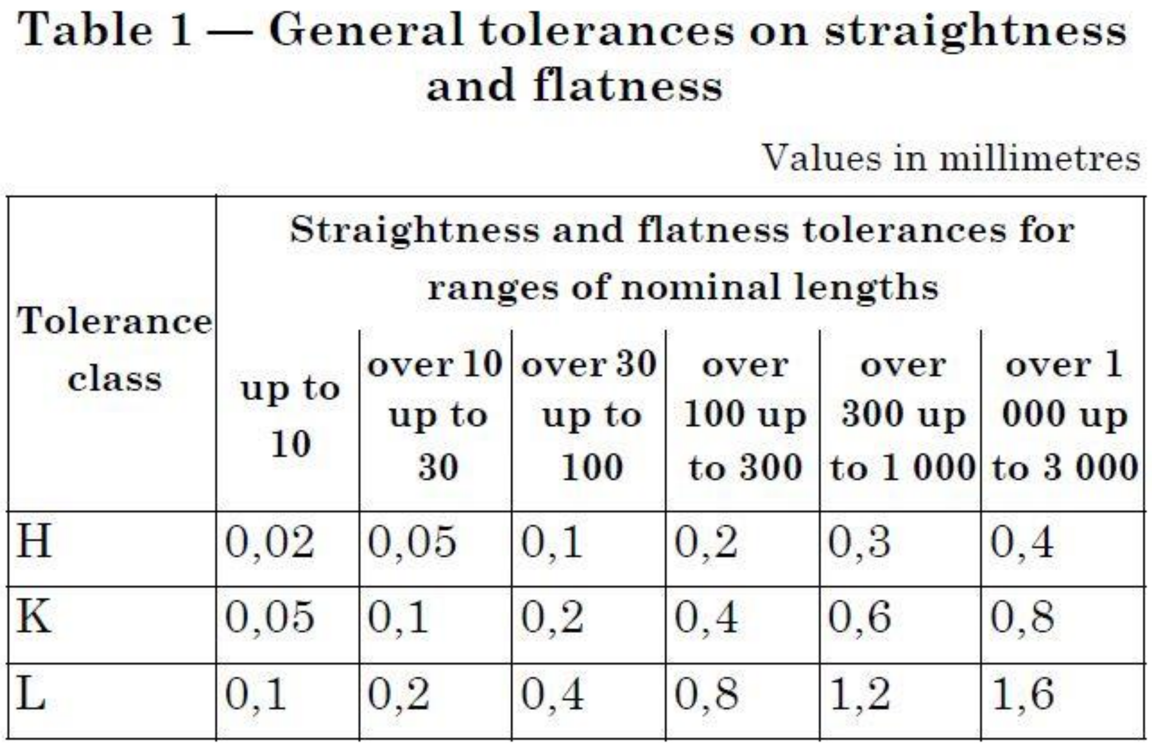
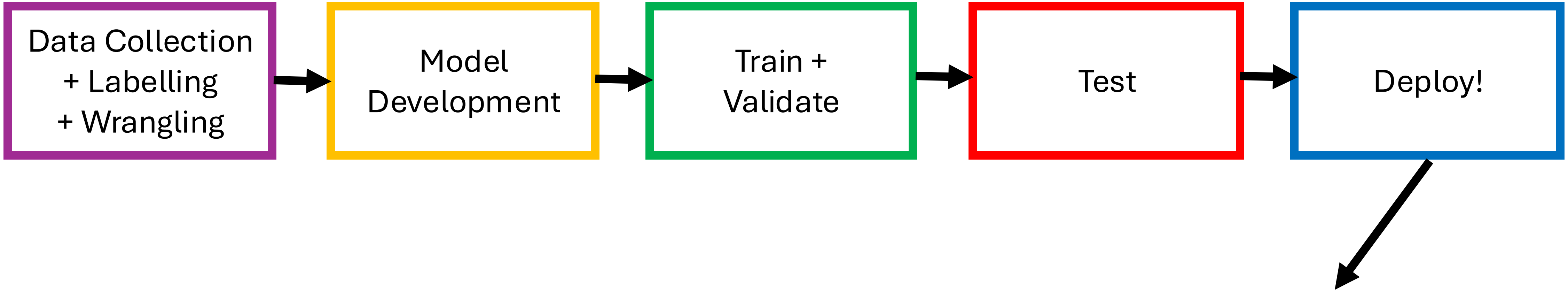


Table 1 — General tolerances on straightness and flatness

Values in millimetres

Tolerance class	Straightness and flatness tolerances for ranges of nominal lengths					
	up to 10	over 10 up to 30	over 30 up to 100	over 100 up to 300	over 300 up to 1 000	over 1 000 up to 3 000
H	0,02	0,05	0,1	0,2	0,3	0,4
K	0,05	0,1	0,2	0,4	0,6	0,8
L	0,1	0,2	0,4	0,8	1,2	1,6

