

# Introduction

One of the most important principles of the [Lean Startup methodology](http://theleanstartup.com/principles) is “fail fast”. No one wants to fail, but failing rapidly makes it less painful, and more importantly, leaves more time and resources to try something new until success.

Since AI projects share this characteristic that [most of them](https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/) will fail, we need to carefully design our AI development methodology in order to remove the biggest failure risks as soon as possible: in other words, we need to quickly know when things are not going to work, and change direction when it’s the case.

But of course, this is not as easy as it sounds. Exactly like in startups, the most impactful AI projects seem impossible at the beginning, and only work as a result of perseverance and (sometimes absurd and obsessive) faith. A good example of that is [Tesla’s controversial choice to remove LiDAR and HD Maps](https://venturebeat.com/2021/07/03/tesla-ai-chief-explains-why-self-driving-cars-dont-need-lidar/) from their autonomous car perception pipeline: no one thought it would be possible at the time this decision was made, but the tremendous progress of AI since then shows that they may well succeed in this project.

What is the right balance between persistence and agility in AI? How can I execute an AI project in a way that:

* Limits as much as possible the risk of failure?
* Makes me quickly realize when I’m going off-track?
* Ensures that if I DO solve the research risk, the rest will follow, and the project will have a big impact?

**Below, I will discuss the 3 most important failure risks that AI projects face (problem fit risk, integration risk, and research risks), and try to give concrete tips and methodology on how to mitigate them.**

# Risk #1: Problem Fit

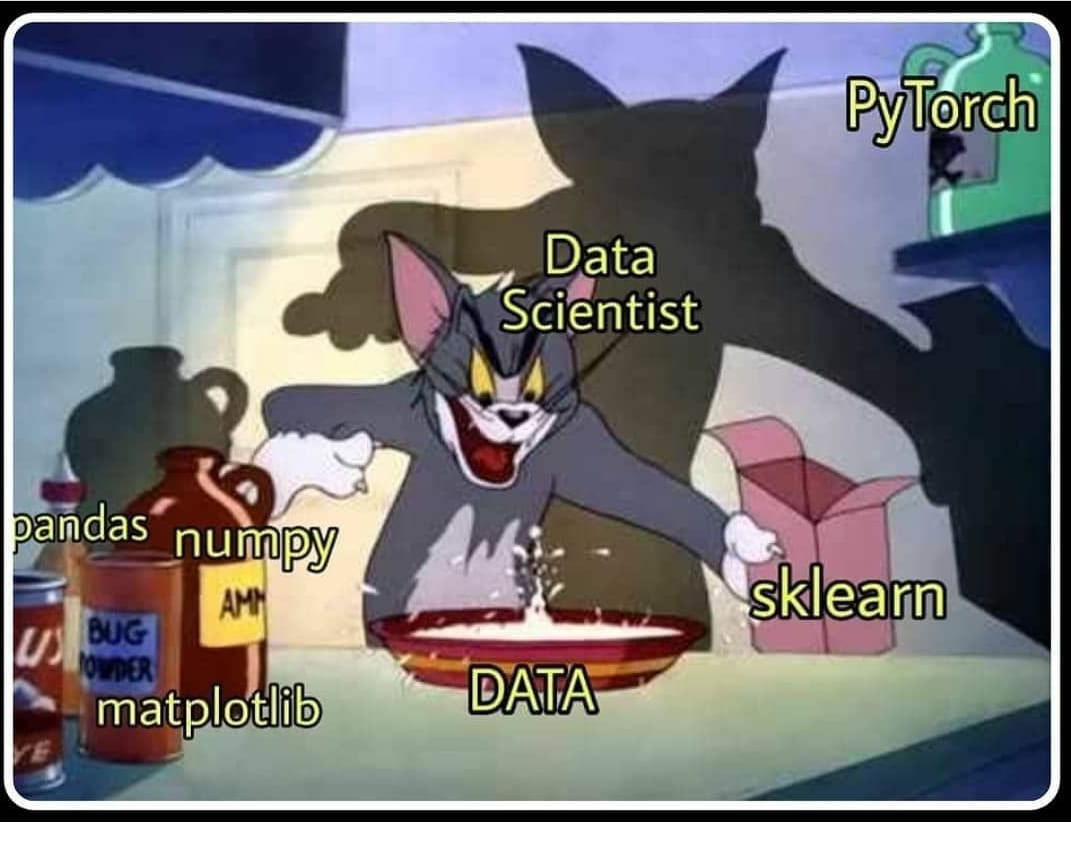
As explained in [my previous post](https://sebderhy.github.io/impactfulai/2021/11/01/Fall-in-love-with-the-problem,-not-the-solution.html), one of the most important risks of failure of an AI project is to build a “[Solution In Search of a Problem](https://www.ycombinator.com/library/8g-how-to-get-startup-ideas)”. It can happen for many reasons: we may have been over-excited by a new paper or AI technique, and somehow managed to convince ourselves that it would solve our problem, or we may just have an incomplete understanding of the problem because we didn’t talk with the right people at the right time.

