Techworks Best Practices in AI

version

Best Practices Working Group

May 28, 2024

Contents

Welcome to the hub for the Techworks Best Practices in AI guide.

The guide is one of the core activities of the Techworks AI Innovation Cross Working Group (AI xWG). The goal of the guide is to provide best practices guidance to electronic systems engineers in the UK to bridge the UK AI skill gap.

To get started:

- Read more about this project and the guide.
- Learn about how you can get involved.
- Read the executive summary.
- Or jump straight to the first page of the guide.

This guide is currently maintained in several formats:

- Online
- PDF
- docx

Contributors

We thank all contributors and reviewers of this project.

Authors

William Jones

Contributors

Alex Wang

Allison Lowndes

Andrew Rogoyski

Andy Bond

Arezou Nayebi

Charles Sturman

Dimitra Georgiadou

Elliot Stein

Gopal Ramchurn

James Watson

Jeremy Bennett

Lauren Thompson

Mark Zwolinski

Mike Bartley

Prethveraj M S

Tim Santos

About

This is the Techworks Best Practices in Artificial Intelligence Guide. It is one of the core activities of the Techworks AI Innovation Cross Working Group (AI xWG).

The goal of this guide is to help address the AI skill gap in the UK by empowering our existing engineers to fully adavantage of this technology. The guide achieves this by laying out best practices guidance for electronic systems engineers that will allow them both to:

- Quickly access and understand key concepts and ideas in AI and Machine Learning; and
- Work through an AI project from first principles to a completed product.

One of the key strengths of this guide is that it is specifically tailored to retraining existing electronic systems engineers. These engineers already have many essential skills, allowing the guide to be highly focused and compact. It provides a marked contrast to other guides, which typically assume very low levels of audience knowledge to reach the widest possible audience.

Getting Involved

The Techworks Best Practices in AI guide is maintained openly, and we welcome external contributors.

As an engineering oriented guide we especially looking for contributors with strong technical skills. However, we recognize the importance of a diversity of skill-sets and experiences in producing a robust guide. There are also many important areas in the guide (for example legal and ethics), which we recognize are out of the usual tech skillsets. In short, if you feel you have something useful to contribute to the guide, please get in touch.

Group activity for developing this guide is currently co-ordinated via Basecamp. The group currently meets virtually once a month on each fourth tuesday. If you are interested in taking part, or you have thoughts or comments then:

- Get in touch

You are also welcome to feedback on this guide directly. You can do this by:

- Sending us your comments
- Opening an issue or a pull request on Github.

This guide is currently maintained in Sphinx on GitHub.

Executive Summary

This guide is for professional software engineers in the electronics systems industry who now need to implement Artificial Intelligence and Machine Learning (AI/ML) solutions. We make the following two assumptions:

1. the reader is an experienced software engineer; and
2. the reader has no prior experience of AI/ML.

The guide takes the form of a series of questions to be answered. Comprehensive supplementary information is provided to help answer these questions. The outcome is an optimally designed, implemented and deployed AI/ML solution.

Using this guide

This guide will help you answer 5 questions step by step:

- Should I use AI/ML?
- How do I define my AI/ML project
- How do I collect data?

- How do I train my AI/ML application?
- How do I deploy my AI/ML application?

Answering these questions will take you from conceptualizing your AI application through to deploying it in the real world.

These 5 questions are broken into smaller questions, requirements, and tasks to be completed. We set out the question in the order they should be considered. We strongly recommend you don't skip ahead, although from time to time you may want to go back and review your answers to earlier questions.

These smaller questions, requirements and tasks for each of the 5 main questions are recorded in a table, showing what the activity is, and where you should record the documentary evidence of that activity being completed. Some of the entries may be optional, depending on earlier answers. Once all the required smaller questions, requirements, and tasks are addressed, you will have your answer to the main question, and can move on to the next of these 5 questions.

Question/Task/Requirement	Evidence	Done
Example requirement 1	Example Risk Assessment.pdf, Example document.pdf	Yes
Example requirement 2	•	•
…	•	•

The 5 Questions

Should I use AI/ML?

Developing an AI application can present significant challenges. Collection of data, testing and validation are challenges. As seen in the previous section, many applications of AI will come with special requirements that can be a challenge in themselves. To address this, the very first step in this guide is to be able answer the following: should I use AI to solve my problem? We break this problem down into two parts:

- What is the engineering case for using AI to solve the problem, over other approaches?
- Does the problem touch on any application or data areas that may effectively prohibit development?
    - What is the risk that the project falls into the unacceptable AI Risk category?
    - What is the risk that the project makes use of problematic data?

Requirement	Evidence	Complete
Evaluate Engineering Case	•	
Unacceptable AI Risk	•	
Problematic Data Risk	•	

How do I define my AI project?

In the previous step we worked on an engineering case for AI in our project. In this step, we will take the first step in realizing our project by setting the goals and bounds of the project. There are three steps to complete in this section:

- Defining goals and metrics for success
- Defining limitations and boundaries on the project
- Completing a risk assessment of the project as a whole

Requirement	Evidence	Complete
Establish Goals and KPIs	•	
Risk Assessment	•	

How do I collect data for my AI/ML project?

In the previous step, we defined the scope of our project. In this step, we move on to the first part of the practical engineering of our AI project: collecting the data. AI/ML applications are, at their core, data driven. At some level, data will need to be collected, or at the very least, processed. It's very important to get this right, as the strength of the data will have a strong effect on the efficacy of training and deploying our AI application. We set out a number of steps for this section, but our primary challenges are:

- Making sure the data we're is collecting useful, truthful, and effective data
- Making sure we transform our raw data into a form that can effectively utilized by AI/ML algorithms
- Making sure our infrastructure for collection, storage, and access is appropriate and robust

Requirement	Evidence	Complete
Collecting your Data Set	•	
Version Control, CI/CD for Data	•	
Documentation	•	
Logging	•	
Data Exploration	•	
Data Cleaning	•	
Validation and Testing	•	
Data Storage and Access	•	

How do I train my AI/ML application?

In the previous step, we collected the data for our AI project. In this step, we will make use of it by using it to train an AI algorithm of our choice to meet the goals of our project. This is also the step where systematic problems from decisions in earlier steps are likely to start manifesting in force. We strongly suggest that readers don't hesitate to revisit earlier decisions at this stage if they prove to be unfruitful. Once again, we set out a number of steps for this section, but our primary challenges are:

- Establishing which AI approach we're going to use
- Engineering a pipeline to train our approach in the best possible way
- Building confidence that this training results in an AI algorithm that does all the things it should, and none of the things it shouldn't

Requirement	Evidence	Complete
Selecting an AI/ML Approach	•	
Data Pre-processing	•	
Creating a Training Pipeline	•	
Testing, Validation & Biases in Training	•	
Version Control, CI/CD	•	
Documentation and Logging	•	

How do I deploy my AI application?

