Hasty

The vision AI blueprint

An actionable approach based on our learnings from working with hundreds of teams implementing vision AI to their offering

Disclaimer: we don't believe in silver bullets. This document won't magically remove all your headaches. But it offers insights based on patterns we saw emerge from working with hundreds of successful teams implementing their vision AI strategy.

“In the past, a lot of S&P 500 CEOs wished they started thinking sooner than they did about their internet strategy. I think five years from now there will be a number of S&P CEOs that will wish they’d started thinking earlier about their AI strategy.”

- Andrew Ng

Hasty

Introduction

Successful AI initiatives are a bit like the Internet in the '90s. Everyone is talking about it, and there are plenty of ideas, but not many are doing it successfully. We're still in the early pioneer days for artificial intelligence.

As with the internet in the '90s, the potential benefits to early adopter organizations are enormous. Many studies show how companies that implement AI successfully at scale outperform companies that don't. From bottom 'til CEO, every manager names it as one of the best options to defend their company's competitive advantage. But only a fraction of companies (studies range from 4 to 40%) implement AI successfully. Mostly, projects blow through budgets and timelines and never make it out of R&D.

Of course, this causes disappointment in the technology itself. Some are already saying that an AI winter is coming because of the disappointing results and that most companies will pull investment into AI. But we disagree.

The problem is not the technology itself but a lack of an understanding of the processes and methodologies needed to deliver projects using the technology. We think that is what makes organizations successful and that everyone should learn from the few companies that already implemented AI successfully.

Since we started Hasty.ai in 2018, we have had the honor of speaking to hundreds of teams about their vision AI projects—from established research institutions like Stanford, over global corporations like Bayer, to small Startups like Pathspot, and even one or two of the FAANG companies. During those conversations, a pattern emerged—a set of characteristics successful teams had and which less successful teams lacked.

Hasty has over 10.000 users from around the world.

Manufacturing

assembly | robotics | quality assurance

Agriculture

weed detection | crop yield estimation | smart machinery

Logistics

package sorting | drone navigation | damage detection

Medical or other

radiology | sports analytics | microscopy

27% corporations

25% startups

23% application builders

21% universities

This document summarizes what we learned from all these conversations and provides a practical guide to succeed with your vision AI initiative.

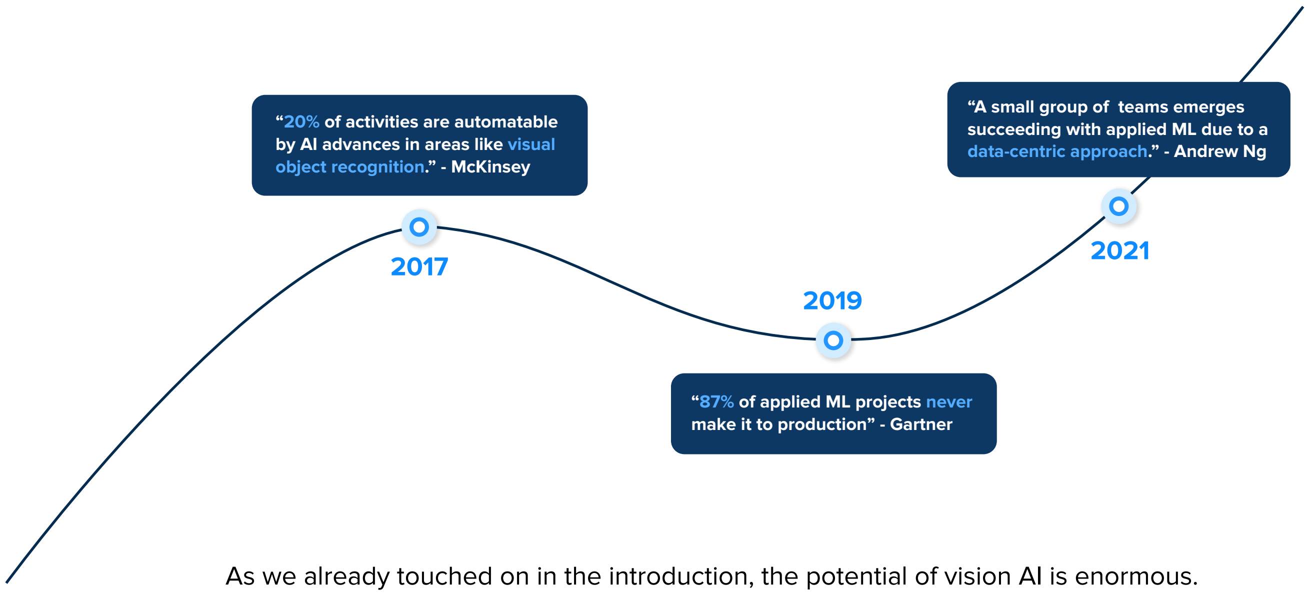
We have a confidentiality agreement with most of our customers, so we anonymized the blueprint without disclosing any company-related information. But above you find an overview of the demographics of our users to understand a bit better whom this blueprint is based on.

The blueprint is for everyone working on applying vision AI to real business use-cases—we try to write it simple enough so that non-technical people can understand it and offer some new thoughts and concepts so that it's relevant for technical folks.