In my opinion, the easiest way to reduce this risk is to sit with the people who really feel this problem, and really understand their needs. If you can’t easily access your customers, product managers or sales people may be a great proxy to understand what you need to solvem, and the constraints your solution should respect. By doing these conversations, you’ll be able to:

* Validate the need for this project
* Design the ideal solution (detached from any technical constraints at first)
* Decide on the reasonable trade-offs that could be made to reduce the project’s risk while maintaining a reasonable value (in other words, the equivalent of a [Minimum Viable Product](https://en.wikipedia.org/wiki/Minimum_viable_product))

Ideally, the output of these discussions should be **a doc with the project goals, inputs, expected outputs, and constraints**. All these discussions obviously require work and time from the people you’ll discuss with, so you may argue that it’s not practical to require so much time from other people to “help you” with your project at such an early stage. However, I believe that this logic is wrong: if the project solves a real pain point, **they are actually helping themselves, not you**. This time investment actually reduces the AI/problem fit risk much more than the output document itself (it will probably change 30 times anyway). Those people are in general the ones whose interests are the most aligned with the company, so if they think they are wasting their time, this is a **huge red flag that this project is not a good use of your time**! Actually, even if they do accept investing time, pay attention to their body language, and signs of them feeling like they’re wasting their time, it may mean that you are wasting yours too...