After training our AI application, we can finally deploy it and (hopefully) achieve the goals set out in our previous steps. This step will likely represent a welcome return to familiarity for the professional engineer, as the process for deploying an AI application is fairly similar to that of deploying any other software application. Our process will proceed broadly in three steps:

- Preparing our trained the model for a live environment
- Engineering a process for deployment and model updating
- Setting up continuous monitoring for our model

Requirement	Evidence	Complete
Deploying Your Model	•	
Testing, Validation & Biases in Deployment	•	

Key Definitions

AI Risk

Direct regulation of AI in the EU is tending to a risk based approach based on the harm than an AI application poses to people. Applications are classified into risk categories, and those that pose a high risk are to be subject to more stringent requirements. These categories are currently:

- Unacceptable risk. Applications that are a clear threat to the safety, livelihoods or rights of people.
- High risk. Applications that are critical to the safety or wellbeing of people, that require special mitigations to adequately cover the risks they pose
- Limited risk. Applications with lower impacts to safety or wellbeing, but which still require some level of mitigation
- Minimal or no risk. Applications with minimal to zero risk.

Providing guidance on any upcoming or current legislation is out of the scope of this guide. However, this categorization of the risk that AI applications pose is a useful tool for discussing best engineering practices. These risk levels will recur in this guide, and we will refer to them as AI Risk.

Data Protection

AI models are data driven. This leaves them significantly subject to data protection regulation, especially GDPR. According to GDPR, if you process data, you have to do so according to seven protection and accountability principles:

- Lawfulness, fairness and transparency — Processing must be lawful, fair, and transparent to the data subject.
- Purpose limitation — You must process data for the legitimate purposes specified explicitly to the data subject when you collected it.
- Data minimization — You should collect and process only as much data as absolutely necessary for the purposes specified.
- Accuracy — You must keep personal data accurate and up to date.
- Storage limitation — You may only store personally identifying data for as long as necessary for the specified purpose.
- Integrity and confidentiality — Processing must be done in such a way as to ensure appropriate security, integrity, and confidentiality (e.g. by using encryption).
- Accountability — The data controller is responsible for being able to demonstrate GDPR compliance with all of these principles.

There are also certain categories of data that must be given special consideration. Providing guidance on legislation remains out of the scope of this guide. However, once again, the ideas put forward in GDPR are important in framing the discussion of best engineering practices, and will recur in this guide.

Safety, Security and Robustness

A core requirement of any system is that it functions correctly, both in conditions of normal use, as well as conditions of misuse (intentional or otherwise), or other adverse conditions. An AI system should be:

- Safe: either functioning correctly or failing safely in all conditions
- Secure: resilient against misuse or abuse (intentionally or otherwise), and
- Robust: ensuring functionality in as wide a range of conditions as possible

While the need to meet these challenges is not unique to AI systems, AI systems bring several unique problems to each of these areas. For example:

- Safety is a continual challenge in AI systems that are fundamentally statistical. Guaranteeing safe behavior in the wide range of situations an automotive AI may encounter, for example, remains a continuous challenge.
- AI models present unique attack surfaces that must be secured. For example, in many situations, models will leak information about the data that they were trained on.
- Similarly, AI models struggle to robustly deal with novel data outside of their training set. This can both cause problems in normal use, and be exploited by attackers.

Safety, security and robustness will be important ideas throughout this document.

Transparency and Explainability

It is important in many cases that we can understand how our systems function. This imperative should be familiar to any electronic systems engineer, through the value of clear code and documentation. We use two terms to describe these requirements:

- Transparency: the communication of appropriate information about an AI system to relevant people (for example, information on how, when, and for which purposes an AI system is being used), and
- Explainability: the extent to which it is possible for relevant parties to access, interpret and understand the decision-making processes of an AI system

Many AI systems are uniquely troublesome in these respects. Many common methods, such as Neural Networks are effectively “black boxes”. These methods provide a solution, but through a means that is ultimately not human interpretable.

Fairness

Fairness is another core requirement of any system, especially in light of the above ideas of transparency and explainability. By fairness, we mean a system that:

- Does not undermine the legal rights of individuals or organizations, discriminate unfairly against individuals or create unfair market outcomes.

Fairness is a challenge in AI systems that learn from data. The decisions these systems make are a reflection of the patterns in the data they are trained on. If the data is biased, the systems trained on them will also be biased. This has been proven expensively and at length in the real world, for example through attempts at creating criminal justice AI or hiring AI.

Accountability and Governance

Any system that has the potential to cause harm requires oversight. Any AI system will therefore require systems of:

- Governance, a framework for managing the development, deployment and operation of AI
- Accountability, clear lines of responsibility for all aspects of the AI applications

The full scope of requirements is unlikely to be fully addressed by engineering. Nonetheless, these principles are important in the greater context of the AI application, and will be an important part of development.

Contestability and Redress

Any system that has the potential to cause harm to people also requires ways for this harm to be recognised and reversed. Any such AI system will therefore require systems of:

- Contestability, those who may be adversely affected must have a route to contest decisions or outcomes
- Redress, there must be a system in place to assess these concerns, and redress the affected party when necessary

The full scope of requirements is unlikely to be fully addressed by engineering. Nonetheless, these principles are important in the greater context of the AI application, and will be an important part of development.

Should I use AI/ML: Task List

Should I use AI/ML?

Evaluate Engineering Case

In this guide, we assume you have a broad end goal in mind to achieve with AI and Machine Learning. Through this guide, we will take you through the process of converting this broad goal to a concrete plan.

As we discussed previously, while AI is a powerful tool in many applications it may not be immediately appropriate for the problem you are trying to solve. This step requires you to enumerate and assess what available alternative approaches to solving the problem, and how they compare to an AI solution. This isn't intended as a deep dive into the details and risks of your prospective systems, but a high level check that this engineering approach is right for you.

At this stage, you may not have a complete and full understanding of the scope and limitations of an AI and Machine Learning approach. However, the design of this guide is such that you should expect to revisit earlier steps (like this one) from later steps as your understanding improves. You should therefore fill this out to the best of your ability, then return to this later as the project and your understanding evolve.

This step is fairly short, and to complete it, you simply need to list the viable approaches to solving your problem (including the AI approach), and compare them. This should include solutions that might not take a data driven approach, as well as data driven approaches that don't use learning, for example, heuristic based approaches. The choice of how to compare these will be dictated by your use case, but you're likely to want to consider the following:

- What are the financial burdens of each case?
- What are the engineering burdens of each case?
- What are the legal burdens?

Examples

In this example, we work through the engineering case for a phishing email detector.

