How do you unit test an ML model?

Wahab Kawafi

Best Practices in AI Afternoon
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-Al
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

<table>
<thead>
<tr>
<th>Tolerance class</th>
<th>Straightness and flatness tolerances for ranges of nominal lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>up to 10</td>
</tr>
<tr>
<td>H</td>
<td>0,02</td>
</tr>
<tr>
<td>K</td>
<td>0,05</td>
</tr>
<tr>
<td>L</td>
<td>0,1</td>
</tr>
</tbody>
</table>
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

- 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.
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.
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
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?

<table>
<thead>
<tr>
<th>$P_{\text{fixed}} / P_{\text{analyse}}$</th>
<th>Rad. (pxls)</th>
<th>Part. Size (μm)</th>
<th>$f_\mu$ (8bit)</th>
<th>CNR</th>
<th>SNR</th>
<th>Dens. ($\phi$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius (pixels)</td>
<td>(4,14)</td>
<td>1</td>
<td>255</td>
<td>5</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>Particle size (μm)</td>
<td>10</td>
<td>(0.1,1)</td>
<td>255</td>
<td>5</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>$f_\mu$ (8bit)</td>
<td>10</td>
<td>1</td>
<td>10,255</td>
<td>5</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>CNR</td>
<td>10</td>
<td>1</td>
<td>255</td>
<td>(0.1,10)</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>SNR</td>
<td>10</td>
<td>1</td>
<td>255</td>
<td>5</td>
<td>(0.1,10)</td>
<td>0.3</td>
</tr>
<tr>
<td>Density ($\phi$)</td>
<td>10</td>
<td>1</td>
<td>255</td>
<td>5</td>
<td>5</td>
<td>(0.1,0.55)</td>
</tr>
</tbody>
</table>

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

• Aerospace NDT (Non-destructive testing):
  o Military Handbook 1823a
  o Probability of detection
  o A90/95
  o Fastener/bracket inspection
Uncertainty Quantification - Is this good enough?

• Uncertainty quantification
  o Ensemble/Bootstrapping
  o Monte Carlo Dropout
  o Test-time augmentation
  o GP Final Layer

• https://github.com/VNemani14/UQ_ML_Tutorial
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