In this post, I am going to document some trends, tricks and patterns I have derived from the Zero to One phase of my startup. I will try to broadly describe to you lessons I distilled while getting an Enterprise AI startup from nothing to its first revenues. First revenues are not success, they are necessary but not sufficient for success, so don't confuse this with recipe for a Unicorn say. If and when we make it big, another post on “How to create a successful AI startup” will come up :) .

Also, please note that all premise here is quite egocentric, that is, it doesn't take into account any discussions with other startup founders or many players in startup ecosystem. So, some of them might not apply to others at all. This is just me (not even my cofounders) trying to organize my thoughts around our journey, goes without saying this is my personal perspective and not an official stand of the startup I founded and work at. Also history doesn't repeat itself, so assumptions which were true for me might not be true for you, so please a feel try to intuit the assumptions when you are reading. Another important point you should remember is that this article contains “one possible set of tricks”, not “THE set of tricks”. Its better to know one set of tricks than knowing none, but please know there can be a very different, even contrary path for a startup to reach from zero to one. All disclaimers done, hope my musings can help you in anyway!

# Audience

Audience of this post is me 5-6 years back when I was thinking of starting a business. So, basically a person with following characteristics should it most helpful:

1. Technology person, who has decided to start their own business seeing an upcoming (or existing) startup trend/ecosystem in their country. Attracted to it because they might have heard of people earning a lot or building cool tech.

2. Doesn’t know much about business except “Let’s build some cool tech that will sell itself”.

3. Outsider to the startup world. (That is you don't know people who have done startups closely so as to take lessons from their life).

4. India based. Indians are quite different from the rest of the world culturally and there are things Indians observe/miss that others don’t.

5. You are not a super-exceptional team, that is not already famous before you started your company.

I will link definitions of startup linked terms in case you are such a beginner as described and aren’t aware of a few such terms. I did not know many startup-y terms when I started my company, I dont understand a few even yet maybe.

Now, even if you are more evolved/different than this startup-wise, you might still appreciate some points of the post, but you will find that other portions state the obvious. That is power of hindsight.

# Big picture of doing a startup – For Beginners

Let’s start with the basics first. Why did you get into a startup ? What are you really required to do when you are running a startup ? While for repeat entrepreneurs and people who come from a professional/social network of entrepreneurs would be very clear about this, an outsider is not. A 2014 version of me would probably not be very clear about some things here. If you read lot of startup literature, this might be a boring section for you as most stuff here is talked about a lot.

## Why start up ?

I did a startup because I wanted to work on tech that I found cool and also I wanted to be my own boss. That was the aim at least in the start. I think startup is an actually good place to get into if you have same aims as me. But as cliched as it may sound “with great power comes great responsibility”. Unlike in a day to day job, in startup you are taking up a lot of risk too. I cannot understate it here, if you dont make something good out of yourself in your startup, you will actually suffer loss and so will your near and dear ones and your early employees who trust you with their future. It takes sometime after you are in a startup to realize whether you were investing time and resources correctly or not. While there is a relived content type of feeling after seeing you have done relatively better for your people and your early employees, it also will be quite anxious a feeling otherwise. I will repeat it again, in early days of a startup, rewards are so sparse it is easy to confuse between a time period between rewards and pure inaction leading nowhere. You need to stay focused and stick to fundamentals during such confusing times, otherwise you might find that you have wasted time quite late. People generally try to understate the risk, getting secluded from social life, psychological hardships and stress, financial troubles and opportunity cost of people trying to enter the startup world as total noobs so that they don't get discouraged. I don't think you should think of the risk you are taking as soft, it exists and you need to be aware of it so that you don’t lose sight of your goal and lax out. It will be quite easy to lax out on the startup journey and you need to hammer it deep in your head that you cannot afford such a thing. That said, the upsides are great. I can ascertain that total absolute freedom to work in your style, taking decisions that actually change outcomes along the way and seeing your concepts change the world is a feeling that all the risk is worth it for. This feeling of achievement cannot be overstated. I assume when the financial gains reach you, it would be great too. (I’m not really rich yet so cannot comment how does that feel).

There are privileged people who do startups too. They can maintain good social life, life relatively comfortably even while doing a startup and will be shown in media as cool startup people. Don’t buy into the hype. Privileged people have more risk appetite and they can afford more failures and they can thus have many trials and need to get successful just at one. Probably that’s where the privileged founder narrative in the media comes from, more risk appetite is associated with better success, just due to the sheer number of trials and learning. For a lower middle class/middle class person (sp. If you are Indian), one failed startup is probably the maximum number of attempts they will get in this space, they need to make it count. Remember: “Make it count”. If you have a gut feeling not enough is happening, you are probably right, take charge and push and “Make it count”.

I think this section was more emotional than I intended to. Let’s try to come to more important business henceforth.

## What is a startup ?

What is a AI startup ? If you are building an AI startup, you need to :

\* Build a fast growing product business.

\* Fast growing means you can show metrics on which you are getting better faster YoY or quarter on quarter. These metrics are kind of fixed : Number of paying customers and revenue in B2B businesses and number of active users in consumer businesses. Unless you are really famous already due to some reason (or rich in which case you fund your own adventures), odds are VC wont fund your technology, papers, patents or github stars. In India, the famous people are ex-FAANG people, people from Indian unicorns, serial entrepreneurs, Stanford (or Ivy League) educated etc. They will find it slightly easy to find really good money for doing lesser than non-famous people. At seed funding level, being educated at India’s top T-Schools like IITs and BITS Pilani (I studied there) has some advantages. VCs will (unless you are special) either fund your startup if you have a product doing well, or a product showing growth and they can see potential in.

\* Product business means you are selling the same concept to all your customers. I mean you cannot sell soap to one person and toothpaste to another and matchsticks to a third person. The reason is, unless you have a product, you wont be able to grow fast as stated earlier. More so in AI as building proper sellable AI products takes time (I will talk about it a lot later in this post) and there is no way you can grow fast if you need to invest time to build for every client. This might sound obvious advice to people from elsewhere, but freshers doing AI startups would often feel they can do everything in the world and need to be aware about it.