MLOps Lifecycle



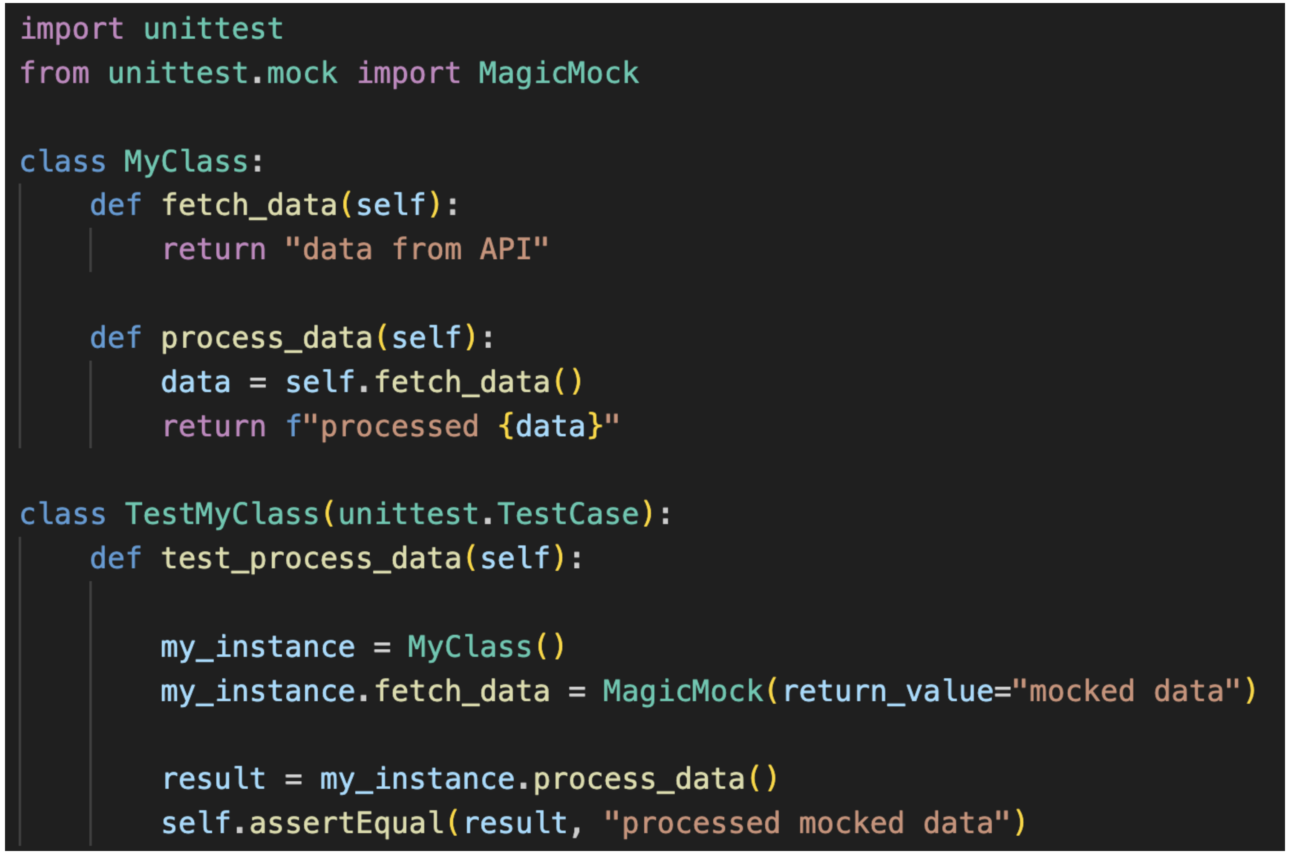
Data Collection + Labelling + Wrangling → Model Development → Train + Validate → Test → Deploy!

What happens if my model does something wrong?

😨

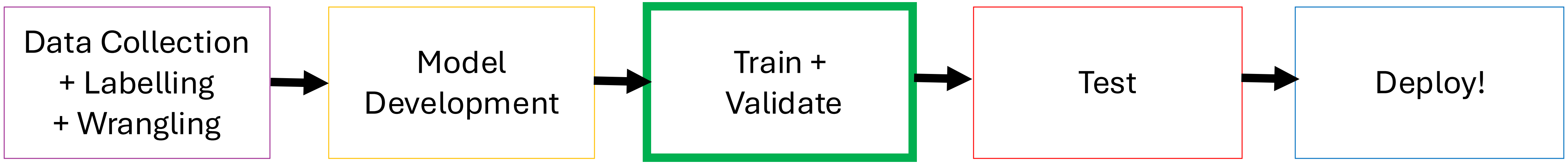
"ML is just code! 🎒 "

- Mock Testing
- What if my model works off of a Webcam/API?
- Do I need to upload my entire dataset to Github to run my unit tests?



```python
import unittest
from unittest.mock import MagicMock

class MyClass:
    def fetch_data(self):
        return "data from API"

    def process_data(self):
        data = self.fetch_data()
        return f"processed {data}"

class TestMyClass(unittest.TestCase):
    def test_process_data(self):

        my_instance = MyClass()
        my_instance.fetch_data = MagicMock(return_value="mocked data")

        result = my_instance.process_data()
        self.assertEqual(result, "processed mocked data")
```

Experiment Tracking



Data Collection + Labelling + Wrangling

Model Development

Train + Validate

Test

Deploy!

