Engineering best practices for machine learning

Alex Serban

Radboud University, Software Improvement Group, Leiden University
The Netherlands
Who am I?

Radboud University

Software Improvement Group

Leiden Institute for Advanced Computer Science

Software Engineering for Machine Learning
https://se-ml.github.io
Machine learning robustness

- Robustness has multiple facets, e.g., algorithmic robustness, system or software robustness.

- Algorithmic robustness describes the ability of an algorithm to maintain training performance when tested on new and noisy samples.

- System robustness describes the ability of a system to cope with errors and erroneous inputs during execution.

- When machine learning is used, robustness is broader and includes trustworthy concerns such as fairness, privacy, transparency, etc.
Robustness in the wild

Figure 1: Natural adversarial examples from IMAGENET-A. The red text is a ResNet-50 prediction with its confidence, and the black text is the actual class. Many natural adversarial examples are incorrectly classified with high confidence, despite having no adversarial modifications as they are

Who to Sue When a Robot Loses Your Fortune

The first known case of humans going to court over investment losses triggered by autonomous machines will test the limits of liability.
End to end machine learning engineering

- The development of **engineering principles** for the design, development, operation and maintenance of software systems with ML components.
Traditional software engineering

- Traditional software engineering tackles challenges related to software design, development, and operation.

- Such challenges can be classified in functional and non-functional.

- An example of functional SE challenge is verifying that a system will satisfy its intended functionality (e.g., through testing or formal verification).

- Examples of non-functional SE challenges are maintainability, scalability, usability, etc. (also called “-ilities” due to their suffix).
Traditional software engineering in machine learning

- Traditional software engineering practices are also relevant for ML projects.
- The tool support for checking traditional practices is mature and openly available (typically free of cost).
- However, in ML systems traditional software engineering practices are not prioritised.
- Contributing factors are general unawareness of best practices due to heterogeneous backgrounds.
- As research code is cloned and modified, these issues perpetuate.
Concrete software engineering issues in machine learning

Pictures generated by forking the huggingface/transformers repository and running the BetterCodeHub tool
Benefits of traditional software engineering

- Research in software engineering has shown benefits of tackling these issues in terms of maintainability, reusability and general effort reduction.

- To facilitate adoption of engineering principles by practitioners, they must be actionable.

- Adopting "off-the-shelf" solution from traditional software engineering in ML should entail similar results.

- **Challenge**: Run a static analysis tool on some of your ML code / open source framework.
Machine learning vs. traditional software

from an engineering perspective

data intensive

inherent uncertainty

empirical iteration
Machine learning vs. traditional software

from a social and organizational perspective

- sky-high expectations
- wide talent gap
- potential for harm
Risks posed by machine learning

**COMPAS** = Correctional Offender Management Profiling for Alternative Sanctions
Predict recidivism – will a person become a repeat offender?
Used to decide who can be released from jail on bail pending trial

---

**Prediction Fails Differently for Black Defendants**

<table>
<thead>
<tr>
<th></th>
<th>WHITE</th>
<th>AFRICAN AMERICAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled Higher Risk, But Didn’t Re-Offend</td>
<td>23.5%</td>
<td>44.9%</td>
</tr>
<tr>
<td>Labeled Lower Risk, Yet Did Re-Offend</td>
<td>47.7%</td>
<td>28.0%</td>
</tr>
</tbody>
</table>

*Overall, Northpointe’s assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes.* (Source: ProPublica analysis of data from Broward County, Fla.)

Regulation is on its way

On 8 April 2019, the High-Level Expert Group on AI presented the **Ethics Guidelines for Trustworthy Artificial Intelligence**.

Trustworthy means:

- Lawful
- Ethical
- Robust

“[T]he views expressed in this document reflect the opinion of the AI HLEG and may not in any circumstances be regarded as reflecting an official position of the European Commission.”

Seven key requirements

Evaluate and address these continuously throughout the AI system’s lifecycle, via:

- **Technical methods**
  e.g., Constraints in the software architecture, embedded in design and implementation. Explanation functionality. Deliberate testing and validation. Measure algorithm quality indicators.

- **Non-technical methods**
  e.g., Regulations, code of conduct, standardization, certification, governance, education, awareness, stakeholder participation, diversity in design teams.
How are software engineering practices impacted by incorporation of ML components in software systems?

What new practices are being proposed by researchers and practitioners?

To what extent are practices adopted by engineering teams?