\* Try to focus on your core business and avoid distractions. There will be many. If you have a good AI team, as of now the field is still nascent enough for your team to basically build into anything possible by AI. So every possible Deep Learning enabled business might sound like a missed opportunity to someone good at training algorithms. What AI founders need to realize is that PoC is just the first step and substantial investment post that is needed to build a product. Developing AI for a PoC is simple, building one which runs in production is not. Unless they FOCUS, they will not be able to build a product and hence not be able to show growth which they require to get funded and grow their business further.

\* Let me emphasize that I think there is a very good opportunity to build AI services business too. You can solve different problems for different clients and rent out your great AI workforce. But that is not something that is being covered on this post. I am trying to highlight tricks to build a AI product startup which can possibly be accelerated by VC funding. A very different approach needs to be taken there.

Story of all startups are different and one cannot copy playbook of one into another blindly. Different startups raise seed funding at different times, accordingly will have PoCs at different levels at time of seed funding, will evolve and pivot differently and thus will grow and raise money with different trajectories. I could recount the story of my startup here, but it would be at a different time, under different circumstances than other startups and hence to learn from story directly would either be an oversimplification or hard to relate. So I will list down things I think are invariants, will be needed in all AI startups without thinking about their path.

Unless you think you are really famous and can attract many rounds of funding just by your CV, I again urge you to stick to the fundamentals as much as possible. “Fast Growing” “Product” is what you need to build, stick to it and dont deviate.

## B2B businesses

Enterprise businesses are basically B2B. Good classification of B2B businesses can be found here: <http://christophjanz.blogspot.com/2014/10/five-ways-to-build-100-million-business.html> . An AI-for-enterprise startup is more on the elephant hunting side if we think according to the blog. That basically means, you have relatively fewer customers with large $ value. On the other hand, there can also be a B2B business that sells to many many customers but charges less $ per customers.

A utopia is a business like Slack (or Zoom) where you have growth like B2C business but you can charge companies as a B2B product. These B2B businesses become huge fast due to fast adoption at consumer level and then revenues follow as businesses pay. If you have an idea of a product which starts life as consumer and then makes employers pay, you can make it big real fast. That said, other business ideas need to be driven through inside/outside sales and marketing and dont rely on consumer based growth and network effect.

By the nature of it, a low $ product needs to be a simple one and a high $ product needs to be complex. If product is cheap, you cannot really have customer satisfaction teams and customer support. That means, you have to make a simple-to-use product UX wise and utility wise. The customer will have to adapt themselves to use the product and how well your UX helps them adapt determines the win in low cost high volume B2B products. OTOH, someone paying a high $ for the product will not be open to change their practices and will instead expect the product to cater to their way of doing things. That means AI-for-enterprise products will try to solve broad problems, will have to integrate with many other software, will need to provide some customer support and will need to adapt to customers. The take away is that enterprise products are more complex so as they can subsume different idiosyncrasies of large companies buying them, low cost B2B product need to be simple.

## AI B2B startups

By AI B2B startups (I also use the term AI first startups to describe such companies) I mean startups which are selling AI tool or its output directly to their customers. I dont count startups here who have an already selling product and they are augmenting their main product’s capability using AI. For an AI first startup , the USP is their AI capabilities while for a company enhancing their product using AI, their USP is their pre-existing product. These are different beasts. A company for whom the product sells an AI algorithm or its output will need to develop very different capabilities. Apart from figuring out the regular stuff like Product Market Fit, Pricing, Product Development and Sales, they will also have to figure out building a good dataset to train AI on, getting good AI talent and creating/deploying AI algorithms that actually work. Essentially AI startups are like other SaaS companies but also have traits that are unaddressed by SaaS playbook. I will hopefully address some of these issues you face in the post as we go ahead. Some of the concepts I introduce will apply to other DeepTech startups too where you have to develop your own technology before you can sell it.

**If you are beginner thinking of starting a AI first company, stop and course correct if you tick more than 2-3 boxes below. If you have:**

**A. Low expertise in business domain where you want to sell AI in.**

**B. Low expertise in building Deep Learning/Machine Learning systems.**

**C. Cannot raise a LOT OF seed money.**

**D. Don’t have decent Data Ops/Analytics/tech capabilities.**

**E. Haven’t figured out AI USP of your product at your university (or if you are brilliant during your self study).** Basically a lot of research work needs to be done by your team after starting the company. You dont tick E and you dont have to tick B automatically.

You can maybe make it work if you click over 3 boxes, but it will be hard !

We as a startup ticked A and E. We sorted out E somewhat by floating our own R&D team and doing original research to build the good proprietary technology available and A by hiring business experts from within the domain we are selling the product in. However, that required funding $$$, setting up annotation capability to collect datasets and doing timebound R&D to create algorithms which work. If there was one more point missing for us maybe, it would have been way harder to make it even to the first sales.

Low *expertise in business domain* is sort of guaranteed for a first time founder in their 20s/30s as you need to stay longer in industry to know it inside-out. Not saying no young founder can have the knowhow, but probability is small. One good way to sort out this point in early days of a startup is to have an equity compensated senior mentor or an experienced co-founder in the team. As the company grows, and your ESOPs get more repute, you can use them to hire domain experts.

Low *expertise in building Deep Learning systems* within founding team is another obstacle. Even if you understand the use-case well, you need one responsible person to make things happen. There are many people who claim they can create working AI systems, few who can even do it and most of these people don’t come cheap. The best is to have this person in founding team, the other alternative is bring in a well paid person (cash + equity) to get stuff done. This person should be responsible to figure out how the AI will work, how the technology and IP will be developed so that you stay abreast or ahead of the competitors.