If you have any questions about the blueprint or how to apply the concepts to your project, please reach out to our Strategic AI-Project Lead Tobias Schaffrath Rosario (tobias@hasty.ai). Also, we're always thankful for any feedback, good or bad.

Index

Vision AI is at an inflection point — the technology is finally ready to scale


"20% of activities are automatable by AI advances in areas like visual object recognition." - McKinsey
2017
2019
"87% of applied ML projects never make it to production" - Gartner
"A small group of teams emerges succeeding with applied ML due to a data-centric approach." - Andrew Ng
2021

As we already touched on in the introduction, the potential of vision AI is enormous.

But why is that? What is the promise of AI that makes investors throw billion after billion on AI startups?

To start explaining this, let's think about the current situation for most established companies. What is your most valuable asset as a company today? It, probably, is the quality of your goods. But it's getting easier and easier to produce high-quality goods, and the competition across markets is increasing, resulting in downward price pressure. Where companies used to have 2-4 competitors that could deliver goods of the same quality, they now have 40 - all with the same extremely efficient manufacturing and economies of scale.

So how can a goods company stand out today and deliver more value than its competitor to customers? By offering value-added services that only companies with deep knowledge about the product, their customers, and the market have. Typically, here well-established companies are way ahead of new market entrants.

That brings us back to AI and why it is so promising for so many organizations. AI lets us, if done well, transfer human knowledge into a machine. For companies worldwide, this means that you can take your most valuable asset, put it in a box, and sell it as a new product or service that scales nicely and gives you a competitive advantage.

AI is also tough to reproduce for your lower price competition. They most likely don't have the same knowledge and data as you do. So you can start taking back market share and extend your offering in a way that lower price competition can't match.

Vision AI, in particular, is mighty because it allows you to build products and services that seamlessly interface with the real world without complex integration needs. With vision AI, you can deploy AI solutions into agricultural fields, into mines many thousands of meters

underground, or into drones flying above remote forests - and you can see what your customers see. All you need is a camera and some cheap computation hardware.
Sounds great, doesn't it?

However, so far, the technology did not deliver its initial hype from a few years ago. In the past, most projects trying to apply the technology to real business use cases failed.

Some shrug off the technology itself already. However, from our experience, success is not a question of technology but a question of process. All of the teams we saw succeed had in common: they followed an agile, data-centric approach to ML. And we're not alone with this analysis, but we see more and more established players in the ML world advocating for this. There are even grad courses at Stanford and Berkely teaching agile and data-centric ML by now.

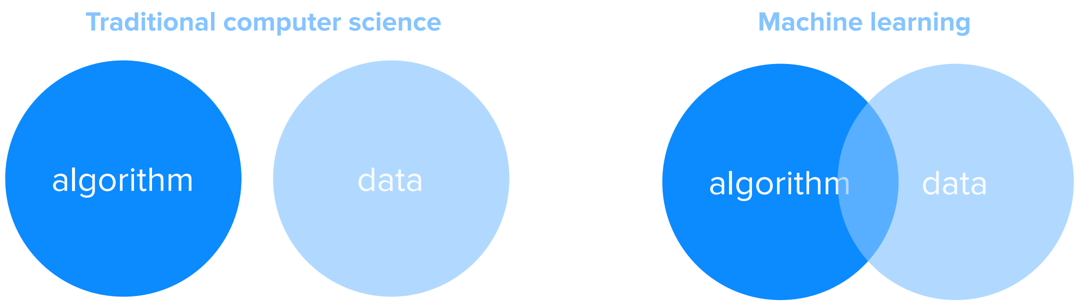
Why do so many projects following the traditional ML approach fail?

Don't worry, we won't give you a long and dry history lesson, but to understand why so many applied ML projects fail, it is essential to know a bit of the history of vision AI at least.

It took the research community over 50 years to figure out the technical problem of finding objects on an image and predicting with high accuracy what this object is. Vison AI was always a computer science problem, one of cracking mathematical and algorithmic riddles.

Once researchers solved the riddle and were able to perform complex tasks like instance segmentation on complex datasets like COCO or ImageNet, people put a checkmark behind the problem and thought the technology would now follow the typical software innovation cycle: practitioners could just implement the logic developed in the early explorative papers.

However, machine learning algorithms differ significantly from traditional software for which innovation adoption cannot work in the same way. Most teams trying to adopt ML like standard software is the single most reason why so many projects still fail today.



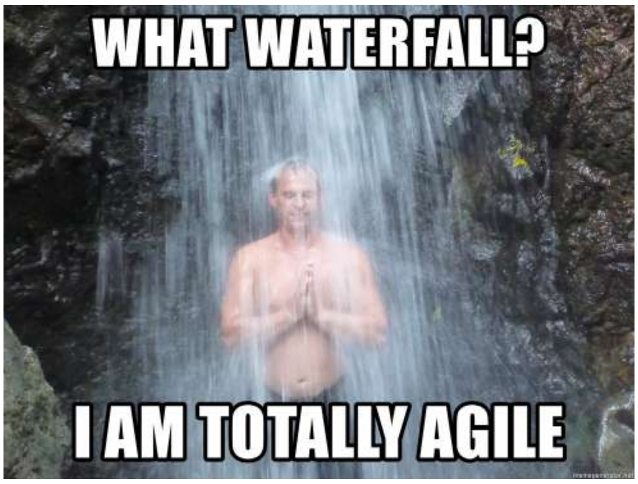
Whereas traditional algorithms aim to be data-agnostic, machine learning algorithms change depending on the data by definition.