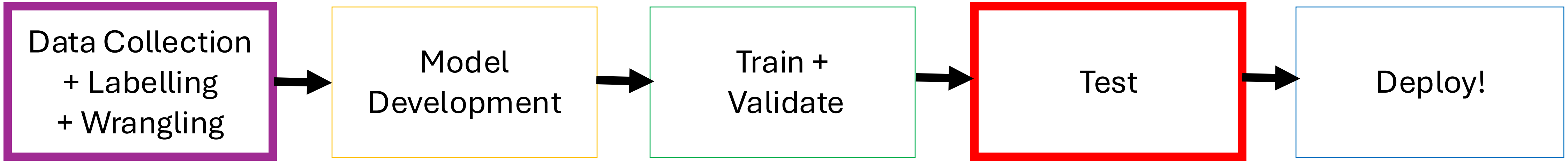
- ML Experiment Tracking is how you unit-test your model during ***training***.
- Treat it like a CI/CD github action. Every time you "commit" you test. Every time you "train" you test.

W&B

TensorBoard

mlflow™

How do you unit test an ML Model?

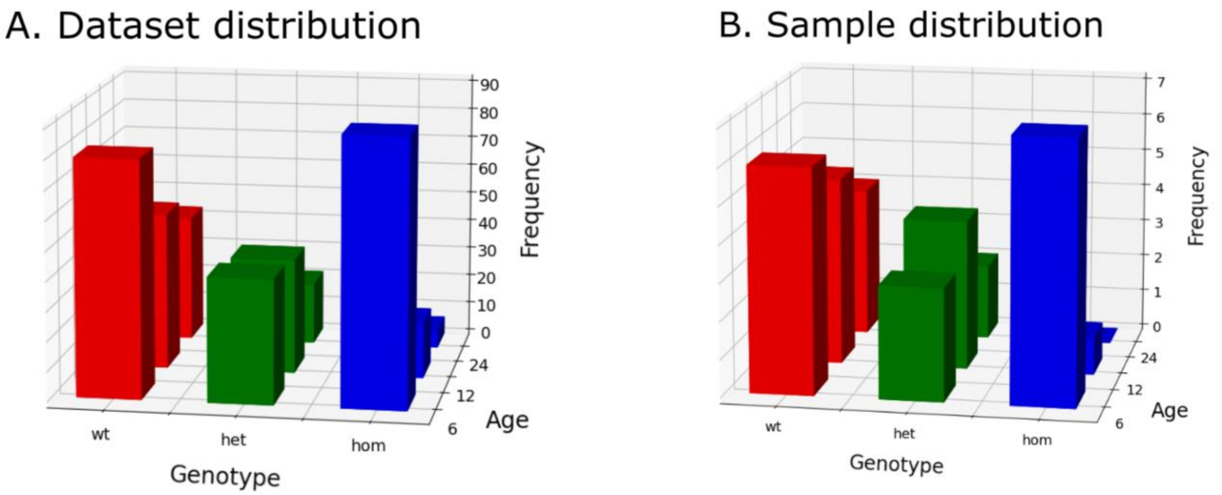


- Unit testing during deployment
  1. **Dataset design**
     - **Limited**
     - **Unlimited**
  2. IO
     - Input sensitivity
     - Output uncertainty

Dataset design (limited)

- Most of the time our data is really limited. High p low n problem.
- Should you stratify your sampling? OR should you bias your samples?
- Stratified sampling vs dataset curation

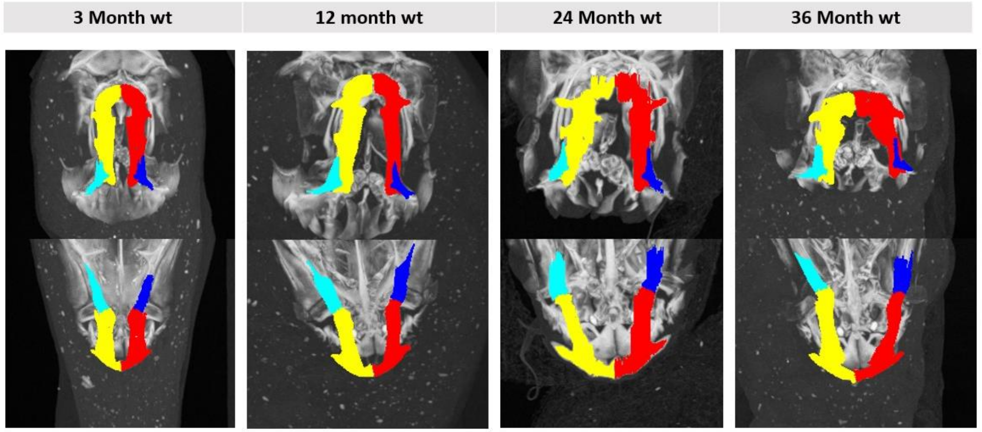
- "I'm not interested in bones, I'm interested in broken bones"
- Deployability is determined by the edge cases.
- Stratified sampling amplifies survivor bias in the dataset.