*Its bad to not to be able to raise a lot of money initially*. Raising good money is excellent at any stage of the company (of course not diluting yourself like crazy) a good guide here: <https://www.saastr.com/benchmarks-in-saas-for-seed-and-series-a-rounds/> (In India, somewhat lower ticket sizes given low cost of running business). However, at seed stage, you probably will need to raise more for a AI company than a traditional SaaS business. Its good if your investor offers you a “nice” valuation for being an AI startup, or you will have to dilute extra equity if you are being judged as a typical SaaS (So its better for you to find an angel who is open to understanding how AI works than someone who purely invests on business metrics at seed stage). This extra money will be used to take care of many extra investments an AI startup needs. These are : A. Building Datasets to make your AI outputs better. B. Creating an AI team (and the cost of infrastructure required by it) which will create algorithms better than the ones available in public domain. This is your IP and USP which differentiates your company from another founder who decides to start a competitor with same or higher seed funding. TL,DR : Investments in AI are higher than SaaS to make revenue. I am not the only one that thinks so : <https://a16z.com/2020/02/16/the-new-business-of-ai-and-how-its-different-from-traditional-software/> .

*Data Ops/Tech/Analytics capability* is needed in a AI first startup just like any other SaaS. So its not just an AI team you need to solve everything. Right from day 1, you need to setup ops team to annotate data for your AI to train on. You also need to build product just like any other typical software product and have dashboards which show the client the needed information. In my estimate, there is an equivalent amount or even more of Data Engineering required as compared to the AI training to make AI work for a client. Very detailed analysis and conversation needs to happen when Data + Ops + Tech + Analytics + AI pipelines are being setup as these workflows will be repeated 100s of times during course of time. So its best to have these capabilities right when you start.

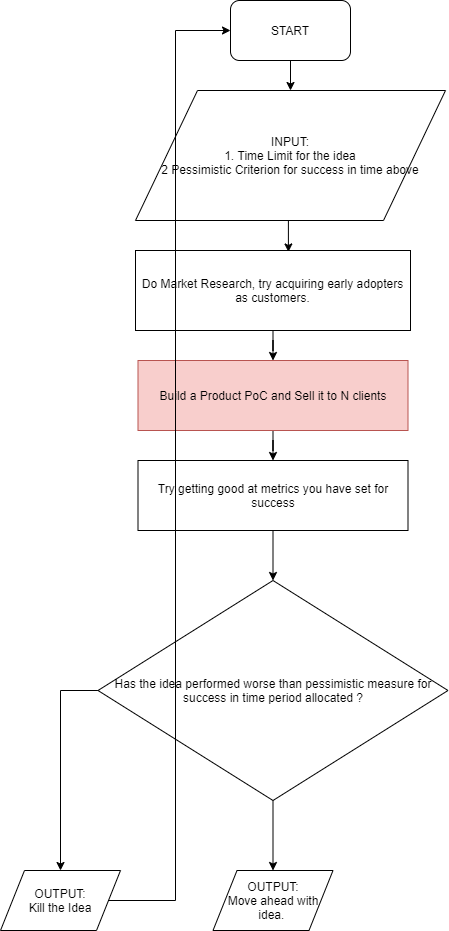
If you have already done *some research at your university* in field you plan to start a company in (or during the time when you were on a self learning spree) you have a starting advantage to differentiate your AI tech from a potential competitor . You also have brownie points when you speak to potential client or investor and you have some direction already about how would you setup AI and tech pipelines in future. A person who starts after setting up the company will have to figure out many things you have strong intuition about while on a timer. Thankfully for many of us self starters as of August 2020, a lot of smart talent from universities wants to go into FAANGs or established startups of the world (maybe because they have debt) and we have a window to get in and build from scratch.

Now that I have tried to address beginners with potential points I think they should be aware about, let’s jump to the core of the article : tricks to get from Zero to One.

# What product to build

The first question any team will need to answer is what product to build. The answer will be quite clear to people who have one or more co-founders/mentors who have a deep experience in some domain. Somewhat clear to people who have pre-existing research and background from their university days. You might have seen that my startup had a deficit in both these when we started, so we had to figure this out the hard way. If you are in a situation like us, remember that *your early startup idea, the one which you start with, will mostly not work*. A third scenario where you are very confident about your idea is if you are replicating business model of another pre-existing company slightly ahead of yours in curve.

AI has applications in many fields : HR, Insurance, Banking, Retail, Healthcare, Defence/Law Enforcement, Manufacturing, Agriculture and many others which I might not be aware about. How to choose when your founding team doesnt have a specialization in one of the fields already ? There is just one way, that is many rounds of { *Market Research → Product PoCs → Customer Validation} cycle*. Before you think you should try the many (painful/unyielding) rounds of cycles, think if you can bring in a domain specialist in the team, this step can shorten the time period. You will have to spend some amount of funds/equity to try the cycle just like bringing a new co-founder, but a faster result can make bringing a cofounder worthwhile. However, good + experienced people might come very costly (equity/fund wise) and might not want to buy into the brand you have at early stages. You should prefer running a few iterations of the mentioned Pivot Cycle over bringing in a co-founder whom you doubt about being excellent at domain expertise.

Figure 1: The Pivot Cycle

## The Pivot Cycle

The pivot cycle is to follow algorithm: *Repeat (Market Research → Product PoCs → Letting customers choose you)* until you are sure you are reaching early revenues. Just like the name suggests, its a three step algorithm run in multiple cycles to determine answer to one question : “Does the business idea pay as much its supposed to ?” or “Is the consumer utility as popular as its supposed to be ?”.

Very important to be clear about Pessimistic Criterion for success in allocated time period to judge whether an idea is worth pursuing in long run. You should be absolutely clear about it. “X [DAU](https://baremetrics.com/academy/daily-active-user) of avg session length T by end of MM/YY else I shut the business” in case of B2C or “Y [MRR](https://www.zuora.com/billing-topics/monthly-recurring-revenue-definition/) by selling to N customers by MM/YY or else business is not my cup of tea” is a concept you should have when you start an experiment.

Remember, there is an opportunity cost of pursuing an idea, which is, not working on a business you might be successful at. You will need to test each idea against a time bound benchmark if you need to be able to optimize in a space of ideas.