Whereas this sounds quite trivial, the consequences of thinking this through are immense:

If you approach ML in the traditional way, you'll automatically end up in a waterfall process because you'll create your data asset first. Then you tackle the "actual" problem of training a model. But, ironically enough, from modern software engineering, we know how inefficient waterfall processes are and that an agile approach generates much more performant results.

Agile ML to the rescue

To be agile in ML, you need to treat the model as a combined product of code and data and therefore develop those two in tandem. As a result, you gather feedback much earlier in the process and always do what yields the highest ROI.



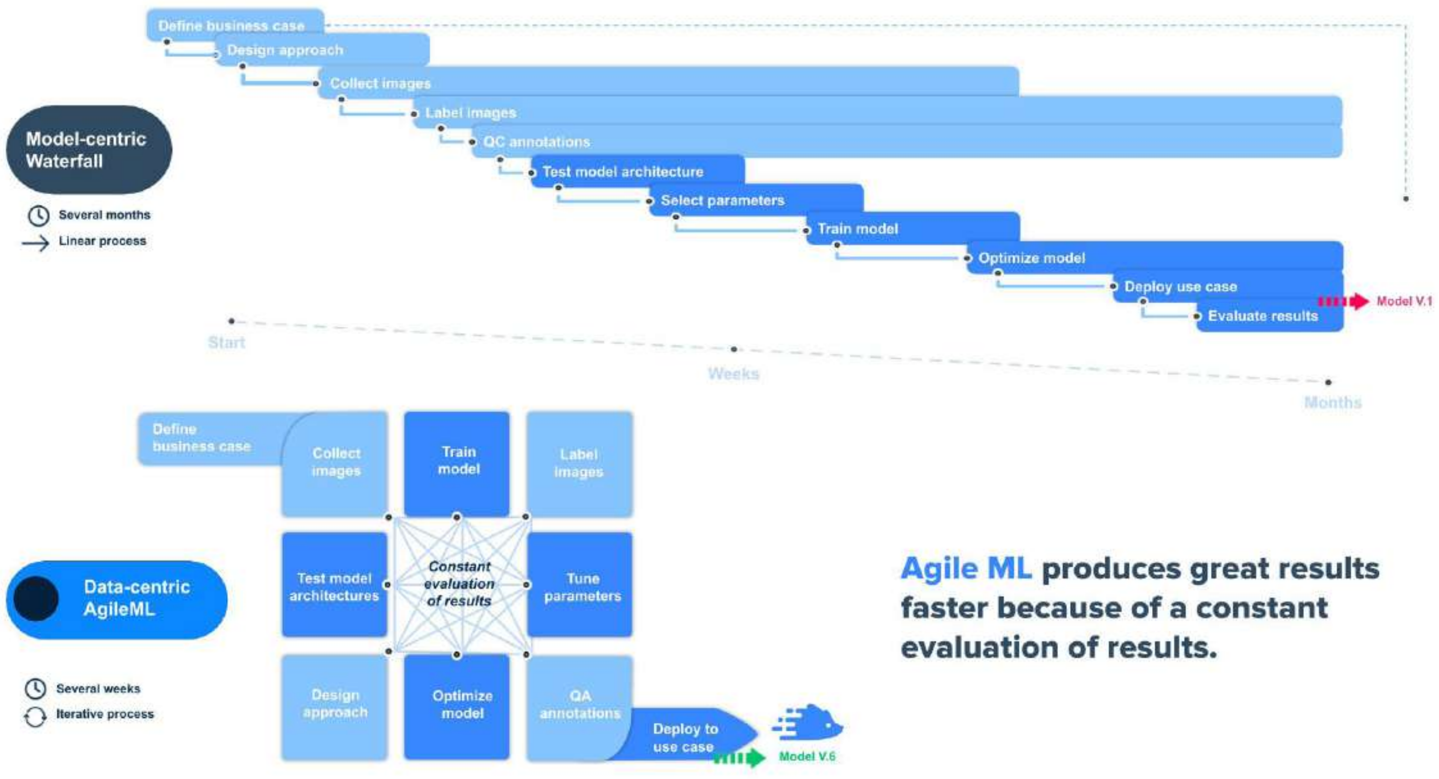
After agile ML has been a niche topic for a long while, we see it slowly but steadily becoming mainstream. This is why we think that vision AI is at an inflection point because the knowledge on how to implement it successfully becomes accessible at a broad scale. More and more tools are available that enable an agile workflow — the agile-evolution that happened in the software engineering world with tools like Gitlab and co is now happening in the ML world.

In the rest of this blueprint, we'll dive deeper into what it means to be agile in vision AI development and how you can follow an agile ML process as well.

We've been there ourselves

We had to learn the hard way that you can't approach ML projects like traditional software. In our life before Hasty.ai, we were trying to build vision AI proofs-of-concept in the German manufacturing industry. All of them failed. We had a technically brilliant team (one of us was on the Kaggle world top100 list, another one on the top200), but the technical challenges were not the ones that took us down. It was challenges imposed by deficient and insufficient data that we couldn't overcome. We tried all tools and approaches we could find, but we were stuck in a slow and expensive waterfall process. So we started to build our own tooling, and after great feedback from our network, we started Hasty.ai.