# Risk #2: Data Fit

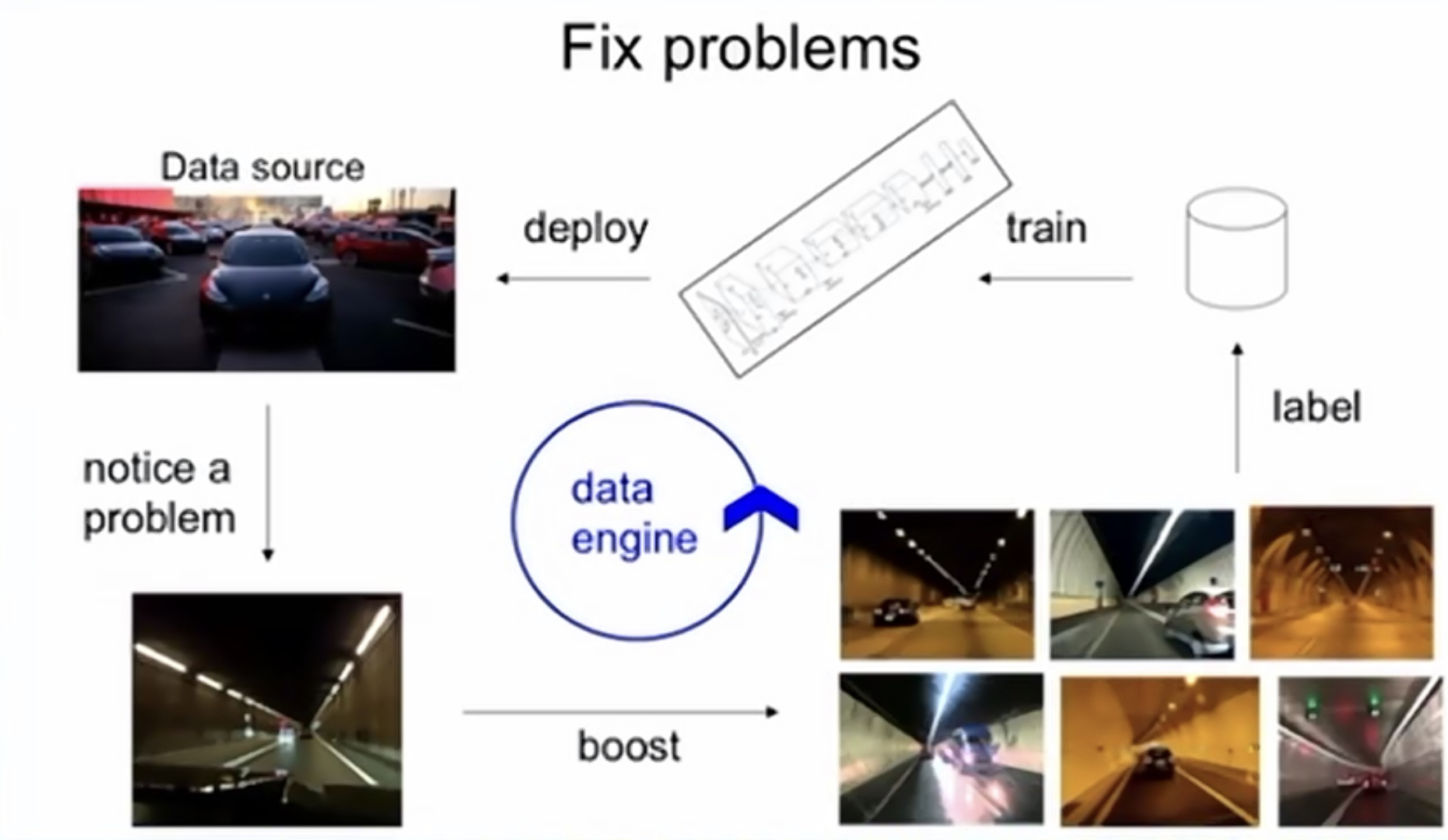


Business leaders sometimes think that in order to succeed with an AI project, you just need to bring many AI scientists together, give them a lot of data, and wait patiently until the results come. However, this strategy makes 2 critical mistakes:

1. Since AI by definition is about making the machine solve a task “by itself”, it is not and should not be a numbers’ game in terms of headcount. Actually, we very often see small “commando” teams winning against large teams from huge companies (as in… guess what… startups!). For example, the competition [DAWNBench by Stanford](https://dawn.cs.stanford.edu/benchmark/), which aimed at finding the state-of-the-art techniques to train neural networks as fast as possible, was won by a small but VERY talented team of [Fastai students](https://www.fast.ai/2018/04/30/dawnbench-fastai/).
2. In data, quantity is not the only thing that matters. Of course, you cannot do much without a decent amount of data, but it’s usually much less than people think (I’ve found that in many cases, ~1000 data points are enough to start getting results). On the other hand, the quality and variance of data is key. In particular, you need to have data that reflects as closely as possible the cases you’ll want to handle in production (including corner cases), but which also contains enough variance so that the network can really learn the patterns that really matter in your data.

We hear all the time that “AI is all about data”, but people often tend to focus on data quantity rather than quality. Of course, you cannot do much without a decent amount of data, but **it’s usually much less than people think** (I’ve found that in many cases, ~1000 data points are enough to start getting results). On the other hand, **data quality and variance is key to success in AI projects**. More specifically, you need to have data (for training, but most importantly for testing) that reflects as closely as possible the cases you want to handle in production (including corner cases), but which also contains enough variance so that the AI model can learn to extract the patterns that really matter in your data.