Its very easy to fall into the 10 year trap in a situation where you don’t know what your business idea exactly is even before you jump wholeheartedly into it: <https://twitter.com/NavalBot/status/1280230191755902978> . You might feel that this is such an obvious thing, you would need to be exactly sure what you want to do before you start but if that was the case you would not have heard of so many startup pivots. I think **most** first time entrepreneurs (I dont know about repeat ones) will figure out what their ideal business is only after a few pivots, however confident they are about their early ideas while starting. So you need to put harsh thresholds on ideas and cut/adapt ideas mercilessly to reach your ideal business. You should not get married unless you are very sure, to a person or to a business. Very sure in a B2B business is if more than one customers start writing recurring cheques for the same thing.

Another important thing which cannot be expressed as methodically is the sequence of ideas to be tried. At our business, we tried ideas sometimes very different and other times derived from each other. From a consumer app to different types of Medical Imaging AI to Retail AI to API platform to Social Media Analytics using AI before we found out what was the thing we could sell. You might want to keep the experimentation a bit lesser than that :-) . We also did experiments in parallel rather than one by one which I found effected focus and hence I recommend experimenting with ideas one-by-one. Some thoughts I personally formed on deciding on AI business ideas:

1. If you are thinking of getting into areas with regulatory capture (say healthcare, defence), you just dont need a domain expert, you also need government regulations expert in team. Just having the regulation cleared can be a massive barrier to entry. That said, clearing government norms is almost as much work as building AI and building and selling product. Best to avoid areas with government regulation unless you have very experienced team.

2. Low hanging fruits in AI, or things you can use a freely available Deep Learning model to solve, are generally thought of as commodities by investors (so early investors may often be confused as whether they should invest or not), but these might also be very solid and quick scaling business ideas if you know target audience to sell them. You might have to worry about your moat though.

3. If you choose the path like us that is an AI product in a large niche, your moat will be building specialized narrow R&D capabilities and rich datasets. More on this later.

Also remember that framework that investors think in is different from what entrepreneurs (techie people specially) think in. You might need to develop more empathy towards investors and how they function. So, let me define the early life of a startup in stages :

Figure 2: Early Life Path for a Startup

While I am speaking here about how to get through the pivot cycle and figure out the most important things according to me, idea and execution, VCs will almost always be more interested in looking at “Business Practices” or “Repeatability in Business”. This is just the next step after finding out the idea, which is making the startup success replicable for growth stages. So while in the Pivot Cycle, you will run experiments to figure out 1. The Product and 2. The utility, the investor will also be interested in knowing a third thing beyond 1 and 2, whether you can easily prove the utility to more and more people. This repeatability in business requires A. setting up salesforce in form of digital marketing, inside or outside sales and B. creating a financial model for your startup, which you need to do only after you finalize the business idea you want to pursue.

# Iterating the Pivot Cycle

Let’s try to understand one iteration of the pivot cycle. The steps in one cycle as I said earlier are as follows :

1. Come up with a budget, an idea and a reasonable pessimistic time bound target for the idea.

2. Do market research about the idea before you invest in it, you should do it along with step 1.

3. *Build the product and sell it to N first clients. {Whoa!}*

4. Try your best to achieve metrics you have set as targets and take a strict non nonsense call at the end of the time/budget predecided.

While 1,3,4 is something one can think of as definitely possible in a fixed timeframe (finite one time tasks), the feasibility/ease/hardships of building an AI Product and selling it to first N clients in a fixed time budget is hard to wrap ones head around (specially if they dont have experience building selling a product). Let’s for example say possible goals can be 8-12 months to build and sell an AI first product from scratch for $5000/year to 40 clients (SaaS) or for $70,000/year to 3 clients (enterprise). Beginners dont really have a standard to 1. set such goals and 2. compare whether the goal set is hard or easy, feasible or infeasible. This is an interesting phenomenon that humans can intuit, internalize and estimate quantities well when they have an inbound meter/category to compare against : <https://youtu.be/NNnIGh9g6fA?list=PL150326949691B199&t=495> (Check 5 minutes of the video from the timestamp I link here to get an idea of what I am trying to share, the whole video is very interesting though). If you are a beginner you can safely assume that either way you guess feasibility of goal set wrt point 3 above(say 8-12 months for such a target should be chill or this should be hard) based on your optimistic or pessimistic nature has no data point attached. I myself personally was clueless about good benchmark to set to take decision on an idea when we started our company, “When is an idea good enough to pursue as the product of the company?”. So temporarily while you read this section, you will have to live with my measure of feasibility/not-possible of a goal you can set for a business idea. We did a few iterations of the pivot cycle in our startup before early revenues and thus my judgment of feasibility is based on these 4-5 datapoints. From my experience 8-12 months is ambitious to build an AI product from scratch collecting around 250k US$ in Annual Revenue and is a hard target to achieve. You will need to hurry from day1 and take shortcuts and maybe keep the maximum horizon a bit longer per experiment than 12 months (say 15-18 months per idea, total 2-3 iterations of Pivot Cycle in 3 years if you can stretch your seed funding for that much). These numbers are based on markets which pay slightly less (Asia for example) and so your equivalent goal in North American market will be say around 500k USD of annualized revenue in same amount of time.

Conceptualizing, building and selling the product in a fixed time needs you to work very differently (sometimes in an non obvious way) as compared to when you are selling a product whose [Product Market Fit](https://en.wikipedia.org/wiki/Product/market_fit) (PMF) has been achieved. When we observe sales or product management practices of evolved/famous companies around, we generally see the sales and products of companies after their PMF. It is a very bad call to compare a new startup you are going to form where you are not yet sure about the idea with a company which is in growth stage or at least ahead in their PMF journey.