The agile ML process: get to production more reliably



Agile ML produces great results faster because of a constant evaluation of results.

Ok, now that we talked extensively about why you should do agile ML, let's get into what agile ML is and the prerequisites you need to set yourself up to develop your vision AI application in an agile fashion.

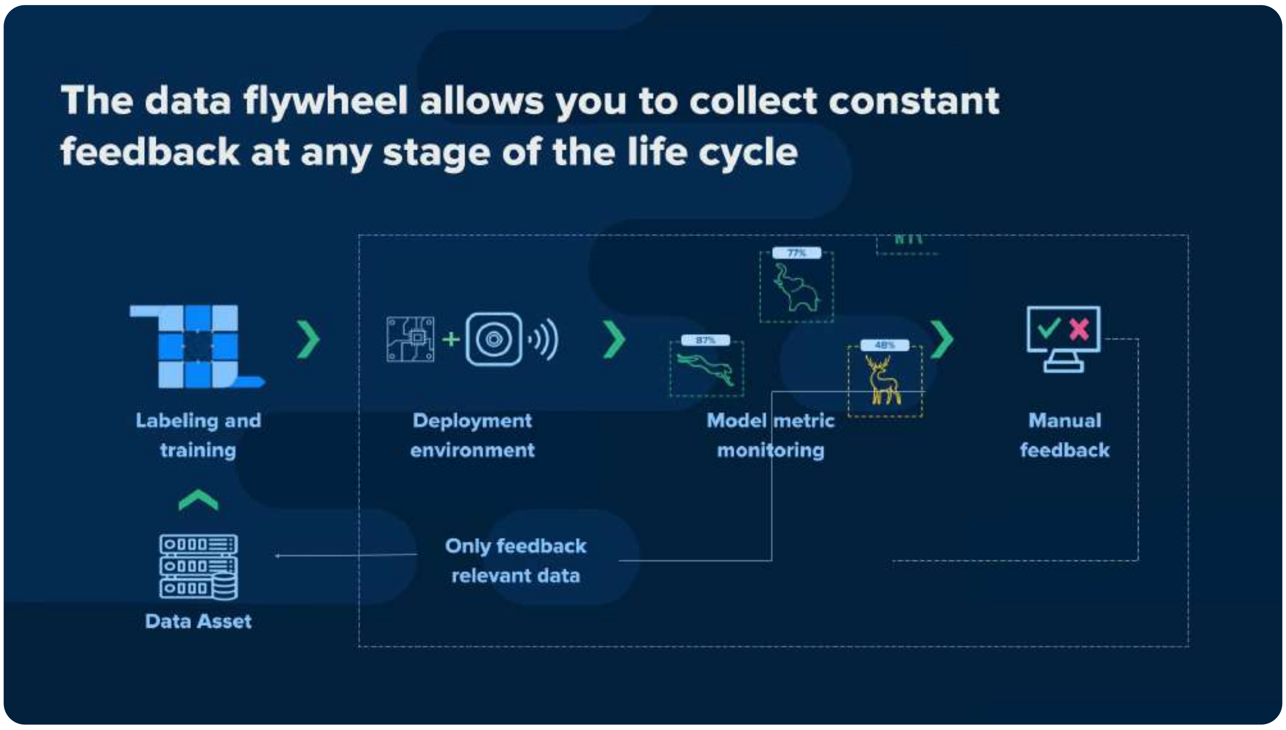
First, we'll walk you through the ideal process; then, we discuss how you can develop your dataset. We look into the tech stack needed. Finally, we talk about the team needed to successfully scale a vision AI project based on our experience working with many practitioners over the years.

The process itself

If you'd follow a perfectly agile process, you wouldn't perform any tasks sequentially but move between seamlessly judging which task yields the best ROI at this point. For example, you wouldn't start with labeling all your data right away but only label until you know that each added image improves the model's robustness. To work in such a manner, you need a mechanism to provide you with constant feedback if you're moving in the right direction, and you need to run through this mechanism fast.

Get constant feedback implementing the data flywheel

It is close to impossible to know upfront which tasks of the ML process yield the highest ROI. But what you can do, is testing different ideas and see what the results are. Teams we saw succeed using agile ML all built some sort of instant feedback loops. We think this idea is so


The data flywheel allows you to collect constant feedback at any stage of the life cycle
Labeling and training
Deployment environment
Model metric monitoring
Manual feedback
Data Asset
Only feedback relevant data
77%
87%
48%

crucial to getting agile ML right, that we even gave it a name: the data flywheel.
The key for the flywheel is that you deploy your first model early on in the process — ideally, while you're still labeling your first batch of data. You then expose the model to real-world data (or at least data close to it) and immediately get feedback about your system's performance. Then you select the images your model is struggling with and annotate those next. After annotating the next batch, you re-train the model, deploy it again, and re-evaluate. Once the performance doesn't increase by adding more images, you know that you can move on to the next step and clean your already existing data. If cleaning data doesn't improve performance anymore, you can finally move to the last stage of fine-tuning your model architecture.