However, getting the perfect “data fit” for your problem in one iteration is an almost impossible task. This is why many machine learning teams are now adopting a new methodology called “[Data-Centric AI](https://www.forbes.com/sites/forbestechcouncil/2021/12/03/the-onset-of-data-centric-ai-and-why-its-here-to-stay/?sh=554b9e1469cc)”, where they basically try to remove as much as possible the friction to acquiring new data (using for example [synthetic data](https://app.livestorm.co/datagen/implementing-data-centric-methodology-with-synthetic-data?type=detailed)). While the term has been popularized by [Andrew Ng](https://www.forbes.com/sites/gilpress/2021/06/16/andrew-ng-launches-a-campaign-for-data-centric-ai/?sh=3c5bddba74f5), most advanced companies in AI have been practicing this methodology for years now. A fascinating example of such methodology implementation is Tesla and their “[Data Engine](https://gradientdescent.co/t/the-tesla-ai-team-s-data-engine/48)”, which is shown below. Actually, this topic is such a game-changer in the world of AI that I’ll probably dedicate a future blog post about it later.



Source: slide from a [3-years-old talk](https://vimeo.com/274274744) from Andrej Karpathy (Tesla’s Director of AI)

# Risk #3: Integration with Other Teams



## Plant a flag on the product roadmap

For a long time, I’ve thought that AI projects should live disconnected from product roadmaps, because of their inherent risk. Indeed, since we never know when (or if at all) an AI project is going to succeed, how on earth could we put its output on a roadmap that will afterwards be presented to customers, the company’s board, or maybe worse: announced publicly! My conviction was that people should not discuss integration at the beginning and put an AI product on a roadmap until the project has removed all of its execution risks.

The problem with that approach is that once the algorithmic risks (i.e. the proper AI part) are lifted, you still have a very long way to go before seeing the AI in production and delivering actual value:

* Other teams don’t know much about your project, so they will suddenly need to deploy a lot of energy to both understand the problem, the solution that you built, and what you need exactly from them.
* You will also need these teams to change these teams’ plans, since deploying this feature was not initially on their roadmap. For example, you may need software development to deploy it, write tests, a QA team to check that it doesn’t destroy the product, etc… But guess what? They also had plans before you came, and no one likes to change their plans...
* The product team has probably committed to other features while you were developing this AI project, so they also won’t be inclined to delay their current features in order to push yours.
* Worse, since almost no one was aware when your project was going to work (if at all), another team may have developed a work-around in the meantime, that may be much less effective than your solution, but will still be very difficult for you to replace.

It took me a long time to understand it, but I am now quite convinced that I was wrong: **even AI teams should commit to “product” outputs.** Yes, there is a risk of not delivering on your promises. Yes, that risk is high. But 1/ Every one who knows a bit about AI is aware of that risk, and 2/ putting an AI achievement on a roadmap before you know it’s feasible (the famous “**fake it till’ you make it**”) has a lot of advantages that I believe outweighs the risk of not delivering on your promises. Just to name a few:

* It will force other teams to get familiar with your project, collaborate with you on it when needed, and invest time to design it properly with you.
* It will also force them to plan and allocate resources for the moment your AI solution will be ready.
* Perhaps most importantly, it will put a positive pressure on you to **build something end-to-end**. Maybe it won’t work perfectly, and maybe you won’t be super proud of it. But it will hopefully already provide some value to the company, and **have the immense advantage of being fully integrated**. This will make your life much easier later on when you’ll need to justify 2 additional months of work for the next version (the one with the real fancy AI in it) of this feature. We AI applied scientists don’t work well under pressure, but **I believe that the lack of commitment, deadlines and expected deliverables sometimes also hinders us, and it’s time to change that**. To continue my beloved parallel with startups, I have often heard that the best companies [sell their product before it exists](https://thenextweb.com/news/why-you-should-start-selling-your-product-before-it-exists). Why should it be different with AI projects?