Our use case is a filter to reduce the success rate of phishing emails. As an example of how we might think about this, we consider three alternative internal solutions:

- A non data driven (or minimally data driven) approach. All emails that are from external email addresses automatically have a header attached warning employees that the sender may not be trustworthy, and to apply caution in respect of the contents.
- A data driven approach that uses engineer designed heuristics instead of a learning approach. For example, emails are filtered if they:
    - Are from a list of known "bad" addresses
    - Contain too many known "bad" words, e.g. "low interest rate loans"
    - Has ".exe" attachments
- An AI approach that learns from a corpus of previous emails to read an emails content, and classifies it as phishing, or not phishing.

The respective advantages of each approach are:

- The non data driven approach is simple to implement, requires few resources to execute, and is simple and cheap to maintain. It will never incorrectly filter out legitimate emails as it passes the burden of decision making to employees (with extra information).

- The heuristic approach is also fairly simple to implement, and requires few resources to execute. It is also very easy to tune for specific requirements (e.g. never do this, always do this).
- The AI approach will likely have the best performance at identifying spam emails, if trained on sufficient data. It is also likely to adapt fairly well to changes in phishing approaches over time, if continually fed with data.

The disadvantages of each are:

- The non-data driven approach provides extra information, but the decision making ultimately places responsibility on the end user.
- The heuristic approach requires each rule to be created manually, scaling poorly to large scale problems. Creating effective heuristic rules that do not result in any real emails being filtered out is also likely to be challenging. Adapting to new phishing approaches is also likely to be a significant burden, requiring new rules to be written or old rules to be rewritten.
- The AI approach will need a large corpus of data to learn from which will have to be prepared appropriately. It will flag real emails incorrectly some of the time (no matter how well trained), and this may not be easy to fix. Adoption will require more data preparation and training.

Picking the best approach (or approaches) to use depends on our context. In this particular example, our non-AI examples are significantly simpler to implement, and are likely to do better for small scale problems. Other conditions may also influence the decision. An organization of phishing-aware staff that largely communicates internally may hugely benefit from the first approach.

Unacceptable AI Risk

In the Using This Guide section, we defined AI risk in several categories: Unacceptable, High, Limited, and Minimal. Before continuing any further in this guide, it is important to consider the risk that your project may fall into the Unacceptable category. These are applications that are a clear threat to the safety, livelihoods or rights of people, and are not suitable to be solved with AI. Examples include:

- AI systems using subliminal techniques, or manipulative or deceptive techniques to distort behavior
- AI systems exploiting vulnerabilities of individuals or specific groups
- Biometric categorization systems based on sensitive attributes or characteristics
- AI systems used for social scoring or evaluating trustworthiness
- AI systems used for risk assessments predicting criminal or administrative offenses
- AI systems creating or expanding facial recognition databases through untargeted scraping
- AI systems inferring emotions in law enforcement, border management, the workplace, and education

At this stage, you may not have fully fleshed out the scope of your application. Nonetheless, this initial assessment is important to pre-empt this risk. If you believe your application may fall into this category, discontinue further work on it until you have resolved this issue.

To complete this step, complete a risk assessment on the harms your AI project may pose to the safety, livelihoods or rights of people.

Examples

Problematic Data Risk

Some types of data are subject to extra difficulties that will either require extra licensing, oversight, or may be effectively impracticable to collect. This section is about evaluating the risk that the data you are likely to wish to collect is available within the constraints of your business. Note that this section *is not* a dive into data collection requirements under GDPR, but a higher level feasibility check. Some examples of types of data that will be problematic:

- Criminal Conviction Data, only processable:
  - under the control of official authority; or
  - authorized by domestic law.
- Data collected by experimenting on humans or animals

    - requiring extra licensing and oversight
- Data surrounding experimentation with infectious diseases

The specifics of problematic data will depend on the domain the data is being collected in. At this stage, you may not have fully fleshed out the scope of your application. Nonetheless, you should, before proceeding further, assess the risk that this applies to you. If you believe your application may fall into this category, you should once again discontinue further work on it until you have resolved this issue.

To complete this step, complete a risk assessment on potential problems surrounding the collection of the type of data you are likely to require.

Examples

How do I define my AI project: Task List

How do I define my AI project?

Establish Goals and KPIs

In the previous section we established, at a high level, the viability of the project we intend to undertake. In this section we refine the proposed project into a set of concrete engineering goals. To do this, we open this section by setting out a little bit more about how AI approaches can be used to solve problems. We then move through a standard goal setting approach (making sure goals are Specific, Measurable, etc.), and discuss some of the challenges that AI approaches can bring for this.

As we discussed, the scope of this guide is such that one can expect to revisit earlier steps while evaluating later ones. This step in particular, setting a concrete definition for the project is one that is likely to cause us to re-evaluate the checks we made in the previous steps. This is fine, and a natural part of using the guide. Our previous effort is not wasted, but informs our current and future decisions.

We can break AI approaches into 3 categories. We provide a more detailed explanation of these terms in the [appendix]. However, briefly:

- In supervised learning approaches we are learning by example. We have a set of input data points with corresponding known output data points (labels) for each. We're trying to learn what this relationship is so that we can predict the outputs that correspond to the inputs for unknown data points. For example:
    - Learning to predict stock prices from economic indicators
    - Learning to translate one language into another
    - Learning the relationship between images chest x-rays and the presence of cancer
- In unsupervised learning we are learning patterns. We have a set of input data points without any corresponding output data points. Our goal isn't to learn an input-output relationship (because there isn't one), but to learn things about the input data points that will generalize to all possible input data points. For example:
    - Learning to identify clusters of like behavior
    - Learning to identify outliers
    - Dimensionality reduction
- In reinforcement learning approaches, we are learning by example with an algorithm that interacts with its environment to select the data points it wants to learn from. Our goal is to create a policy from which tells the algorithm the best action to take at any point. For example:
    - Learning to play chess
    - Learning to play atari video games
    - Autonomous vehicles

Previously, we assumed that you entered this guide with a broad project goal in mind. With the above in mind, the goal of this section is to convert that idea into:

- A set of project goals, defining:

    - What will define when the project will be complete
    - What, specifically it needs to achieve
    - When this should be completed by
- A set of KPIs, defining:
    - How well the project is currently performing on some important benchmarks

We will not cover again the work of general goal setting here, which we assume should be familiar to any competent engineer. However, over other engineering projects AI and Machine learning projects have several uncommon considerations.

The first is that AI and Machine Learning algorithms are often stochastic in nature, behaving in a non-deterministic and statistical way. All goals and KPIs must be aware of this. For example, for a face detection software it is appropriate to set goals such as “95% of this set of faces can be recognised” rather than “the software must recognise all these specific people”.