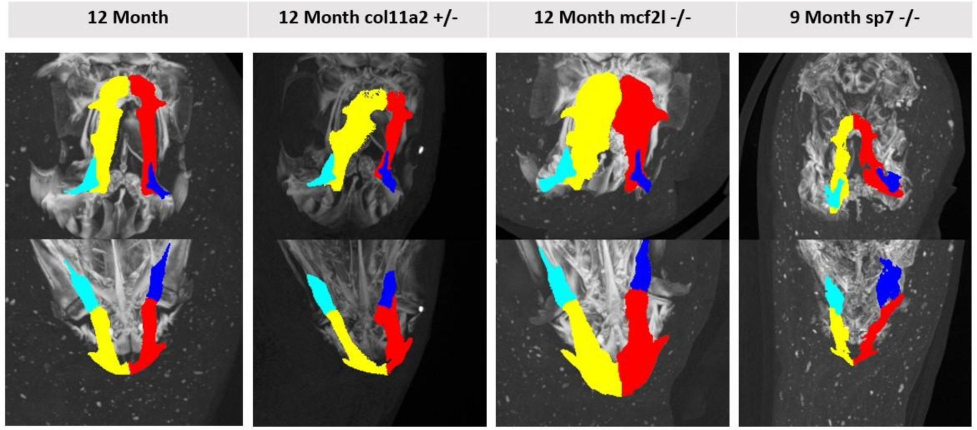

A. Dataset distribution

B. Sample distribution

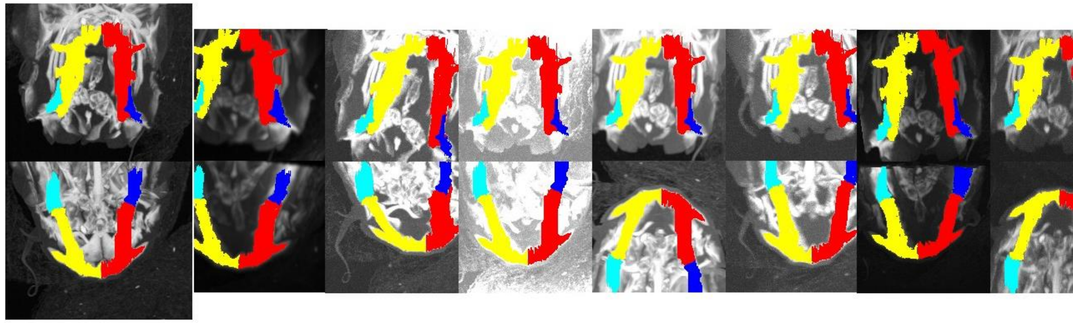
Dataset design (limited)

- Use Data augmentation for unit tests!
- How sensitive is your model to data augmentation?
- "Sensitivity analysis is the study of how the uncertainty in the output of a mathematical model [...] can be divided and allocated to different sources of uncertainty in its inputs."
- Use data augmentation to probe the edge cases