## Polish interfaces with other teams

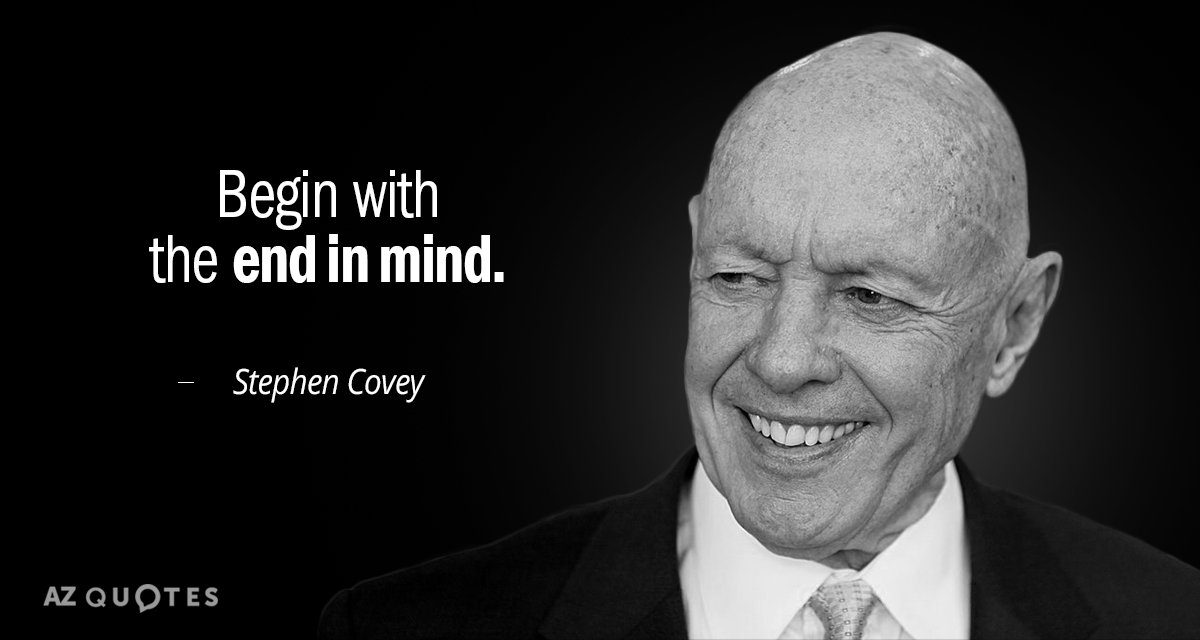
One of the biggest points of friction when pushing an AI model to production is integration with other parts of the system. Suddenly you need to have discussions with other teams about **architecture:** what is the exact format of your inputs/outputs? How will your algorithm be called? By whom? What parts run locally or on the cloud? etc… If you are lucky, these discussions will only generate some “adaptation” work of your algorithm in order to make it look exactly as it should. But if you’re less lucky, these questions may raise significant flaws in your system, and you may have to rethink it entirely. The most obvious case of this is when your algorithm requires too much computational resources, but there are much more subtle cases which could be much harder to foresee.

Therefore, the best way to reduce the integration risks is to have discussions with other teams at the beginning of your project on how this solution could realistically be integrated in the final product, which team should be responsible for what, etc… These discussions will also help the other teams involved in this project plan more precisely the resources needed in the months ahead.

# Risk #4: Research

This is the risk only you can reduce. During the research phase, you’ll need to break your problems into subproblems, and find ways to solve each of them. Unfortunately, both of those tasks could be very hard, and this is where you’ll get to express all your AI and research talent. Exciting and scary at the same time right? Even though there is unfortunately no magic formula here, below are a few tips from my experience to assess the level of risks involved with a problem or subproblem.