What are the effects of practices adoption on the quality of systems that incorporate ML components?
Investigating machine learning engineering practices

- Review literature
- Create practice catalog
- Survey adoption and effects
- Interview practitioners
- Add trustworthiness
- Add AutoML
- Add architecture

Academic and grey literature

400+ practitioners

Serban et al, “Adoption and effects of software engineering best practices in machine learning”, ESEM 2020

Awesome reading list

29 practices in fixed format

ranking of practices and links to effects

“State of ML engineering practices” report

+14 practices and link to seven requirements

“State of AutoML” report

Tactics linked to quality aspects

Alex Serban
cs.ru.nl/~aserban
Online catalog of engineering practices for ML

Originally, 29 practices. Now grown to 45.

Grouped into 6 categories.

- Intent
- Motivation
- Applicability
- Description
- Adoption
- Related practices
- References

Ranked on difficulty

17

Alex Serban
cs.ru.nl/~aserban
Example practice

Title

- Intent
- Motivation
- Applicability
- Description
- Adoption
- Related practices
- References

Use Sanity Checks for All External Data Sources

January, 2021 • Alex Serban, Koen van der Blom, Joost Visser

Intent
Avoid invalid or incomplete data being processed.

Motivation
Data is at the heart of any machine learning model. Therefore, avoiding data errors is crucial for model quality.

Applicability
Data quality control should be applied to any machine learning application.

Description
Whenever external data sources are used, or data is collected that may be incomplete or ill formatted, it is important to verify the data quality. Invalid or incomplete data may cause outages in production or lead to inaccurate models.

Start by checking simple data attributes, such as:

- data types,
- missing values,
- data min. or max. values,
- histograms of continuous values,

and gradually include more complex data statistics, such as the ones recommended here.

Missing data can also be substituted using data imputation; such as imputation by zero, mean, median, random values, etc.

Also, make sure the data verification scripts are reusable and can be later integrated in any processing pipeline.
Measuring practice adoption

Survey among teams building software that incorporates ML components.

Questions:

- **General**
  - ex. Team size, team experience, country, kind of organization, type of data, tools used.

- **Practices**
  - ex. “Our process for deploying our ML model is fully automated.”

- **Effects**
  - ex. “We are able to easily and precisely reproduce past behavior of our models and applications.”

- **Not at all**
- **Partially**
- **Mostly**
- **Completely**
The adoption of best practices by tech companies is higher than by non-tech companies, governmental organizations, and research labs.
Practice adoption increases with team size and experience.

Larger teams tend to adopt more practices.

More experienced teams tend to adopt more practices.

Except teams with more than 5 years experience!
ML-specific practices are adopted slightly more than traditional SE practices

Among ML teams, the adoption of ML-specific practices is highest, followed by general Software Engineering (SE) practices and SE practices adapted to ML.
Practice adoption by data type

The adoption of practices is largely independent of the data type used.
Title

Nr • Category • Difficulty

- Intent
- Motivation
- Applicability
- Description
- Adoption
- Related practices
- References

Use Sanity Checks for All External Data Sources

Intent
Avoid invalid or incomplete data being processed.

Motivation
Data is at the heart of any machine learning model. Therefore, avoiding data errors is crucial for model quality.

Applicability
Data quality control should be applied to any machine learning application.

Description
Whenever external data sources are used, or data is collected that may be incomplete or ill formatted, it is important to verify the data quality. Invalid or incomplete data may cause outliers in production or lead to inaccurate models.

Start by checking simple data attributes, such as:
- data types,
- missing values,
- data min. or max. values,
- histograms of continuous values,

and gradually include more complex data statistics, such as the ones recommended here.

Missing data can also be substituted using data imputation; such as imputation by zero, mean, median, random values, etc.

Also, make sure the data verification scripts are reusable and can be later integrated in any processing pipeline.
Example practice

Title

Nr • Category • Difficulty

- Intent
- Motivation
- Applicability
- Description
- Adoption
- Related practices
- References

- Not at all
- Partially
- Mostly
- Completely
Example practice

Title

Nr • Category • Difficulty

- Intent
- Motivation
- Applicability
- Description
- Adoption
- Related practices
- References

Related

- Check that Input Data is Complete, Balanced and Well Distributed
- Write Reusable Scripts for Data Cleaning and Merging

Read more

- Data management challenges in production machine learning
- ML Ops: Machine Learning as an engineered disciplined
Capture the training objective in a metric that is easy to measure and understand
Share a clearly defined training objective within the team
Use versioning for data, models, configurations and training scripts
Continuously measure model quality and performance
Write reusable scripts for data cleaning and merging
Enable parallel training experiments
Share status and outcomes of experiments within the team
Use a collaborative development platform
Work against a shared backlog
Communicate, align and collaborate with multidisciplinary team members
Ensure data labeling is performed in a strictly controlled process
Continuously monitor the behaviour of deployed models
Enable automatic roll backs for production models
Make data sets available on shared infrastructure
Automate model deployment
Use continuous integration
Perform checks to detect skews between models
Check that input data is complete, balanced and well distributed
Log production predictions with the model’s version and input data
Peer review training scripts
Enforce fairness and privacy
Use sanity checks for all external data sources
Test all feature extraction code
Use static analysis to check code quality
Enable shadow deployment
Automate hyper-parameter optimisation and model selection
Run automated regression tests
Actively remove or archive features that are not used
Assign an owner to each feature and document its rationale

Practices are ranked by the average of: their rank on Completely, their rank on Completely+Mostly, and their rank on Completely+Mostly+Partially.
Most adopted practices

Practices related to **measurement** and **versioning** are widely adopted.