Let me explain why. In the early stages of your idea, you are trying to solve a complicated mess of [chicken-egg problems](https://en.wikipedia.org/wiki/Catch-22_(logic)) or dead ends. Please try and appreciate the following dependency graph about planning in an early AI startup :

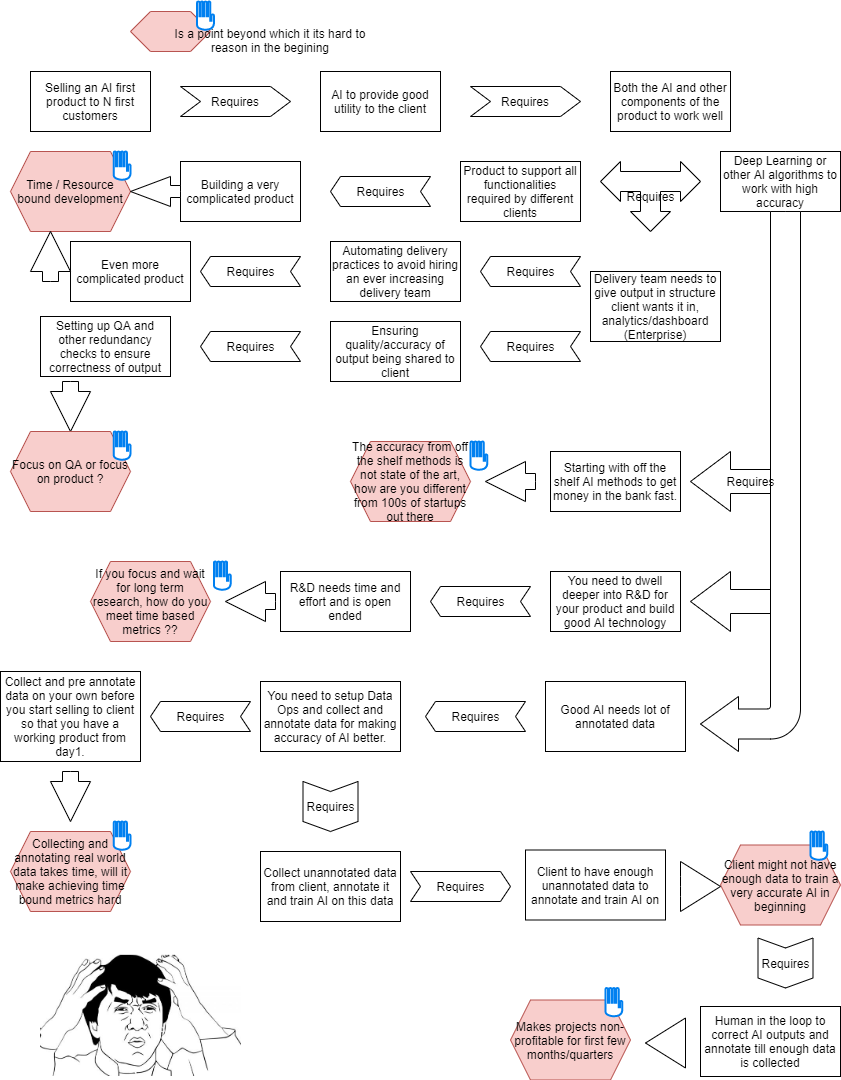


Figure 3: Questions you need to find answer for when you try to make quick sales to validate your idea in a AI for enterprise startup

Unlike in Product Management of a more “stable” startup, its very very hard to plan in detail all aspects of a AI product very early on if you are going to run a pivot cycle because there are things you cannot quantitatively estimate (look at the cannot think further symbol in the diagram above). You get slightly better with each pivot, but there will always be open ends. Every pivot will need all of technology building, product building, setting up data ops and closing first few sales in time bound manner, which effectively means doing things in a scrappy way. As you can probably realize, perseverance and pragmatism are two skills first time entrepreneurs should get better at to survive the pivots. This is quite in contrast to if you are experienced and know exactly what to target, that’s why I suggested if you can get an excellent domain aware co-founder if possible, you will have to struggle less. However, if this domain expert cofounder makes error in identifying the problem to solve, it is going to be really hard to do the pivots with all the new constraints. Pivot cycle requires you to take hard decisions, cut things and burn bridges after the time of an idea runs out. Someone whose reputation is on the line might find it harder to do so. There have been times when I said no to a client who came back after a very long time and we had pivoted the idea because it did not make the cut, very unfortunate and painful but cannot help it really!

My lesson from being in a soup of open ended problem is : “Don’t strive for perfection. Strive for survival.” Introducing our friend Small Wins : <https://www.semanticscholar.org/paper/Small-wins%3A-Redefining-the-scale-of-social-Weick/3c45ee6989cfe84f1f4866e26dbe7fae93de26b8> . Small wins is a political concept which says that you dont need to solve the entire problem which overwhelms you, just figure out the most important set of subproblems and focus on them. Basically, your aim should be to turn people’s opinion from “I don't think what you are trying to do is possible” to “Cool idea you have got there, but that’s not how its supposed to be done” or better “Here take my money and solve your s\*\*t by doing ABC”. Once that is achieved, there is better chance that an aggregated and more ambitious attempt will be made to solve the entire problem. I will of course tell what the important subtasks were which we choose to solve in our startup first, but please note that these focal points might be different for different startups based on their team, business and investors. In the initial phases of the enterprise AI product idea, the following points are what we gave more importance to:

1. Optimize for getting cash in the bank fast. That proves your business idea has utility.

2. In light of 1, build initial versions of products to make sure that you can deliver best results fast to early clients. Dont go for premature optimization, remember you are not even sure this is the idea you will be working on if it doesn’t prove itself.