## Start with the end

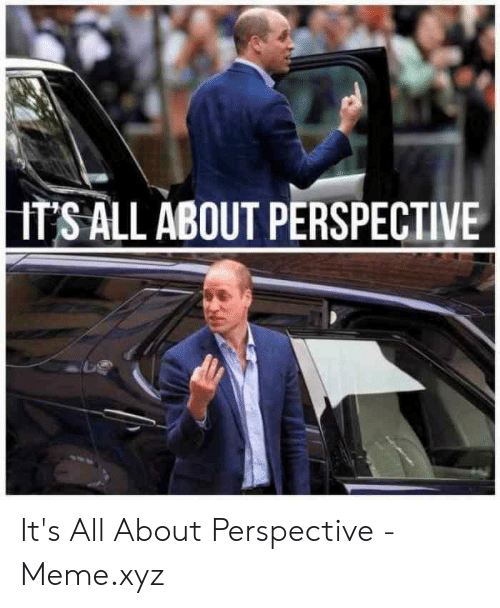


If your AI pipeline is split into several parts, you should start by solving the LAST part, and then progress backwards in the pipeline to solve each block. This may seem counterintuitive, but the reasons behind this principle are actually quite simple:

1. You’ll be able to check early on that your outputs are acceptable in terms of format, quality, etc…
2. You will naturally generate accurate specs for the previous blocks in your pipeline.

For example, let’s say you want to recognize a very small object in images. You may decide to first apply a [super-resolution](https://en.wikipedia.org/wiki/Super-resolution_imaging) algorithm to your input picture, in order to recognize the object more easily afterwards. But how big should the picture be? What is the required quality for this algorithm? These are questions you can only answer by first developing the recognition block.

## Look at things from a different angle



Sometimes, we can solve a problem by looking at it from a different and original perspective, typically by mathematically modeling things differently. For example, we can represent a surface quite naturally as a set of connected points in 3D (this is called a [polygon mesh](https://en.wikipedia.org/wiki/Polygon_mesh)), but we could also represent it as the 0 level set of a very smooth function of the 3D space f(x,y,z) = [distance (potentially signed) of the point (x,y,z) to this surface](https://en.wikipedia.org/wiki/Signed_distance_function). This representation was one of the core ideas behind the paper [Kinect Fusion](https://www.youtube.com/watch?v=KOUSSlKUJ-A), one of the most impactful papers in the world of 3D scanning and localization. This representation inspired the more recent breakthrough paper [NeRF](https://www.youtube.com/watch?v=JuH79E8rdKc) (although their representation is a bit different), which is able to render very realistic new views of a scene from several input views.

## Stage the development with releases

Once you have an initial assessment of the research risks of your project, try to sit again with the product team, and map them into a roadmap of feature releases. Ideally, you’d like to arrive at something like this:

|  | v1 (Minimum Viable) | v2 (creates significant value) | v3 (Super-great amazing holy grail release) |
| --- | --- | --- | --- |
| AI/Market Fit | Required | Required | Required |
| Integration | Required | Required | Required |
| Feature #1  (Research Risk #1) | Required | Required | Required |
| Feature #2  (Research Risk #1) | Not required | Required | Required |
| Feature #3  (Research Risk #1) | Not required | Not required | Required |
| Feature #4  (Research Risk #1) | Not required | Not required | Required |

Make sure you do not have more than 1 strong research risk before you get to a first product release. Indeed, if you have 10% chances to solve a risky problem in general, then you only have 1% chance to solve two risky AI problems in a row, so it’s most likely a non-reasonable investment of your time (unless maybe you find a cure for Cancer at the end!).

# Conclusion: should I pivot or persist?

Throughout this blog post, we’ve seen many tips that help limit as much as possible the risks of failure, and also make it happen as fast as possible if it’s inevitable. However, the hardest question still remains: how do I know when to keep on trying, and when to change course?

Here again, there is no real magic, but we actually have a big advantage here in AI compared to startups: our field is full of mathematics and modelisation. That’s the moment to use them! Try to mathematically model your problem and subproblems, convince yourself theoretically or experimentally that something is feasible, or on the contrary try to find counterexamples when you feel your “proof” gets stuck. Simplify your problem into smaller ones that look solvable, and check that you can actually solve it. Try to break the problem into smaller pieces until you find the core resisting piece(s). Finally analyze (maybe even prove) WHY this piece is resisting, and what was the assumption you made there that turned out wrong. This “debug” process may actually help you find the key to make your AI work!