The great thing about the data flywheel is that it doesn't only speed up the development process immensely, but it also provides your ML system a natural way to scale and adapt to new use cases. You can build your end-application around the data flywheel: it'll keep collecting images with which the model struggled from the real world. You then can re-annotate those images and re-train the model, resulting in an ever-adapting system. Mike Tyson once said: "Everyone has a plan until he gets hit in the face." There are no models out there which you only train once and then never change again. You need an adaptive approach that provides you a strategy for reacting to a punch in the face.

Be fast

To benefit from the flywheel to its fullest extent, you need to be able to run through the process quickly. Here, successful teams imitate agile software development again, and they run through the flywheel in sprints. At the beginning of each sprint, they define tasks from which they expect to improve the model's performance. Then, after the sprint, they evaluate if it worked and if they should continue or change the strategy.

The faster your sprints are, the faster you'll produce great results. Or, to put it differently, the quicker your sprints are, the better your results will be in the same time frame. To ensure that you're sprinting fast, we saw teams using mainly two leverages:

1. Automation of manual tasks: the best thing you can do to be faster is to automate as many manual tasks as possible. Especially for annotating data and cleaning the data, many tools offer you different degrees of automation for this task. Spending the time researching which solution fits your needs is definitively worth it as you get this time back very quickly. With Hasty.ai, for example, we saw customers annotate up to 20x faster and do the quality control of their data 35x cheaper than with traditional tools and processes.

2. Minimal integration efforts: to be fast, you can also minimize your team's time to switch between the tasks. Make sure to build your tech stack in a way that your team doesn't need to write thousands of lines of glue-code building integrations. We'll talk more about this in the section about the tech stack.

The data asset (the most critical piece of the puzzle)

Teams that view ML as a mix of data and code, not code only, have a distinguished view of their data asset. They don't view it as a commodity but understand that their data asset defines their competitive advantage — that's especially true for complex use cases.

Data > code

When you apply ML, you're most likely not re-inventing the wheel but use algorithms that researchers developed a few years ago. So the code is public knowledge and replicable. Of course, it is hard to hire the right talent to re-write the code, but it's relatively easy to outsource or to use one of many tools available that offer you neural network architectures out of the box.

Building a dataset is much more challenging, though. If you don't want to create a simple app like classifying dogs and cats, you need to collect custom data first — and typically, only certain businesses have access to use-case-specific data. Then, you need to annotate the data, which can be much harder than most people anticipate for complex use-cases. You often need to be an expert even to understand what you see on an image. All of the knowledge and effort you need to create a usable data asset are hard to replicate for others if they don't know your business as well as you do. So this is why your data asset is your moat and not your algorithms.

You don’t need big data; you need good data
From research, we’re primed to think that the more data we have, the better. However, teams that apply AI successfully prioritize data quality over quantity in our experience — a factor often overlooked in research. Countless experiments were able to show that cleaning insufficient training data improved the model’s performance more than trying to fine-tune the model.

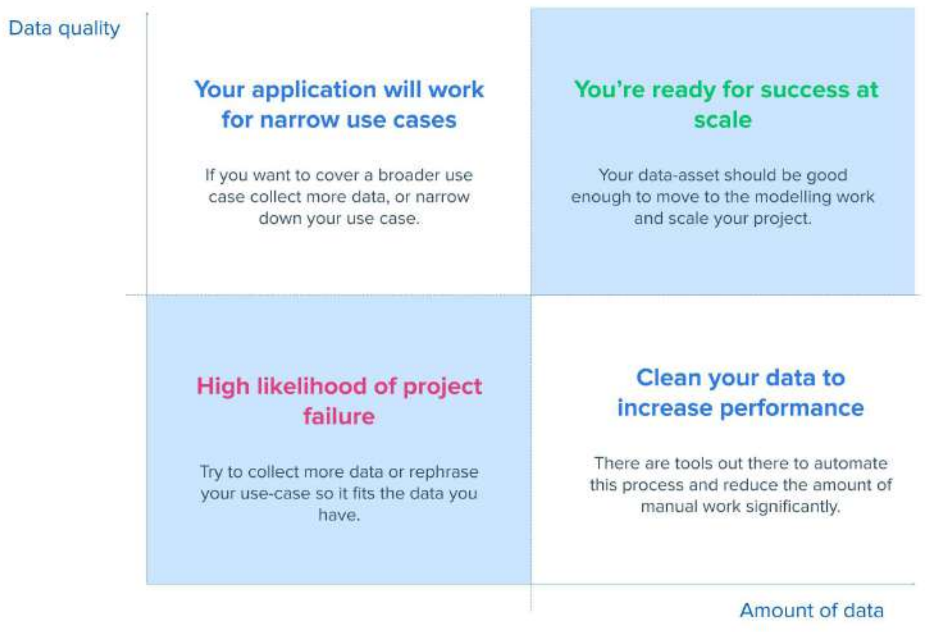
From the teams we spoke to, we identified the following as the top 3 characteristics of a high-quality data asset:

1. The distribution of the data matches the distribution of the variables of interest in the real world.
2. The labels are correctly assigned to the classes (a dog is labeled as a dog and not as a cat).
3. Annotations are pixel-perfect (you don’t annotate noise around the objects).

The amount of data available still matters, though. But the question should not be: “How much data do we have?” Successful teams ask: “How much good data do we have?”