3. Run Business Development in parallel with Product Development if you want to achieve the time bound goals.

## Building the initial versions of Product

Ok, so now you have decided to try out an idea in the pivot cycle. How do you go ahead with it ? Here are a few ideas/suggestions from me. I think there are three important themes that an AI entrepreneur needs to think of while developing the product :

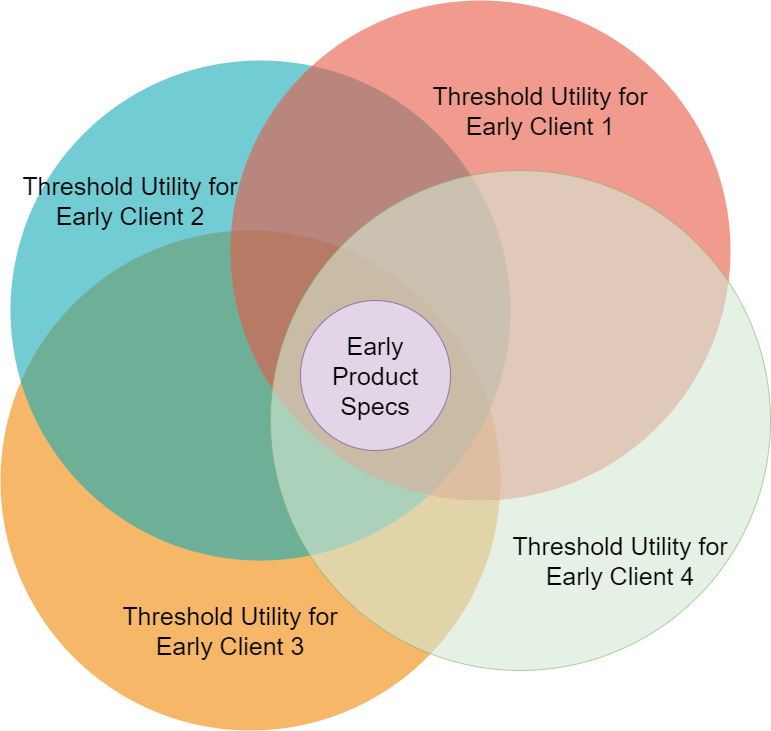
### Enterprise product v/s Enterprise utility

This is probably a relevant point only for enterprise-serving startups. I cannot imagine if and how other startups can use this hack. The hack arises from the complexity(and thus high ticket size) of fully-functioning enterprise product. Let’s try and understand why enterprise products are usually more complex. In a consumer software, consumers typically pay nothing and hence they will be willing to adapt and change their lifestyles around your product. That means you can design and make simple products without trying to customize it for each consumer’s lifestyle. In fact, it is better than the products are simple as you cannot do consumer support at all and hence they should just “get it”. More complex products with more features might not work just because users have no one to explain it to them. Typical SaaS ticket size products are slightly more complex to address some functionality at the cost of some complexity. The users pay small amounts and unlike consumers are more convinced about the utility when they start using the product to digest the complexity that comes with enhanced functionality. That said, you still dont do customization at client levels in small ticket-high volume SaaS. An enterprise OTOH thinks differently. Enterprises evaluate products on the basis of perceived utility they will deliver. Two important things you need to remember:

1. Generally, if an enterprise wants to buy a utility, it will be very hard to sell a subset of the utility just your product solves. For success, you need to solve enterprise’s problem rather than just selling your product.

2. However, enterprises will typically pay way higher than say an individual paying for SaaS because they need a lot of customizations in the way of utility they are procuring and they understand its hard to deliver all that. The customizations help them show results of using a new innovation quickly.

There is a threshold utility (which is generally quite high) below which enterprises dont want to procure a solution. This threshold utility often means entirety of the usecase they associate with your product and minimal change to an existing workflow they have. So for a document processing AI startup they will want a startup to automatically take data, train and run AI, process it, send it back to their system and automatically show dashboards informative to their stakeholders. Not just that, even in the data processing, they might need some AI algorithms specific to their own usecase apart from the generic ones you might have built. Net net, the utility the enterprise will ask for will be a superset of what you will be taking to them in initial stages. Think of 2-3 enterprises you are trying to sell to in parallel. Most of the time, they will have some overlapping threshold utility and some non-overlapping parts. If you build the entire threshold utility for each client inside your early product, your product will be optimized to serve just one client. That should not be done, your early product should be versatile and nimble that it can fit in use for all your early clients. Your early product should be the overlapping threshold utility part of your early clients.

  
Figure 4: Early Product needs to be the overlapping pattern across different threshold utilities of early clients

Now I have stated three points:

1. The early product should have the intersection of threshold utility of various early clients.

2. An enterprise client will not buy product unless their threshold utility is fulfilled. So you need to fulfill threshold utility.

3. The aim in the pivot cycle is to quickly sell to first few clients and get money in the bank.

There seems to be a gap here. How to sell when you are only building an early version of product satisfying part of threshold utility. This is where the hack is. Enterprise products should be relatively high ticket size and you would need a customer support team. This team typically consists of analysts, analytics professionals, developers and QA team. While in long term their role is to see that the analytics is running and AI output being sent to client has no errors, in early phase of startup you can additionally use their help to fill in the subset of threshold utility that you are not incorporating in your early product. Basically, while you build a core set of capabilities wihin your product, you invest somewhat more in a relatively large customer support team which builds custom modules to bridge gaps between your product and the expected utility for the client. What is product and what is custom modules ? I understand it wont be clear now. Let me try to show what I saying by taking example of the document processing AI product I was mentioning earlier.

Let’s say that our hypothetical document processing AI *“ProcureDocs AI”* product uses NLP to:

1. Highlight potential hotspots of a procurement documents. Places where one needs to read, reread and give more thought.