The second is that human interpretability is a difficult topic in AI and Machine Learning, and it may be difficult to define why the software is doing the things or making the decisions that it makes. We will cover this topic (and its mitigations in more detail) in later sections, but it is important to be aware that metrics that rely on a deep understanding of functionality may be difficult to resolve at later stages.

To complete this step:

- Define a set of Goals and KPIs

Risk Assessment

In the last section, we did a preliminary assessment of some high level project sinking risks to establish that our project was viable. In this section we do a more detailed analysis of the risks of our proposed project.

Risk assessment is a basic engineering skill and we will not cover the basics of performing a risk assessment here. There are a multitude of resources available on how to do this. In future versions of this document, this section we discuss some common risks in more detail.

How do I collect data for my AI/ML project: Task List

How do I collect data for my AI/ML project?

Collecting your Data Set

Defining the Plan

The first step in creating our AI application is to create and (with caveats) implement a plan to collect a dataset to drive your AI application. The plan will include:

- What data you are going to collect
- Where/whom you are going to collect it from
- How you are going to do this

The best way to initially approach this is to approach it as you would any novel software problem: do not reinvent the wheel and never build anything yourself that you could fairly appropriate from somebody else. There are many free datasets for a wide range of problems publicly available. Observe what types of data others who are solving problems similar to you have collected, and what you can learn about the datasets they used. It may be appropriate in the first instance, and if a suitable dataset exists, to initially use a public dataset and iterate. If you do use other datasets, do make sure you respect the licenses that may come with them.

In most business cases, you will at some point end up collecting your own data. Even if you don’t, it is important to be aware of what kind of data is desirable for AI and machine learning, and what kind of data is not. When looking at potential data, some key criteria to consider are:

- Accuracy
    - Does the data accurately measure a quantity you are interested in?

    - Not all data can be trusted. Data from questioning human participants for example, can be inaccurate and contradictory.
- Completeness
    - Does the dataset represent a complete view of all data points of interest?
    - Does it have more data about some quantities than others? Should it?
        - Your models cannot learn from examples that are not in the data
- Relevance
    - To what extent is the collected data relevant to the measure of interest?
    - Including data that is only weakly relevant may cause more problems than it solves
- Missingness
    - Are there missing values in the data?
    - Distinct from completeness. Completeness is about overall coverage, missingness is about which bits of your collected data are not present.
- Timeliness
    - Is the data still relevant now?
- Subjectivity
    - AI methods are fundamentally quantitative, and deal best with quantitative data
- Attainability
    - Can the data be realistically obtained (and in the quantities required)?
- Standardization
    - Is the data collectable/attainable in a standardized format amenable to computation

What data you intend to collect is likely to be very tightly tied to where you collect your data. The best source of data is usually the source that gives the best data by the criteria we list above. This is not always the only consideration though, it is also wise to consider:

- Licensing. This applies both if you're using an existing dataset licensed by a third party (even a free one), or if your data might contain licensed work. As an example of the latter building a dataset of artwork may require you to consider the licenses of those artworks.
- Personal Data: classes of data (e.g., personal data) must be treated specially. More on this at the bottom of this section
- Special Cases: Depending on the data and end goal, you may be required to take additional steps in data collection. For example, data collected by experimenting on animals is likely to require extra licenses and oversight.

Implementing the Plan

We discussed supervised and unsupervised learning in the Establish Goals and KPIs section. If you are dealing with a supervised learning problem (as is likely), the largest concern of data collection is how the data can be labeled. In a supervised learning application, we want to learn to predict some quantity from our data. To do that, we need examples which match our data and that quantity together. For example if we want an AI application that detects spam, we need to collect as data a set of emails, and divide them up into two categories - spam or not.

Fundamentally, you have two choices of how to do this. Firstly, you can contrive a way to achieve this automatically. If you are predicting how sales from your website occur based on how people engage with it, it might be a fairly simple affair for you to match these two bits of information up. Otherwise, if you can not contrive a way to do otherwise, the data must be labeled manually, by hand. For example, in a dataset of pictures of animals, the only way to effectively know what animal is present in the picture is to get a human to decide. In general, we wish to avoid this - for the purposes of this type of task, humans are expensive, prone to error and hard to scale.

Other than this, the process of how you will collect your data is simply the practical realization of what you have set out in the previous steps. Your focus here is making the process as simple and replicable as possible. As a general rule, the more automated the process can be, the better. Automated collection processes scale better, and involving

human factors in the collection process is usually an excellent way to introduce an extra set of errors. Automated data collection is not possible in every application though. If you do have to have people involved in your data collection process, try and work as hard as you can to maximize the consistency of the process for them.

It is worth saying that regardless of the initial choices you make, it is likely you will revisit this step as you follow the other instructions and find out what works for you and what does not. This is perfectly fine, and it is much better initially to pick a dataset, make a start, and iterate, rather than trying to get it perfect the first time.

The criteria to complete this step are:

- To create and implement (with exceptions, see below) a Data Collection Plan. This is a plan that details:
    - What data you are going to collect
    - Where/who you are going to collect it from
    - How you are going to collect the data
    - Any extra considerations

What data you are going to collect should include a data blueprint, a sample of exactly what you think the data you are going to collect should look like. The where/who should include specific populations you can feasibly target. The How should be a plan of action to collect the data from the first step from the groups you defined in the second step including, where required, a plan for how the data is to be labeled.

If your AI application is in the high AI Risk category, or you are dealing with personal data, some special considerations exist. Create this plan, but do not implement it until you have worked through the following:

For high AI Risk applications:

- You will be required to provide adequate documentation around data collection, and log all data collection activity. You will also be required to have human oversight over the data collection process. Make sure you visit the Documentation and Logging section of the guide that covers these areas before collecting any data
- You have a duty to make sure the data you collect is of a high quality to minimize discriminatory outcomes. Make sure you visit the Data Exploration and Biases section of the guide before collecting data.

For AI applications dealing with personal data:

- You must collect data according to GDPR regulation. This topic is expanded on in our Appendix on GDPR.

Examples

In this example, we work through the case of an oncologist looking to create an AI application to help other physicians detect the presence of tumors in a chest x-ray. There is an *example data set creation document<dataset_creation>*, and a description below:

The data I, as our imaginary oncologist, will need for my application is a set of chest x-rays, and whether they contain cancer or not.

Data	classification
x-ray0.png	cancerous
x-ray1.png	non-cancerous
x-ray2.png	cancerous
…	…

In respect to where I can find the data, a starting point is obviously the data that I can collect from my own patients. I may be able to get data from other patients from people in my professional network, or simply search online (there are several publicly available datasets on this particular topic).

In respect of how I can go about collecting (and labeling) my data. I can get chest X-rays from the sources described above. To label them, I can use my own expert knowledge, and/or ask other physicians to also contribute to corroborate.