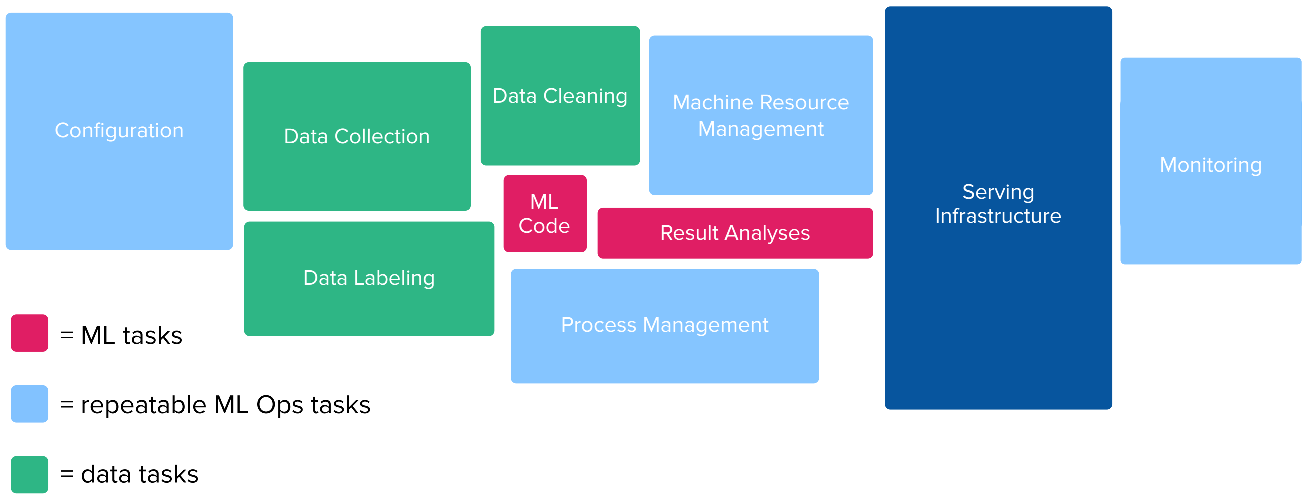
Whereas you can control data quality at any event, sometimes it is impossible to collect more data. But even with little data, scaling your ML application can work. Then data quality becomes even more essential. Also, if you only can work with little data, make sure to narrow down your use case as much as possible. The more specific the task is, the more robust your model will be with little data.

As every use case is entirely different, It is impossible to prescribe genetically how much data you need or how good the quality needs to be. Still, the decision matrix below gives you some indication on how to build out your data asset successfully:


Data quality
Your application will work for narrow use cases
If you want to cover a broader use case collect more data, or narrow down your use case.
You’re ready for success at scale
Your data-asset should be good enough to move to the modelling work and scale your project.
High likelihood of project failure
Try to collect more data or rephrase your use-case so it fits the data you have.
Clean your data to increase performance
There are tools out there to automate this process and reduce the amount of manual work significantly.
Amount of data

The tech stack, avoid the ML Ops Frankensuite

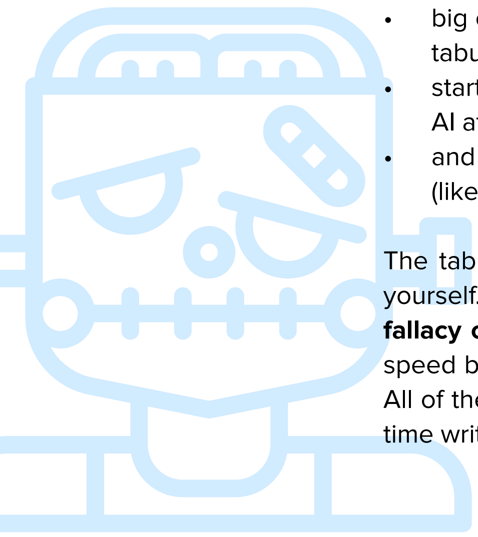
So far, we have only focused on building an ML application: creating a data asset and training models. Whereas these two tasks are the most critical factors for success or failure, many more jobs are to be done to bring an ML application to production. Below, you find an overview of the most important ones:



All of those tasks are referred to under the umbrella of ML Ops. They have in common that they are highly repetitive and (almost) the same for most use-cases. Therefore, we saw successful teams outsource their ML Ops and not build the tech stack themselves. They ended up not focusing on the differentiating part: creating the data asset and training models. It's a bit like with Dev Ops tools; no one would rebuild a collaboration tool for software engineering but use GitLab or -hub.

There is a boom of ML Ops tools happening like there has been with Dev Ops tools a decade ago, which can be summarized into the following categories:

- big cloud providers that offer general-purpose solutions for all types of AI (NLP, vision, tabular data, ...) in one dev tool;
- startups that provide solutions for one particular ML Ops task but for different types of AI at the same time;
- and startups specializing in the ML Ops process end-to-end for one specific type of AI (like ours, we're bringing you the vision flywheel out-of-the-box).



The table below puts some numbers behind buying an ML Ops platform vs. building it yourself. The benefits are clear, but when you buy your tooling, make sure to **avoid the fallacy of being dependent on a Frankensuite of tools**. We saw teams that lost a lot of speed because they used more than a handful of different tools to manage their pipeline. All of the tools themselves were great, but the teams ended up spending 25-50% of their time writing glue-code integrating them instead of focusing on what matters.