2. Detect possible objectionable clauses by/for a vendor in procurement documents.

[I have no clue about problems with Procurement documents and so all the specs above might be laughable when you look at the market, as long as you can understand that there are some specs for the NLP engine to achieve, we are fine, I did not intend to show anything beyond that]

So lets say your initial clients are a large mining firm, your local city council and a law firm that uses your product to enhance its services. All potential USD 50k/year deals. However, the following is what they want as their threshold utility to use your product :

|  |  |  |
| --- | --- | --- |
| Client | AI features Threshold Utility | Integration features Utility |
| City council | Auto verify if vendor is listed in directory or the internet, Check for unreasonably high line item bids, Purchase on unnecessary items, Check for unreasonable prequalification and Check for ambiguous language in specifications. | An interface to upload MS Word document, an email to be sent to concerned officials in case an issue is seen. Dashboard for the people of the city to visualize and look at possible problems. |
| Mining Firm | Purchase on unnecessary items, Check for unreasonable prequalification and Check for ambiguous language in specifications | MS Word documents as input, an output Excel with rows as MS Word document name, line number and possible gotcha/errors. |
| Law Firm | Auto verify if vendor is listed in directory or the internet, Check for unreasonably high line item bids, multiple contracts to the same company, Changes in minimum procurement thresholds, Missing Documents mentioned within contracts. | Image, PDF and MS Word archieve to be automatically scanned and PDFs flagging lines along with links to original documents to be provided |

Nice, so first of all let me define what I call as product and what I mean by custom modules . Product is the set of things you develop which are generic enough to cater to different clients. So if you want to run a new client’s data through your AI product without functionality of the product changing, you will have to do minimal to no development. The components of products are regularly tested and developed to scale as the clientele and usage grows. So suppose you have these 3 early clients, one way to solve things would be to just add every required feature into the product. That won’t be possible however (2 reasons) given A. You will not be lucky enough to start product development after fully understanding the specs for all your early clients, while deployment will be going on for one another one might be in PoC stage and another one in initial talks phase. B. Developing things which can generalize across any number of clients takes time, esp the integration part. Think of creating a dashboard for everyone who wants to use your product : generic, can be utilized across all possible clients, all the stakeholders in each client firm ; this will require a lot of time and iters to build. The more components you build to be used across clients (that is as a part of core product), the more maintenance they want and they reduce the product’s adaptability. So if you include all early client specs into the product, you will end up either wasting all the time budget you have allotted for one iteration of Pivot Cycle into balancing between requirements or end up making one mini product for each client if you want to achieve time bound sales goal. Both are bad scenarios and slow, the latter still probably allowing you to make a sale in the time budget. The best solution I can think of is diving the specs into core product and custom modules.

Early Product should have things you can be reasonably sure that will be useful across all early clients you have. These will be your core AI modules (NLP inference pipelines that you have built), or maybe some way to use different Machine Learning pipelines using same interface. So in our example above, you will have inference for Item Detector, Price Detector, vendor name detector, Requirement Detector, Ambiguity/Unusual language classifier etc in Core Product. You can have different trained model for each of these algorithms, but they can be designed to be used by common interfaces in the core product. So, you can also build higher order interfaces like a detector or a classifier where you can fit any specific model while inference. So for example a classifier interface in code can be used to detect both unusual language and requirements when different models are loaded. We also need components to host AI models as services, run different type of workflow based on configurations, annotation architecture to create training data etc as part of the early product. Essentially the idea is, you put up a configuration file for a client containing workflow and weights of models trained and your product can process documents related to that client. The early product will also include basic I/O like say JSON input files as input and JSON as output. If you are lucky and most of your clients want similar I/O, you should be able to have integrations as a part of the early product too. Requirements that do not fall under the core product specs should be developed as custom modules for clients in early phases. So basically module to automatically email output to one of the clients, automatically read files from fileserver of another client, OCR to convert images to text documents, dashboard for one client, Word to JSON conversion and other such things are better written as custom modules at client level to quickly be able to process their data. Writing generic versions of so many utilities will take too much time to plan and then build so that they co-exist with one another. Time is of essence as you are still trying to figure out whether this product works. So you have a product team building things that become generic across clients and maintaining and scaling components, so that, you don't need a mini product for each client, there is a lot of functionality you build using your delivery team, quick and dirty, to make sure that you are able to satisfy threshold utility of the clients without compromising the product’s adaptibility. You are betting here that there is domain specific data about client integrations and non-essential AI/tech features (right now being built as client specific modules) that you will gather along as you sell to more and more clients and include them in product later if need be, but for now you are trying to keep the product minimal and sell as fast as possible. Only the core features go into the early product.

Another thing you might ask is whether having a core product is even necessary ? We are just testing an idea here, why not just build separate set of “modules” for each client and sell them fast ? Why to even keep even a relatively smaller set of common functionality ? Apart from ofcourse less copy pasting code, more maintainability and less fuss while being deployed (Remember you just dont have to make code run once, you have to keep it running for the first revenues to come in), this also helps you to think hard and define the invariants of your products. These invariants make a product company different from an enterprise services company. A enterprise services company will not say No to most projects within their circle of competence, a product company will say No to projects not fitting these invariants. Also, according to me addressing these invariants will bring some structure into the first few deployments making them fast, as you are solving a recurring problem. Faster deployments will help you make a decision in the time budget. You dont want to waste time by preparing for circumstances you are not sure you will encounter, but for the pattern of problems you encounter frequently, you should reduce down the complexity so that you can delegate them and dont need to solve them twice or thrice. Let us take examples of invariants that you get as components of the early product of ProcureDocs AI. Invariant components of this product will turn out to be, at a very high level, entity extractors and classfiers, and then more specifically, you will have entity extractors like Item Detector, Price Detector, vendor name detector etc and classifiers like Requirement Detector, (classifies a paragraph say into requirement/not) Ambiguity/Unusual language classifier (classifies things as ambiguos/non-ambiguos) and so on. Once these invariants are set, other members of your team take responsibility of working on individual components and improving them. When I tried to Google hard about what such invariants are called in Product Management lingo, I found that a closely linked term is Product Architecture : https://www.productplan.com/glossary/product-architecture/.

What you have done in this step are not rules set in stone for the product. Once you are reasonably sure about product, you can choose to invest in the future than solve for the now. But, when you need a small win on a time budget, your risk appetite should be low and you can do some of what established companies would avoid as short-termism.