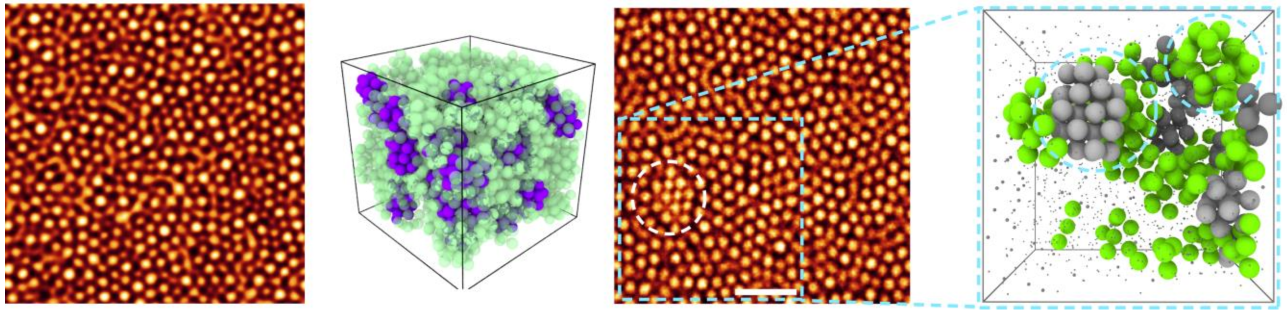
Jaw Augmentation

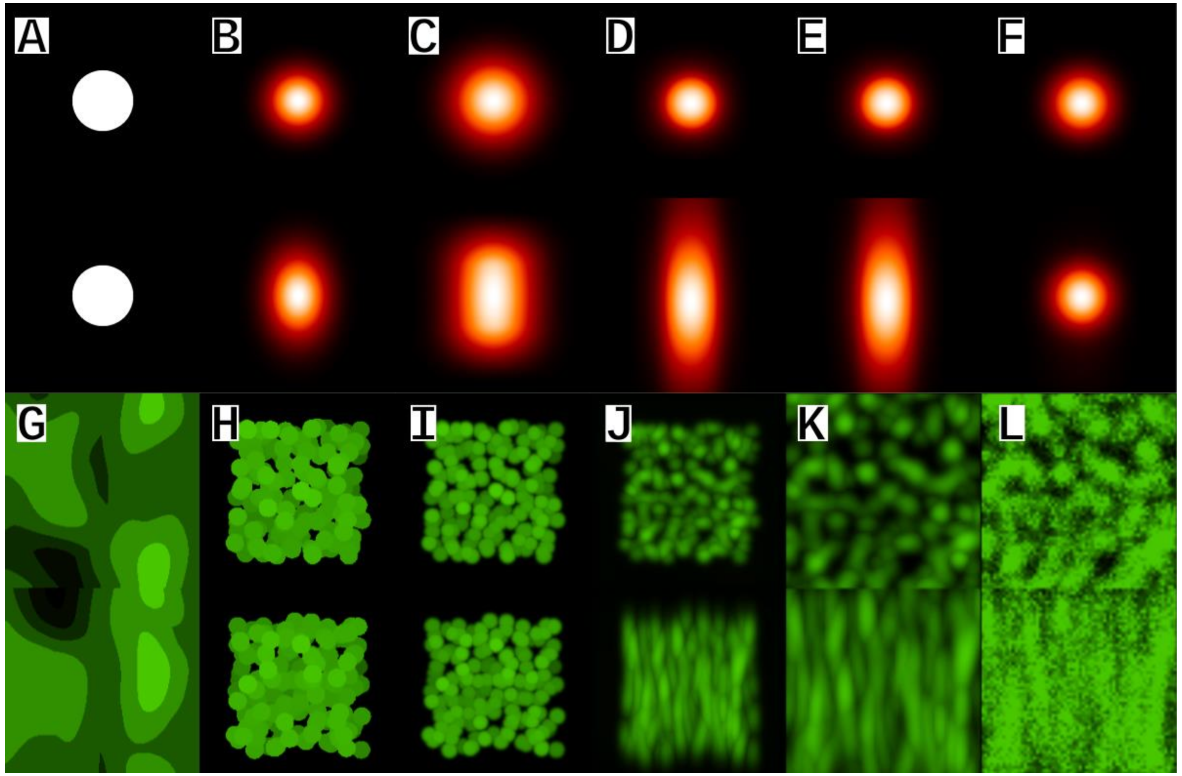


Dentary 🟥 🟨 Quadrate 🟦 🟦

Dataset design (unlimited)

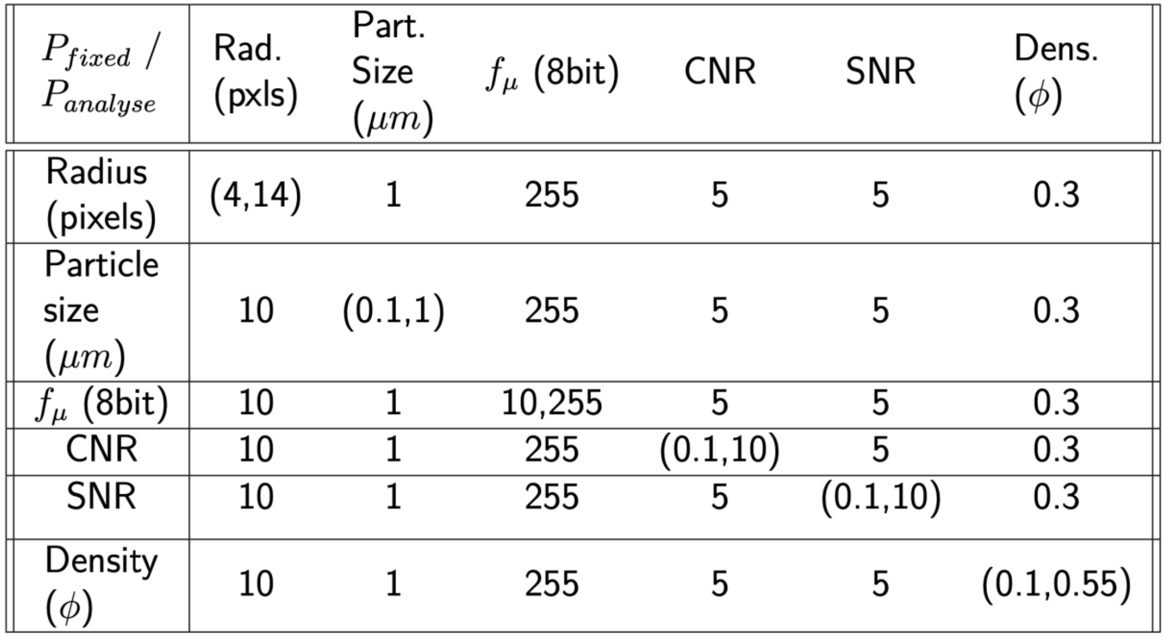
- Unlimited: When you can simulate your data
- Example: Detecting spheres
- How to train an ML model on simulations?





Example: Detecting Spheres

- How to train an ML model on simulations?
- How do I generate the parameters for my simulation?
- One-at-a-time?