Infographic: Building vs. Buying an ML Ops platform

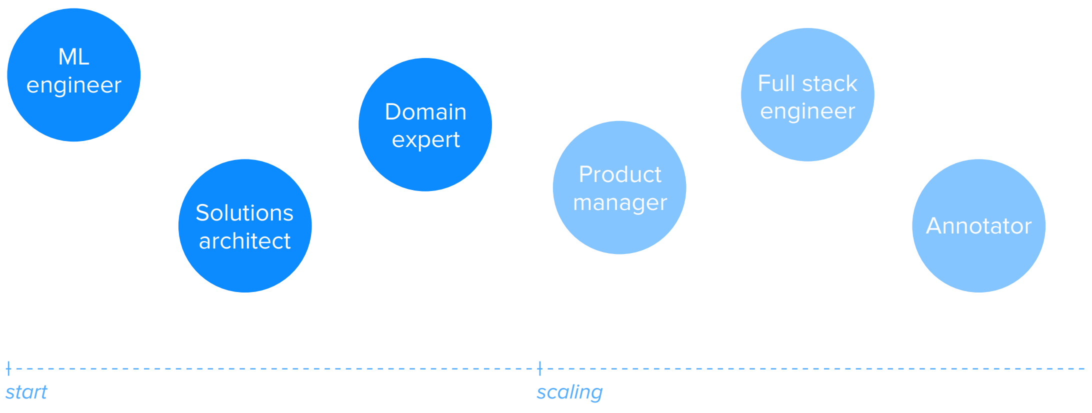
Build	Buy
Duration	
6-12 months + life cycle management	**1hr onboarding** automated life cycle
People	
5+ humans 3+ Full Stack, 2+ Dev Ops	**0 humans** fully automised
Costs	
$500,000 (1 years salary for 3 ML Engineers ($114,121 per) and 2 DevOps ($99,604 per)	**$3,500 starting annual subscription** (Hasty.ai pricing as a reference point)

(The estimates are based on our experience to build an MVP. To build Hasty as it is today, we needed 3 years and are a growing team of 30+ developers today.)

The team

Now that we covered the ideal process talked about building your data asset, gave some recommendations on the tech stack, it’s time for the last piece of the puzzle: the team to develop and scale an ML application.

Most teams we saw who deployed vision AI applications to production successfully had people acting in the following roles:



🟦 = team members at the start (must-have)

▨ = additional team members during scaling (could-have)

One person can have multiple roles at once, and many persons can have the same role building sub-teams.

Typically, teams start with a domain expert who knows the use case and understands the data, an ML-savvy person who knows what’s possible and how to do the first steps, and a solutions architect to build the integration with existing systems.

Then, when the project starts scaling teams some teams add some annotators on the way to cope with labeling needs, but with annotation automation tools this is not necessarily needed anymore. Finally, when the solution works as it should, teams add a product manager and a full-stack engineering team to the project if a custom interface should be built.

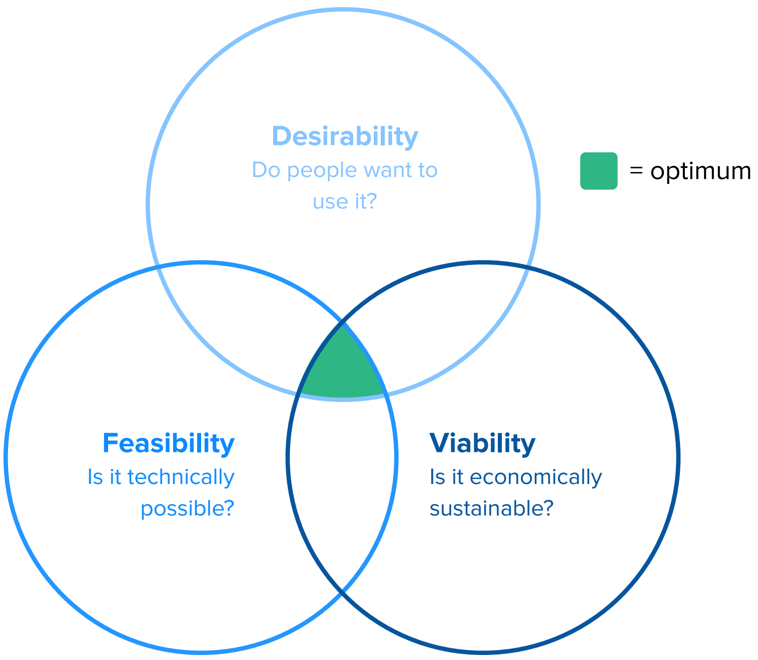
Hasty

Applying agile ML to your project

Drumm roll, please...We finally made it and now start talking about the actual blueprint after discussing all the theoretical considerations behind it in depth until here. This section provides you with an actionable plan to apply to your project to ensure that you're more successful with your vision AI project.

Disclaimer: we generally don't believe in silver bullets, and so this blueprint isn't one. It's our try to compress all of our learnings so that it's actionable for you. Our goal is to inspire you on what you could try to find the best approach that works for your specific use case.