In respect of extra considerations: I am working with personal health data. I'd likely need to obtain consent from the patients, must respect GDP, and may have additional requirements to fulfill in respect of my medical license, or an ethics board to satisfy. It's likely that this application would also fall into a high AI Risk category, and be subject to extra requirements.

Version Control, CI/CD for Data

Any electronic systems engineer should be familiar with version control. These ideas are just as important in developing AI applications as any other software product. In this step we explore:

- Version control systems for all data collection code
- Version control systems for all data collected

Our reasons for developing version control for data collection code are the same as they would be for any other software project. We have similar requirements of the datasets we collect with this code. Just like our code, our data is not something we can consider static. Not only is it possible we will collect more, but our existing data may be reorganized, fixed, or updated.

Version control for code is very well established, with a range of standard free tools (e.g. Git, Mercurial, Subversion) available. Version control of data requires a little more work. The standard tools used for code control are only appropriate for a (relatively) small number of small files, tracking a relatively small number of changes. Many datasets will not meet these criteria. In these cases we can either extend existing version control with “large file” control, or with an entirely separate data version control system. Free and Open Source examples of the above are Git LFS and Data Version Control respectively.

To complete this section, you must:

- Set up a version control system for your code and data

Examples

Documentation

Another task that any electronic systems engineer should be familiar with is documentation. As with version control, good documentation is just as important, or perhaps even more so, as any other software development project. In this step we will explore:

- Documentation for the code and;
- Documentation for data

Compared to many software engineering projects, AI projects can often suffer from large variances in behavior due to the stochastic nature of the algorithms involved. A consequence of this is that it is imperative to maintain clear documentation. We must be able to clearly distinguish between acceptable variances in behavior due to irreducible randomness in our processes, and unacceptable variances in behavior due to mistakes.

Documentation should follow standard best practices. This is quite a large topic that has been extensively covered elsewhere (e.g. Write The Docs), that we won’t repeat in this guide. Instead, we devote this section to discussing the additional considerations required for documentation of data. Documentation for data serves not just a similar function to documentation for code in terms of bringing clarity and transparency to what has been done, but it also has a strong role in ensuring reproducibility. In many cases, and especially when dealing with challenging data (such as data involving human factors), how you went about collecting this data is just as important as what you ultimately collected. Ultimately, we suggest that documentation for data should consider the following three things:

What you collected. A good description should (if reasonable) include a way of positively identifying what was collected, where it was collected from, and when.

How you collected it. A good description should both include a succinct high level description of what was done, and include enough detail to allow a full replication. Especially for complicated data collection paradigms, the devil is often in the details. Seemingly unimportant details can become important later.

Why you did it this way. Should both rationalize the process you took and, crucially, why you did this instead of other things. This gives important contextual hints to anyone trying to replicate your results that may be absent from a pure “how” description, and can help them avoid any problems you encountered in the process.

To achieve this we suggest that:

- Each piece of data comes with metadata, describing what it is
- Each group of data should be accompanied by a short document describing how and why it was collected.

This may sound like a significant overhead but, especially if your data is being collected digitally, the burden is not especially high. Populating metadata can often be significantly automated, and written documentation may overlap

significantly with the existence and documentation of relevant code. For example, consider a dataset of images collected by a web scraper. It would be very easy to include a hash to positively identify what was collected, the web location it was collected from, and at what time during the scraping process. In respect of how this data was collected, the code for the webscraper itself provides a strong description of this. Even in respect of why it was done this way, the documentation for this code provides a significant amount of context.

High AI Risk applications:

- For high AI risk applications, documentation is no longer an internally driven process to improve productivity, but a (likely mandated) part of the requirement to demonstrate traceability and auditability of the software
- High risk AI Applications must have human oversight. Automated documentation of data collection will require a level of human oversight and validation.

The criteria to complete this step are:

- Creating documentation for all code written to this point, and standards for future code
- Creating documentation for any data collected to this point, and a process for documenting future data

Examples

Logging

Logging is a core requirement of the software development process. It must be accepted that software will break or not fulfill its function correctly, and when it does we need to be able to diagnose those faults effectively. Furthermore, software is rarely static, and in order to change it we must understand how it works. Finally, in many cases we may wish (or be required) to audit and review our software, and logging is an important part of this. In this step, we will explore:

- How to create a logging process for out data collection

When creating a logging process, the first question is always “What should we log?”. The best place to start with this is to instead start with the question “what questions do we want to answer about our software?”. Obviously, the answer to this will depend on the specifics of our data collection, but some recurring questions you will often need to answer are:

- Where did I get this piece of data?
- When did I get this piece of data?
- What was the state of my collection program when I collected this data?
- Was collecting this piece of data successful?
- Why was collecting this piece of data unsuccessful?

For high AI Risk applications:

- For high AI risk applications, logging is no longer optional. Logging of all actions relevant to proving compliance with the EU AI Act (see appendix) must be undertaken.

The criteria to complete this step are:

- Creating a logging process for all data collection code

Data Exploration

This step is separated from the Data Cleaning step for clarity, but in reality these two steps are likely to be quite closely linked together. In this step we will look at the process of exploring the data we collect.

In our steps so far we have designed an AI application, designed a dataset we think will achieve our goals, and have taken the initial steps to ensure that there is a robust coding framework around this. Before going any further, we need to take a look at the data we have collected and try and understand it’s key characteristics, strengths, and weaknesses of the data to establish:

- An understanding of the the key characteristics, strengths, and weaknesses of the data
- What patterns and relationships exist in the data
- Whether it is likely to be useful for the purpose we intended

- What further data collection should fix, and what it should do more of

We discussed previously in the guide that you may wish to return to earlier steps, and this data exploration step is one of the steps which is likely to encourage this. While this initial examination obviously can not understand the end result of our full AI pipeline ahead of time, we can build up an understanding of our data. It's likely that, especially for the first time, our data may not be exactly as we had hoped it would be.

Our data exploration process is about trying to digest information about our dataset. The methods we use to do this are very fundamental, intuitive ideas:

- Looking directly at the data and subsets thereof
- Trying to understand the data in an intuitive visual way
- Trying to understand the data through summaries and heuristics

The best approach for this will vary, but standard approaches for these are:

- Tabular reports
- Data Visualization
- Data Profiling

Tabular reports. Forming tabular reports is a very simple way to look at our data directly. This is simply structuring our data set in a row/column format. There are no hard rules about how we might want to do this. We could look at subsets of the data, look at ordered columns, or anything else. Just looking at the raw data can often be very useful. We can check our intuitions about the data, identify potential patterns, and notice errors that may be hard to identify other ways.