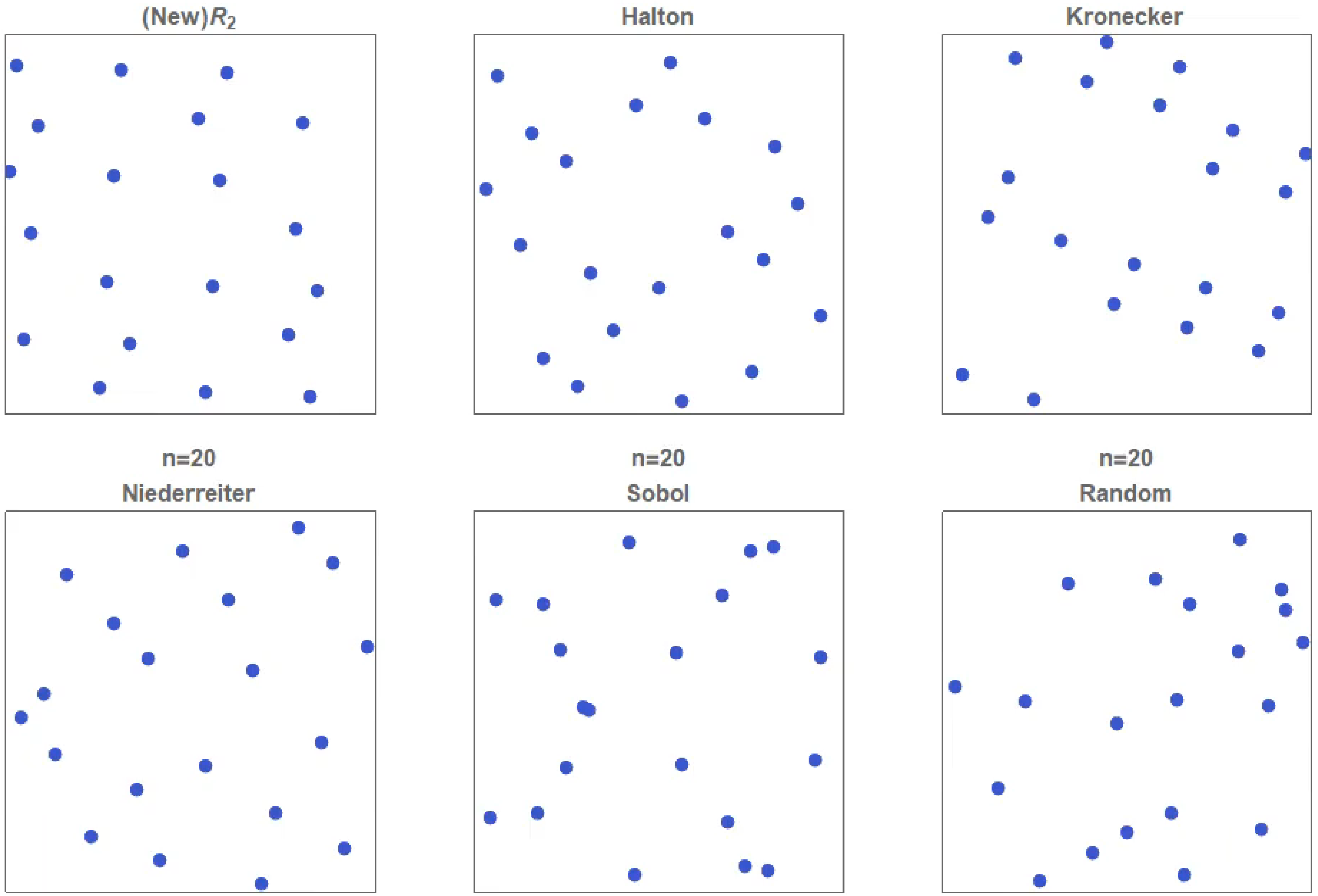
P_{fixed} / $P_{analyse}$	Rad. (pxls)	Part. Size (μm)	f_μ (8bit)	CNR	SNR	Dens. (ϕ)
Radius (pixels)	(4,14)	1	255	5	5	0.3
Particle size (μm)	10	(0.1,1)	255	5	5	0.3
f_μ (8bit)	10	1	10,255	5	5	0.3
CNR	10	1	255	(0.1,10)	5	0.3
SNR	10	1	255	5	(0.1,10)	0.3
Density (ϕ)	10	1	255	5	5	(0.1,0.55)

Table 2.4: Diagonal distribution of parameter sweeps. This allows the investigation of the effect of each parameter separately.

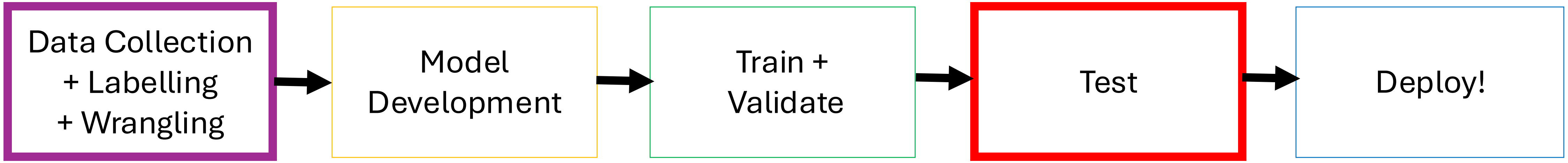
Dataset design (unlimited)

- Systems Engineering
- Design Space Exploration / Low-discrepancy sequence
  - One-at-a-time
  - Sobol sequences
  - Latin Hyper Cube

- Generating sample points:
  - scipy.stats.qmc.sobol()
- Attributing variance in model predictions to input parameters:
  - scipy.stats.sobol_indices()



How do you unit test an ML Model?