The blueprint has four phases. Phases 1 and 2 are about getting started with your project, phases 3 and 4 are about scaling and maintaining the system. You should iterate through each phase in sprints until you hit the goal. We created the blueprint around the framework of "desirability, feasibility, and viability".


Desirability
Do people want to use it?
🟩 = optimum
Feasibility
Is it technically possible?
Viability
Is it economically sustainable?

Phase 0: Assess the desirability and feasibility of your idea

When you first have an idea for a vision AI project, you should ask yourself if people would want to use the end application (desirability) and if it's possible to build from a technology perspective (feasibility). Here, you're in a pre-primary research phase.

Assessing desirability is something only domain experts can do by using their knowledge about the end-users and the market in general. Here internal ideation sessions or user interviews with internal and external stakeholders are always helpful. It's a topic complex enough for a blueprint on its own, so it's good to read up on it if you're new to this.

ML engineers typically assess feasibility because they know what's going on in the research community and which applications are technically possible. If you don't have access to ML engineers with experience in vision AI, you can always reach out to us at tobias@hasty.ai. We're happy to provide you an assessment based on our expertise.

Phase 1: Assess the viability of your idea

In this phase, you should answer the question if it's possible to implement your idea in an economically viable way, i.e., it's a question of ROI. It's much harder to assess than desirability or feasibility because it's nearly impossible to accurately predict how much data and model fine-tuning you need beforehand.

Here, you'll be using agile ML methodologies to get an answer without a significant investment. The core idea is to start with a small team on a small dataset to get a feeling for your use case, train your first models early on, and in doing so, you'll be able to estimate the needed effort to reach your goal in 1-2 sprints of time.

If it turns out that your idea is not economically viable, you can kill the idea and re-ideate on it early in the process at a fraction of the cost of traditional ML. The rest of your budget you can use to test your next idea until one is a hit.

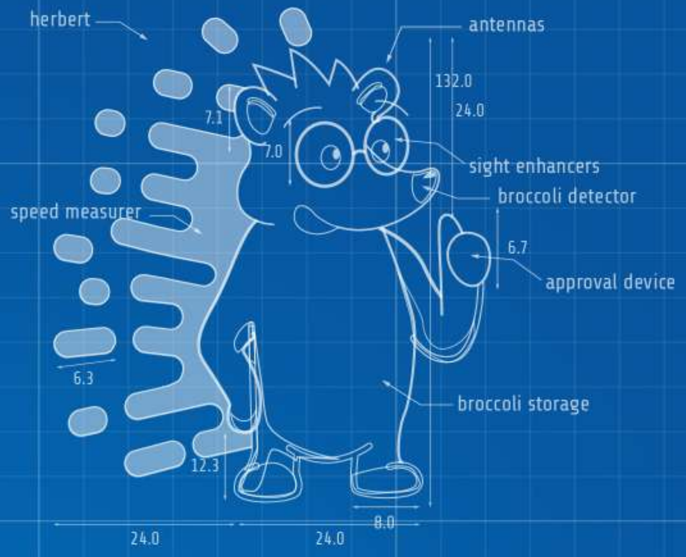
If it is economically viable, you can move over to the next phase.

Phase 2: scale your project

If you came until here, the hardest part is over. With a data-centric, agile approach, you can now keep enhancing your data asset. Constantly retrain models. Use the models' feedback to improve your data asset further. Build the custom integrations. And scale the team as described above until you achieve your performance goals.

Phase 3: maintenance

Congrats, you did it! You built an ML application that satisfies your needs, and you can ship it to your customers. Now all you have to do is to monitor your model's confidence on inferences. As long it's not decreasing, you don't have to worry about anything. If you experience a drop in confidence (most likely due to data drift), you should look at the images your models struggle with, annotate those and retrain your model again.



The vision AI blueprint

Iterate through each phase in sprints until you reach the target.

Phase 0: Assess the desirability and feasibility of your idea

- Talk to an ML engineer to ensure that it's technically possible to build your vision. You can also contact tobias@hasty.ai.
- Do market or user research to ensure that people would like to use the app.

Phase 1: Assess the viability of your idea

Before the actual sprint:
- Define the specific task your AI should do
- Define pre-and post-processing steps
- Assess integration needs to existing systems or if you need to build your custom interface
- Define an annotation strategy
- Define the duration of the sprint

During the sprint:
- Collect and label data and constantly re-train a model while doing so (doesn't need to be fine-tuned yet)
- Label the images your model struggles with until the performance doesn't increase anymore
- Clean your data
- Fine-tune the model

After the sprint:
- Assess if you should continue the project based on ROI or not. If not, see if you can rephrase the use case.

Phase 2: scale your project

- Continue the flywheel of labeling edge-cases, cleaning your data, and fine-tuning your model until you reach your desired performance
- Build the needed integrations, your custom interface, and pre- and post-processing steps

Phase 3: maintenance

- Monitor your model's confidence in production
- If the confidence decreases, label images with poor model performance and retrain the model

Hasty

Written by Tobias Schaffrath Rosario, Strategic AI-Project Lead

September 2021

Hasty.ai is an agile ML platform for building Vision AI-powered products. Annotate your image and video data, then build, optimize, and deploy your computer vision models into your product -- faster and more reliably.