The top 4 adopted practices are all related to **model training**.

**Top 5**

1. Capture the training objective in a metric that is easy to measure and understand

2. Share a clearly defined training objective within the team

3. Use versioning for data, model, configurations and training scripts

4. Continuously measure model quality and performance

5. Write reusable scripts for data cleaning and merging
Least adopted practices

The two most neglected practices are related to feature management.

Outside research, Automated ML through automated optimisation of hyper-parameters and model selection, is not (yet) widely applied.

Bottom 5

1. Assign an owner to each feature and document its rationale
2. Actively remove or archive features that are not used
3. Run automated regression tests
4. Automate hyper-parameter optimisation and Model Selection
5. Enable shadow deployment
Measuring effects of practice adoption

For four effects, we hypothesized a relation with a specific selection of practices.

- **Linear regression**
  Confirmed hypotheses.

- **Non-linear regression – Random Forest**
  Demonstrated non-linear influence.

- **Importance of each practice – Shapley**
  Some very important practices have low adoption.

<table>
<thead>
<tr>
<th>Effects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agility</td>
<td>The team can quickly experiment with new data and algorithms, and quickly assess and deploy new models</td>
</tr>
<tr>
<td>Software Quality</td>
<td>The software produced is of high quality (technical and functional)</td>
</tr>
<tr>
<td>Team Effectiveness</td>
<td>Experts with different skill sets (e.g., data science, software development, operations) collaborate efficiently</td>
</tr>
<tr>
<td>Traceability</td>
<td>Outcomes of production models can easily be traced back to model configuration and input data</td>
</tr>
</tbody>
</table>
Different practices, different outcomes

Analysis of survey responses shows that desired outcomes such as traceability, agility, team effectiveness, and software quality are each related to specific sets of practices.

Per desired outcome, we list the three practices with the largest influence.
Key findings

Tech companies are leading in adoption of ML engineering best practices.

Larger and more experienced teams tend to adopt more practices.

General software engineering practices enjoy slightly lower adoption than specific machine learning practices.

Best practices for feature management are the least well adopted.

Desired outcomes such as traceability, agility, effectiveness, and quality are each related to specific sets of practices.

From 2020 global survey on adoption of 29 practices, among 350 teams.
Software Engineering practices in the age of ML

How are software engineering practices impacted by incorporation of ML components in software systems?

What new practices are being proposed by researchers and practitioners?

To what extent are practices adopted by engineering teams?

What are the effects of practices adoption on the quality of systems that incorporate ML components?

- **Trustworthiness**
  More practices? Link to requirements?

- **Architecture**
  Practices as tactics to reach architectural goals.

- **AutoML**
  Transfer from research to broad adoption?
Back to the
Seven key requirements

Evaluate and address these continuously throughout the AI system’s lifecycle, via:

- **Technical methods**
  e.g., Constraints in the software architecture, embedded in design and implementation. Explanation functionality. Deliberate testing and validation. Measure algorithm quality indicators.

- **Non-technical methods**
  e.g., Regulations, code of conduct, standardization, certification, governance, education, awareness, stakeholder participation, diversity in design teams.
New practices, mapped to trustworthiness requirements

Lawful
- Human agency and oversight
- Technical robustness and safety
- Privacy and data governance
- Transparency
- Diversity, non-discrimination, fairness
- Societal, environmental well-being

Ethical

Robust
- Accountability

- T1 - Test for social bias in training data
- T2 - Prevent discriminatory data attributes as model features
- T3 – Use privacy-preserving ML techniques
- T4 – Employ interpretable models whenever possible
- T5 – Assess and manage subgroup bias
- T6 – Assure application security
- T7 – Provide audit trails
- T8 – Decide trade-offs through an established team process
- T9 – Establish responsible AI values
- T10 – Perform risk assessments
- T11 – Inform users on ML usage
- T12 – Explain results and decision to users
- T13 – Provide safe channels to raise concerns
- T14 – Have your application audited
ML engineering practices for research

Write Reusable Scripts for Data Cleaning and Merging
March, 2021 • Alex Serban, Koen van der Blok, Joost Visser

Intent
Avoid untidy data wrangling scripts, reuse code and increase reproducibility.