Data Visualization. Directly examining data is useful in a way that should not be discarded. However, most of us will find it more useful to process data visually. There are a very large number of ways to do this. Edward Tuft lists a series of key ideas that are often cited for this:

- show the data
- induce the viewer to think about the substance rather than about methodology, graphic design, the technology of graphic production, or something else
- avoid distorting what the data has to say
- present many numbers in a small space
- make large data sets coherent
- encourage the eye to compare different pieces of data
- reveal the data at several levels of detail, from a broad overview to the fine structure
- serve a reasonably clear purpose: description, exploration, tabulation, or decoration
- be closely integrated with the statistical and verbal descriptions of a data set.

Good visualizations are often as much of an art as a science. As with many things for which this is the case, the best initial approaches are as follows:

- Start from the basics, the simple tools that everyone else uses (line plot, scatter graphs, heatmaps)
- Unashamedly appropriate good ideas from other people doing similar things
- Explore your own ideas to find out works for you and what does not

Data Profiling. Data profiling tries to capture yet another approach to understanding our data, this time through the use of summarizations and statistics about our data.

- What groupings (or clusters) within in our data
- Averages (mean, median, etc.)
- Spreads (standard deviation, quartiles, etc.)

As part of this we might want to explore how different bits of our data relate to each other. For example:

- Comparing different data
- Correlating different data

Alongside these broad techniques, we might also choose to do a detailed “drilldown” into our data and run a more detailed analysis of some parts. We might look at some more advanced statistics or visualizations of these subsets, or even perform a small scale trial run of some AI approaches we are considering later.

To criteria to complete this step is to:

- Create a data exploration process
- Create a data exploration report

Data Cleaning

The raw data that is collected through our data collection process is not necessarily the best data to use in training our algorithm. Some parts of our data might be poor quality for one reason or another. Following our previous examples, we should look for:

- Accuracy
- Completeness
- Relevance
- Missingness
- Timeliness
- Subjectivity
- Attainability
- Standardization

Hopefully, the data analysis we discussed in the previous step will have uncovered many of these issues already. If there are any issues with the data quality you have not explored, so far, this is the step to do them (using the techniques in the last section).

As with visualizing the data, the best way to clean it will depend on your problem and precisely what it is you’re trying to do with it. However, there are some common problems for which there are well established solutions:

Basic Cleaning. Not all data collection processes perform ideally. Particularly, if many data sources are aggregated together, it’s very possible to see errors such as duplicated data points, or inconsistent labels (“eleven” vs 11, for example). The following is a list of these common errors, and what can be done about each:

- Formatting
    - Convert all labels to one format or representation. For example, all numbers as digits, or all dates as DD/MM/YY.
- Data duplication
    - Check for duplicate entries and remove them.
- Structural issues
    - Not all data sources may have collected the same data. Choose which ones to keep
- Irrelevant data
    - Remove if if you are not going to use it
- Errors
    - Fix or remove any data points with errors (e.g. spelling mistakes from user input)

This is not an exhaustive list, but you should at the minimum consider each of these in your data cleaning.

Missing/corrupt data. A very common problem is for some parts of the data to be missing. For many AI techniques we might want to apply later, this is an issue. The ideal thing to do about this problem is to collect a fresh set of data with less (or ideally zero) parts missing. Obviously, this is not always practical. If we can not do this, then we essentially have two solutions:

- Cut all entries with missing data points out
- Use an AI technique that is robust to the missing data

- Impute our data, filling in estimates of the missing values

We won't cover the techniques for any of these in detail, for which there are a wealth of resources elsewhere (imputation), but we note that the extent to which any of these strategies is plausible depends on how the missingness in our data has come about. There are three ways the data might be missing:

- Missing totally at random (MCAR) - the data is missing completely randomly with no pattern at all
- Missing random (MAR) - the data is missing randomly, but in a way that is explained by the data you are using to predict things
- Missing not at random (MNAR) - the data is missing in a way that is not explained by the data you are using to predict things

As an example of the first, might be some bits of our data being deleted randomly by a bad sensor. An example of the last would be a survey about depression - participants with severe depression are more likely to refuse to complete the survey about depression severity. An example of the second is a little more tricky. Consider a dataset of blood pressure, containing both a group of old participants and young participants. Clearly, data won't be missing entirely at random because older participants are more likely to have their blood pressure measured. However, as long as this is the only difference between the two groups, this is not usually a problem. We can e.g. treat them separately.

The distinction between these can be a bit subtle, but the headline from this is as follows: When data is missing either completely randomly (MCAR), or for a reason we fully understand and can measure (MAR), this is fine. We can use data imputation, or (for some AI models) adapt our model to this situation. When data is missing for a reason we do not understand and/or can not measure (MNAR), we (typically) can not do either and this is not. If you find yourself in this case, typically you need to reformulate the problem. For example, accounting for the fact that the data point is missing as a point of data in itself.

Removing Outliers. Removing outliers is technically simple - there are a range of techniques to do this that we will not re-cover here. The reason we have included this short section is to cover when you should consider removing outliers and when you should not.

Shortly, outliers are data points too. It can be tempting to remove them if they are, for example, making large contributions to an average that you feel is skewing your measure of interest. However, doing this without a valid reason causes problems. Especially, removing data points because they do not agree with your expectations will ultimately lead you to believe them to be true even when they are not true.

The valid reasons to remove an outlier data point are:

- You believe that it is a measurement error
    - It is an outlier because the process user to measure is in error
- You believe it is a data entry or processing error
    - It is an outlier because the way it has been entered into storage or processed is in error
- You believe that your sampling process is incorrect
    - It is an outlier because the way the data point has been chosen over others is in error

A datapoint is a measurement error if the process used to measure it has measured it incorrectly.

To complete this step:

- Clean your data and establish a cleaning process

Validation and Testing

We established a good idea of what qualities our data has (and does not have) from our previous steps. Unfortunately, if and when we collect more of this data, we have no guarantee that things are going to stay the same. Similarly, we want to be sure that in the process of storing and moving the data around, we are not introducing new errors. We therefore want to set up an automated monitoring and testing process to look at the data as we collect it.

Very broadly, we hope with this validation and testing process to be checking two things:

- Does my data look like I expect it to?
- Is my data remaining of a high quality?

- Or at least, is it broken in all the ways I expect it to be and none of the ones I do not

The way we go about this is with a tool that should be familiar to any software engineer: writing tests. The criteria for these tests may have to be a little different from usual. Our data collection process is ultimately stochastic, and so a lot of our tests will have to be stochastic too. Examples include:

- Checking if data falls within the min-max ranges of that data element (based on all previous data registered),
- Defining several validation rules, and display data units violate these validation rules.
- Analysis of data sets, i.e., examining gaps in data, missing values, existing trends, and so forth

To complete this step:

- Create a validation process.

Data Storage and Access

Having collected this data, we need to think about how it is going to be stored and accessed. The solution to this will vary significantly depending on the scale of the operations we are dealing with. Small, lightweight projects might be adequately managed in a spreadsheet, big ones might require a large data warehousing system integrating multiple sources of data.