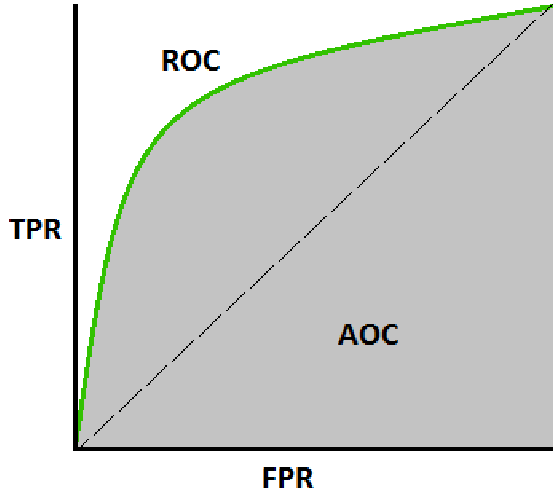


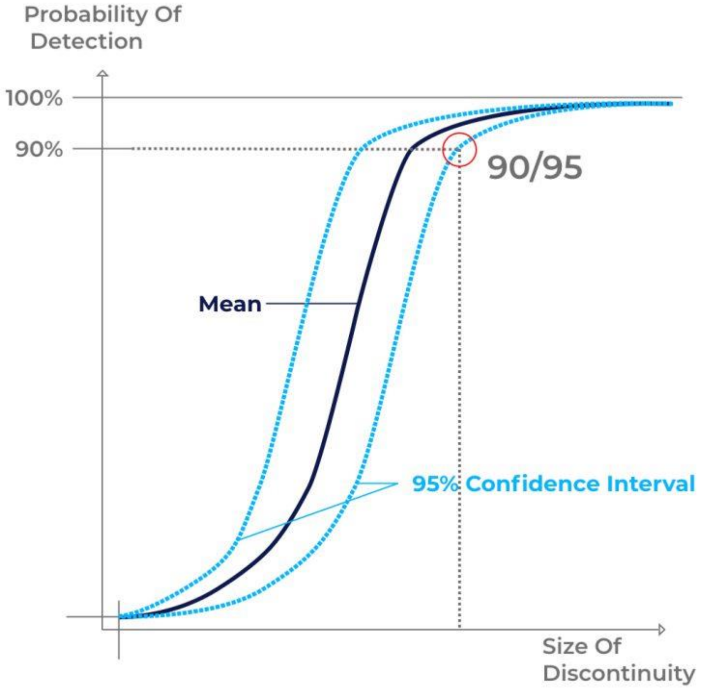
- Unit testing during deployment
  1. Dataset design
     - Limited
     - Unlimited
  2. **IO**
     - **Input sensitivity**
     - **Output uncertainty**

Sensitivity analysis

- Medicine:
  - How are blood test thresholds set?
  - AUC ROC
  - Is this good enough?
  - Move sobol indices here!

- Aerospace NDT (Non-destructive testing):
  - Military Handbook 1823a
  - Probability of detection
  - A90/95
  - Fastener/bracket inspection





Uncertainty Quantification - Is this good enough?

- Uncertainty quantification
  - Ensemble/Bootstrapping
  - Monte Carlo Dropout
  - Test-time augmentation
  - GP Final Layer
- https://github.com/VNemani14/UQ_ML_Tutorial