Motivation
Data cleaning and merging are exploratory processes and tend to lack structure. Many times these processes involve manual steps, or poorly structured code which can not be reused later. Needless to mention such code can not be integrated in a processing pipeline.

Applicability
Reusable data cleaning scripts should be written for any ML application that does not use raw or standard data sets.

Description
Most of the time, training machine learning models is preceded by an exploratory phase, in which non-structured code is written, or manual steps are performed in order to get the data in the right format, merge several data sources, etc. Especially when using notebooks, there is a tendency to write ad-hoc data processing scripts, which depend on variables already stored in memory when running previous cells.

Before moving to the training phase, it is important to convert this code into reusable scripts and move it into methods which can be called and tested individually. This will enable code reuse and ease integration into processing pipelines.
ML engineering practices for research

Share Status and Outcomes of Experiments Within the Team
March, 2021 • Alex Serban, Koen van der Blom, Jool Visser

Intent
Facilitate knowledge transfer, peer review and model assessment.

Motivation
Team members have different ways of managing and logging experiment related data. Adopting a common way to log experiment data and share it within the team enables members to collectively monitor and assess training outcomes.

Applicability
Experiment tracking and sharing should be used for any training experiment.

Description
Although different team members have their own style of managing experiments and tracing their outcomes, it is recommended to adopt a common way of logging data; that is understood and accessible to all team members.

Sharing the outcomes within the team has several benefits for peer review, knowledge transfer and model assessment.

Several collaborative tools enable central logging of experimental results.

Whenever possible, it is recommended to use one of the tools available internally or externally (e.g. Sacred or W&B).
ML engineering practices for research

Use Static Analysis to Check Code Quality

March, 2021 • Jocel Visser, Alex Serban, Koen van der Blom

Intent
Avoid the introduction of code that is difficult to test, maintain, or extend.

Motivation
High-quality code is easier to understand, test, maintain, reuse, and extend. The most effective way of ensuring high code quality is to make use of static analysis tools.

Applicability
Code quality control should be applied to any type of code.

Description
By ensuring high code quality you can avoid the introduction of defects into the code, enable new team members to become productive more quickly, and more easily reason about the correctness of your code.

Static code analysis can be done in various ways:

• Linters: A linter is a tool that finds undesirable patterns in program code and reports these back to the programmer. Linters can be activated in a code editor, and integrated development environment, or they can be run on the command line.

• Quality gates: You can integrate a static code quality analysis tool in an automated build and testing script that runs every time a developer commits code changes to the versioning system. When quality issues are found, you can choose to have the commit rejected.
ML engineering practices for research

Use A Collaborative Development Platform
March, 2021 • Joost Vasse, Alex Serban, Koen van der Blom

<table>
<thead>
<tr>
<th>Team</th>
<th>Difficulty</th>
<th>Scale</th>
<th>Effect</th>
<th>Coverage</th>
</tr>
</thead>
</table>

**Intent**
By making consistent use of a collaborative development platform teams can work together more effectively.

**Motivation**
Collaborative development platforms provide easy access to data, code, information, and tools. They also help teams to keep each other informed, make and record decisions, and work together asynchronously or remotely.

**Description**
Broadly used collaborative development environments include GitHub, GitLab, BitBucket, and Azure DevOps Server.

Some collaborative development environments are offered as cloud services, others may be installed on-premises, or both. Commonly offered capabilities include:

- Version control
- Issue and progress tracking
- Search, notifications, discussion
- Continuous integration
- A range of developer tools as (third-party) plugins

Collaborative development environments have been developed for, and gained widespread adoption by, “traditional” software development teams.

Adoption by org. type
![Adoption by org. type chart](image)
Take away

Software that incorporates Machine Learning (or other AI) **challenges** traditional software engineering practices, due to data intensity, inherent uncertainty, and iterative empirical design.

Demand for **robust** and **responsible** development and use are not unique to ML, but become more acute.

Engineering **practices** are being modified and developed at a quick pace. **Adoption** varies and **effects** are not well-understood.

Software Engineering researchers should **embrace** the challenge of ML, investigate and enhance practice development.
Reading list
We reviewed scientific and popular literature to identify recommended practices. Check out this Awesome List with relevant literature.

Catalogue
The best practices that we identified are described in more detail in this Catalogue of ML Engineering Best Practices.

Preprints
Full details of the methodology behind our survey are described in scientific articles. Read the preprints here.

se-ml.github.io
Visit our project website for more details, to take the survey yourself, and to stay up-to-date with our latest results.
Team

Alex Serban
Koen van der Blom

Holger Hoos
Joost Visser

https://se-ml.github.io/members/
LIACS, Leiden University, The Netherlands
ICIS, Radboud University, The Netherlands
University of British Columbia, Canada