This section is not intended to be a reintroduction to the technical details of data storage and management, but a review of the problems and considerations around using them in these types of workflows.

Structured vs unstructured storage. When collecting your data, you have the option of storing it in an unstructured form (e.g. in a raw data format), or storing it in a structured format (e.g. a database).

While you are ultimately very likely to need your data to be in a structured format at some point for your analysis, it is not always advantageous to store it in this format.

- On one hand, storing data in a structured format decreases the flexibility you have to store that data, and requires you to commit ahead of time to a data model that involves that format that you choose.
- On the other hand, if you can pre-specify a structure for your data, it will significantly reduce the amount of time and effort we need to spend structuring it later.

The tradeoff between these two will depend on your application. General considerations are:

- How much does storing raw data cost me, versus structuring it?
- How much effort will it be to convert raw data into the structured format I will need later, if I don’t do it on collection?
- What is the realistic opportunity cost for forcing my data collection into a structured format, versus unstructured collection?

Data validation & multiple data sources. The data collection process is rarely perfect, and these problems are often compounded when data is collected from multiple sources, and integrated together later on for analysis. This is one of the causes of many problems that we discussed in the data cleaning section:

- Formatting
    - Convert all labels to one format or representation. For example, all numbers as digits, or all dates as DD/MM/YY.
- Data duplication
    - Check for duplicate entries and remove them.
- Structural issues
    - Not all data sources may have collected the same data. Choose which ones to keep
- Irrelevant data
    - Remove if you are not going to use it
- Errors
    - Fix or remove any data points with errors (e.g. spelling mistakes from user input)

While you should continue to deal with these issues as part of your data cleaning, you should also solve as many of them as you can as part of your data management process. For example, applying a universal enforcement that all numbers should be entered into the system as digits.

Cloud vs Local. It can often be very attractive to pay for someone else to store data with a cloud service. We will not cover the general merits of cloud vs local storage here, which we assume are already known.

The main additional considerations in the contexts of AI and machine workloads are what you are doing to do with the data you have stored. Training is an access intensive process, and how you intend to get the data from wherever it is stored to the computing power doing the storage has a strong impact on performance.

Shortly, whichever of local vs cloud computing you intend to use for training your model, you should consider how this is going to translate into storage.

Performance Requirements. Not all storage is created equal. The target audience of this guide should be well aware that there are a range of physical solutions from magnetic tapes to SSDs for storing and retrieving data that tradeoff cost and performance, with similar cost models for cloud storage systems.

For AI systems, it is worth considering that data that is going to be used directly for training will need to be accessed quickly and repeatedly during the process. In many cases making an investment in high performance storage may speed up the process significantly. However, not all applications will be like this, and not all data will directly be part of the training process. Raw unstructured data, infrequently accessed, for example, may be a prime candidate for lower performance storage.

To complete this step:

- Establish your data storage strategy

How do I train my AI/ML application: Task List

How do I train my AI/ML application?

Selecting an AI/ML Approach

There is not a set way of picking an AI algorithm to use in every case. In the appendix we list how some common algorithms match up to the problems we describe below, but this should not be taken prescriptively. The most practical suggestion in most cases is to use these as suggestions for as a starting point, and iterate. What others are doing with similar problems is another good starting point.

We break up the problems you need to think about when selecting an algorithm into three parts. The first are practical considerations:

- What computational resources do you have available?
- How much data do you have, and how are you going to get it to the computer that you are using to train?
- How many resources (time, money, etc.) are you willing to spend on training?

A lot of the decisions in this area boil down to scaling. Almost any algorithm can be deployed at some scale. The key question is what is the best algorithm that I can deploy for my scale of data and available resources? It is important to remember as part of this that not all algorithms scale equally - either with amount of data or dimensionality.

The second set of considerations are use case based. Especially, what, if any, special requirements do you have for your end system? For example:

- Do the outcomes need to be explained in a human understandable way?
- Do I need to be able to enforce certain behavior in my model?
- Do I need to be able to explain or understand uncertainty in my model?

A lot of the above are common requirements that significantly limit the available choices of algorithms.

Our third and final category are technical considerations:

- How do prospective approaches match up with the underlying assumptions of each algorithm?
- Is the complexity of my prospective algorithm appropriate for the problem I’m solving
- How do limitations of my data affect my choice?

It is important to remember with these points that all AI models are just models, and will never reflect reality in full detail. Nor do we necessarily want them to - as we see in the appendix on Bias and Variance, often a purposely simpler model can produce better results than a more complex one, especially when data or training is limited. Limitations on training data can be important too. Some AI algorithms cannot deal with data that has missing parts, for example. This must either be addressed, or a different approach used.

To complete this section you must:

- Select an AI approach (or approaches) to use in subsequent steps

Data Pre-processing

This step has a lot of overlap with our previous Collecting Data subsection on Data Cleaning. Our focus here is slightly different, with us instead looking at maximizing the efficacy of our training process, rather than specifically focusing on fixing deficiencies in the data.

Missing data. Depending on your choice of AI Algorithm, missing data may have suddenly become extremely important. If this is the case, revisit the Data Collection section, Data Cleaning subsection in the issue.

Scaling and Transforming data. For many models, the training process can be improved by scaling or transforming the data. For example, scaling all values to be on the same scale.

There are a range of reasons you might want to do this. A common example is two features being on different scales. If we have one feature that takes on values in the range 0-1, and another that takes on values in the range 0-100, some types of analysis may cause us to weigh one more heavily than the other.

Which technique will be applied will depend on your analysis and goals. Some common techniques would be:

- Removing the average from a set of numbers
- Transforming all values to the same scale
- Scaling all values to a normal distribution

Dimensionality Reduction. Sometimes our datasets include a large number of features. This can be a blessing because it gives us more information to integrate into our analysis, but it can also cause difficulties:

- It increases computational cost of training
- It can cause overfitting, especially if the extra features are not useful
- Multiple weak features are often a less useful explanation than one or two strong ones

A way of dealing with this is to reduce the dimensionality of the data. There are two very broad ways of approaching this:

- We can select the most useful features from our existing set of features
- We can aggregate our current features to create a new set of features with beneficial features

The advantage of the first is that we can perform dimensionality reduction with the existing features to select the best ones. The advantage of the second is that, although our constructed dimensions lose interpretability, the process by which we create them typically makes them very strong at explaining our data.

To complete this step you must:

- Create a pre-processing procedure for your data

Creating a Training Pipeline

In this step you must create an automated process to ingest the data collected, cleaned and preprocessed, and use it to train a model.

For small projects, this may be a very simple script consisting of a few lines of code collecting the data and running the training function. For larger projects, larger datasets, or non-standard tools this may be significantly harder.