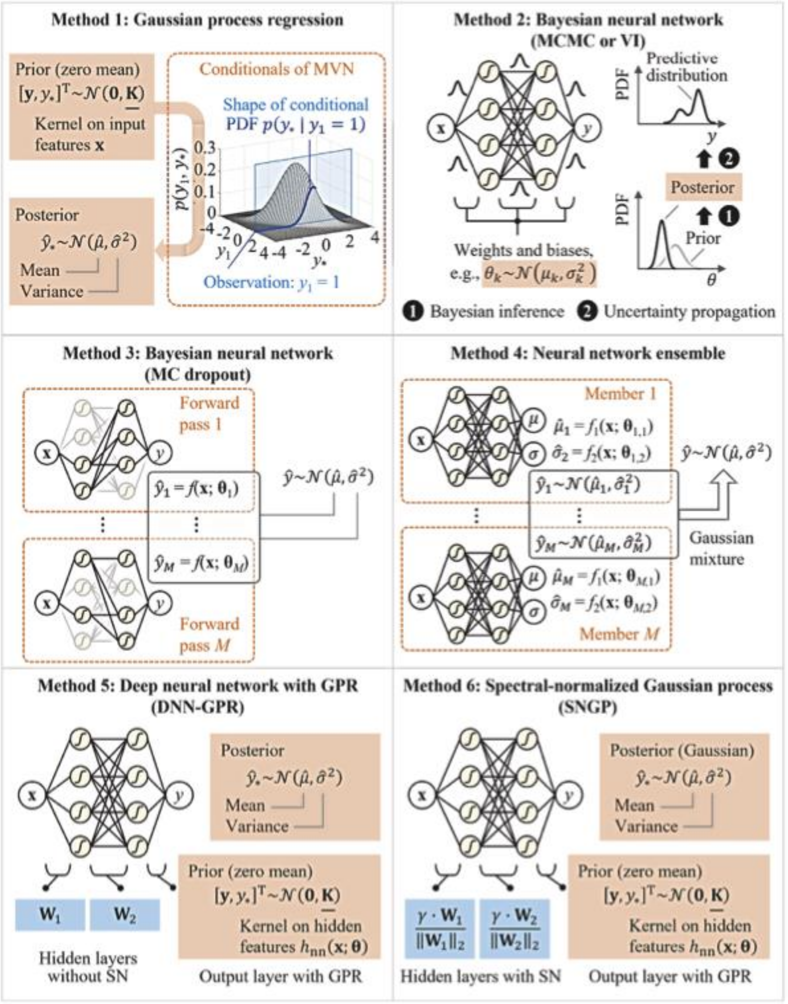

Method 1: Gaussian process regression
Prior (zero mean)
$[\mathbf{y}, y_*]^T \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$
Kernel on input features $\mathbf{x}$
Conditionals of MVN
Shape of conditional PDF $p(y_* \mid y_1 = 1)$
$p(y_1, y_*)$
Observation: $y_1 = 1$
Posterior
$\hat{y}_* \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Mean
Variance
Method 2: Bayesian neural network (MCMC or VI)
Predictive distribution
PDF
Posterior
Prior
Weights and biases, e.g., $\theta_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$
❶ Bayesian inference
❷ Uncertainty propagation
Method 3: Bayesian neural network (MC dropout)
Forward pass 1
$\hat{y}_1 = f(\mathbf{x}; \boldsymbol{\theta}_1)$
$\hat{y}_M = f(\mathbf{x}; \boldsymbol{\theta}_M)$
Forward pass M
$\hat{y} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Method 4: Neural network ensemble
Member 1
$\hat{\mu}_1 = f_1(\mathbf{x}; \boldsymbol{\theta}_{1,1})$
$\hat{\sigma}_2 = f_2(\mathbf{x}; \boldsymbol{\theta}_{1,2})$
$\hat{y}_1 \sim \mathcal{N}(\hat{\mu}_1, \hat{\sigma}_1^2)$
$\hat{y}_M \sim \mathcal{N}(\hat{\mu}_M, \hat{\sigma}_M^2)$
$\hat{\mu}_M = f_1(\mathbf{x}; \boldsymbol{\theta}_{M,1})$
$\hat{\sigma}_M = f_2(\mathbf{x}; \boldsymbol{\theta}_{M,2})$
Member M
$\hat{y} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Gaussian mixture
Method 5: Deep neural network with GPR (DNN-GPR)
Posterior
$\hat{y}_* \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Mean
Variance
Prior (zero mean)
$[\mathbf{y}, y_*]^T \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$
Kernel on hidden features $h_{nn}(\mathbf{x}; \boldsymbol{\theta})$
$\mathbf{W}_1$
$\mathbf{W}_2$
Hidden layers without SN
Output layer with GPR
Method 6: Spectral-normalized Gaussian process (SNGP)
Posterior (Gaussian)
$\hat{y}_* \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Mean
Variance
Prior (zero mean)
$[\mathbf{y}, y_*]^T \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$
Kernel on hidden features $h_{nn}(\mathbf{x}; \boldsymbol{\theta})$
$\frac{\gamma \cdot \mathbf{W}_1}{\|\mathbf{W}_1\|_2}$
$\frac{\gamma \cdot \mathbf{W}_2}{\|\mathbf{W}_2\|_2}$
Hidden layers with SN
Output layer with GPR

Fig. 4. Graphical comparison of six state-of-the-art UQ methods introduced in Section 3. These methods are GPR (method 1), BNN via MCMC or VI (method 2), BNN via MC dropout (method 3), neural network ensemble (method 4), DNN with GPR — DNN-GPR (method 5), and SNGP (method 6). In method 1, MVN standards for the multivariate normal distribution, or equivalently, the multivariate Gaussian distribution used in the main text. In methods (5) and (6), SN stands for spectral normalization.

LLMs: AISI – AI Safety Institute

- Evals: Evaluation questions and answers
- Inspect-ai https://github.com/UKGovernmentBEIS/inspect_ai
- MLCommons AI Safety Benchmark https://github.com/mlcommons/modelbench
- Llama Guard https://github.com/meta-llama/PurpleLlama

- Aleatory and Epistemic Uncertainty
- “Decomposing Uncertainty for Large Language Models through Input Clarification Ensembling” – Hou et. al. 2024

Conclusion

- Unit testing during training (Mock testing & Experiment tracking)
- Unit testing during deployment
  1. Datasets
     - Limited: **Stratified sampling, dataset curation + augmentation**
     - Unlimited: **Sobol sequences, latin hypercubes**
  2. IO
     - Input: **Sensitivity Analysis**
     - Output: **Uncertainty Quantification**
- **Email: a.kawafi@bristol.ac.uk**

Background & Useful Links

- https://eugeneyan.com/writing/unit-testing-ml/
- https://datahazards.com/labels.html
- https://thenerdstation.medium.com/how-to-unit-test-machine-learning-code-57cf6fd81765