Hardware Acceleration. Many machine workloads can be significantly sped up using hardware that shifts computationally intensive parts of the training process to specialized processing units. These technologies generally work by improving matrix arithmetic. It is not always necessary or desirable to do this, but it can be very beneficial for some workloads. Neural Networks are one example where hardware acceleration has been very successful.

Though there are special processors available for this, modern graphics cards also do this type of processing, are often competitive on cost-per-processing power basis, and have additional uses (e.g. as graphics processors).

Storage Mediums. This is an excellent opportunity to revisit the Collecting Data section, under the subsection of Data Storage and Access. The discussion on different data storage mediums is extremely relevant to training.

Large Datasets & Resource Intensive Projects. Larger projects may end up in a situation in which the dataset is not stored on one machine, and/or computation must be spread across multiple distributed units. Future iterations of this guide will cover this issue, currently we suggest that readers seek expert advice on this complex topic.

To complete this step:

- Create a training pipeline

Testing, Validation & Biases in Training

Testing how the model performs through its training process is extremely important. For models of small scale, it is a vital diagnostic tool. For models of a large scale, it allows us to identify problems early, before too many resources are committed to them.

Training Error. It's important to split your dataset into at least 2 parts - a training dataset and a testing dataset. Most (e.g. 90%) of your data will be in the training dataset, and this will be what you use to do your training. Entirely separately from this

Parameters and Hyperparameters. As important as always. We are essentially testing for extremely similar things as the testing and validation for the data collection, but the downstream versions.

Bias. If there are specific biases of concern in your dataset, you should check these during training. This should be done by creating a test of bias, and periodically running it during the training process.

To complete this step:

- Create an automated testing process

Version Control, CI/CD

A large part of this is discussed in the previous section under "collecting data". The extra considerations for the training process are that we must now firmly keep track of two, nested versionings. We need to keep track of not just the version of the code that was used to train a model, but the version of any data used to train the model too.

Large trained models may not be suitable for normal version control and may require similar techniques (though at a comparatively smaller scale) to data version control.

To complete this step:

- Include your training process within your version control

Documentation and Logging

There is significant overlap between this section and the Documentation and Logging section of the Collecting Data section.

One thing to clearly document in the training process is any hyperparameters associated with the algorithm you are using. These will have a significant effect on the training process. For example, learning rate in a neural network, or the parameters of the kernel in a Support Vector Machine. In addition to these, it is important to keep track of and document all decisions you are making in your training pipeline. Qualities like the batch size of training in a neural network might not be a hyperparameter in a certain narrow sense, but can make almost as much difference to the outcome of training.

Logging remains important. it is important to keep track of how training progressed even in failed testing cases and how well your model is training throughout the process.

To complete this step:

- Create documentation and a logging process

How do I deploy my AI application: Task List

How do I deploy my AI application?

Deploying Your Model

Deployment means taking your trained model from an offline environment to a real world or production system. Exactly what you will need to do for this will depend a lot on what your application is and what you want to do with it. However, some things that you should always consider:

Where you are deploying it (edge, vs centralized). As with training, an important part of realizing your deployment will be formulating a way of getting your data from wherever it is collected, to the deployed model. One of the ways that you can reduce the cost of getting your data to your models is to deploy them in a location near the source of the data they will be ingesting. The advantages of this are:

- Model receives data quickly and can respond quickly, possibly in real time
- Data transfer costs are reduced
- Security burden of transfers is reduced

However, there are some disadvantages:

- Spreading computing power reduces the capacity of each individual unit
- Deploying and maintaining a large number of systems in physically diverse set of locations introduces difficulties
- For security purposes, deploying a large number of nodes introduces a larger attack surface

The calculus of this decision will vary between projects, however we note that large projects often find that the cost of transferring large amounts of data can be very substantial.

How you are deploying it (internally, externally). All of the usual caveats and problems in deploying systems apply. Our product must be prepared for all cases of misuse we can reasonably prepare for, malicious or otherwise.

For AI systems there are several additional complications to consider, both when deploying systems internally or (especially) externally. Unfortunately, a theme of these issues is that they are all currently difficult to completely eliminate. This shouldn't necessarily prevent you from deployment, but might require you to take some extra care.

Out of Sample errors. For some AI systems, making queries that are substantially outside the scope of the training data will lead to nonsensical results. This can be a problem because it is not usually clear to the user what is in scope and what is out of scope. Knowing how much a result can be trusted can be problematic.

An extension from this if the system is exposed externally is that an adversary may attempt to purposely design an input to the system that produces unexpected results. As a real world example consider a research project designing stickers that, when applied to speed signs, cause an onboard computer vision AI to misread the speed limit.

This is difficult to fully resolve, and there is no single solution to this problem currently. If it is of utmost imperative that this behavior is fixed, some possible solutions are:

- Use techniques that can guarantee behavior (e.g. statistical models can bound error rate if their assumptions hold)
- Some techniques (e.g. Bayesian methods) can estimate confidence in solutions.

The difficulty with both of these approaches is that they place substantial restrictions on the overall class of available models. Practically in most cases, the solution needs to be considered at a higher level than the algorithm itself. Whatever process it is part of should consider the fact that solutions cannot be guaranteed to be sensible. Rigorous testing should identify and eliminate as many of these problems as possible.

Training Data/Model Leakage. When somebody queries an AI model, they learn two things. They learn something about how the model understands data, and something about the data it was trained on itself. This is a problem because with enough effort, an attacker can steal both of these things.

Data Leakage. Consider the statistic that 87% of the US population can be uniquely identified by gender, ZIP code and full date of birth. It can take a surprisingly small amount of information to uniquely identify individual points of data. Large, data driven systems like AI are especially vulnerable to this. Some AI systems (notably large language models) have also been known to leak training data when queried directly.

This is a problem that is difficult to fully prevent. There are some techniques that can substantially reduce the threat:

- Removal of identifying variables
- Differential privacy mechanisms
- Private model training techniques
- Use of synthetic data

Again, a problem with these techniques is that they introduce limitations and complexity on the final result. Whether this is worth it depends on the application.

Model Leakage. Querying a model gives information about how it sees the relationship between input and output variables. With enough examples of this, it is possible for an adversary to effectively recover the model. Examples don not necessarily even need to be related to the original training set, though it is beneficial.

There are several approaches that can reduce the impact of this:

- Restricting the number of queries to the model
- Watermarking the model, to identify stolen models
- Adding noise, and Differential Privacy approaches

What you are deploying it on (hardware) This section strongly corresponds to the training pipeline set out in the training section, and will be expanded in future.

To complete this step:

- Deploy your model

Testing, Validation & Biases in Deployment

This section strongly corresponds to the testing and validation set out in the training section, and will be expanded in future.