### Building the Early Core Product

Welcome to the second layer of the product onion. While you probably now are convinced by my premise that you need to have a core product and custom module for catering to enterprises, here is some more discussion about the core product. It isnt over yet ! Unlike the previous section on Enterprise Utility, this section is applicable to all sort of AI first products, so if you are a SaaS or consumer product in AI, this section would still be relevant. Few things you should consider here are :

A. Solve problem(s), dont just build an algorithm.

B. Dont optimize prematurely.

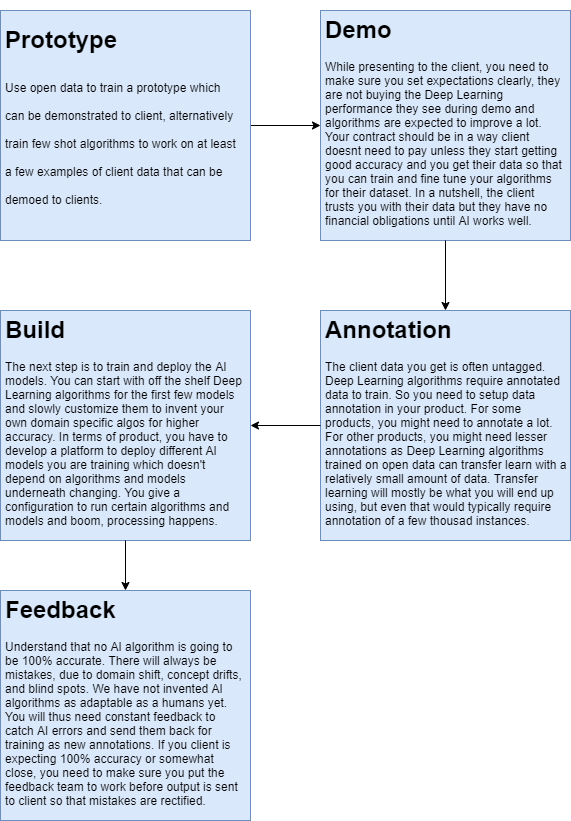
C. AI products follow Prototype → Demo → Annotate → Build → Feedback process. Dont wait for AI to get perfect before you sell/luanch.

So first premise is, dont start from Deep Learning algorithms when you are building the product, but rather from a problem statement. In our Procuredocs AI example above, dont start thinking like, “NLP has classification and language models, now what can be done on enterprise documents related to procurement ?” You need to take a real world problem (rather problems) as a starting point, which people actually want to pay for and plan your AI layer on top of that. Understanding what real problems exist on an enterprise dataset is what the Market Research step of the pivot cycle tries to find out. Now for all possible problems existing on an enterprise dataset (procurement documents for our hypothetical ProcureDocs AI), your Data Science expert needs to think and decide the problem(s) to solve that : 1. Have the largest TAM and 2. is/are feasible using existing Deep Learning/AI tech. Given you dont have research resources of Google and dont have pre-existing research from your university days, you will need to depend upon existing research a lot, at least for early versions of products, before you develop your own R&D in product’s domain. So you need to figure out what all problems on a dataset are theoretically solvable given you had a large dataset at your disposal for training AI algorithms. Remember a real world problem can be solved with a combination of AI algorithms, rather than just one. Foor example, ProcureDocs AI can use different models to recognize sentences with ambiguous requirements and detecting line items and their prices.

There are startups which try to solve problems using AI which have no previous research done on them and maybe not really technologically feasible. I dont know how do these startups really solve them. When we are choosing problems the startup will need to solve, we need to keep in mind that there is a reason that some categories of problems don't have feasible/working solutions in existing world. You may be able to solve it by creating totally new research, but when you are trying to evaluate an idea in a time bound way, its not plausible. More on problem selection later.

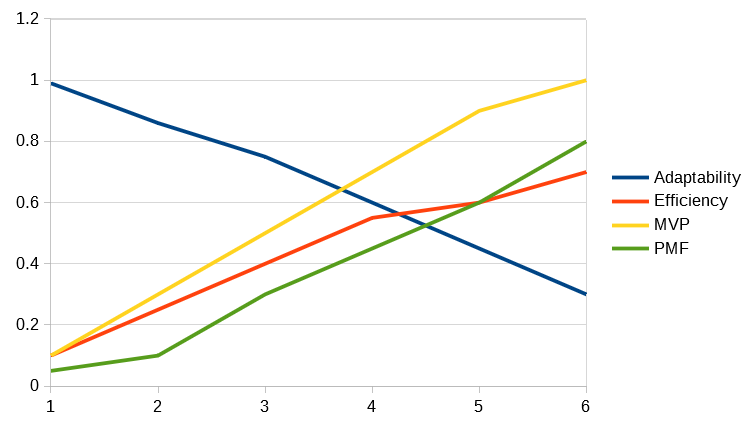
Second point is that you need to avoid falling into the trap most Product people want to fall into. Over optimizing product or running behind perfection. Trying to make product scalable/efficient/generic/pretty/easy to use (.. and 100 other adjectives which PMs would like to list on their CVs..) are useful efforts only after you are sure that you are going to stick to and sell a product. You are probably building something which 0 or 1 or <5 companies have built in the world, or it might not be even feasible to build. Rather than building the best possible early product, the best use of efforts is to just figure out an ugly, inefficient but feasible and fast path from just an idea to early $s in MRR. Once you achieve that, you can work on all the other optimizations. If you waste precious time from your idea time budget optimizing for any of the adjectives and in fear of lower margins for first few clients, you might never be able to get early clientele you want for the product to actually stick around and run. If you believe that perfect products can be designed in first iteration, you need to think about the early beta releases of software products you use now. Were they as good as they are in thir recent versions ? Mostly No. Let me introduce you to Mr. Gall ( <https://en.wikipedia.org/wiki/John_Gall_(author>) ), who says that most successful/awesome complex systems are design improvements of other less sophisticated systems. Very few awesome things were developed in the first epoch. So dont optimize prematurely, focus on your aim during pivot cycle, validate the idea ASAP.

The next point is dont wait for the AI to become perfect before you first approach customers. In a limited time/money budget, its hard for you to 1. Collect enough Data, 2. Annotate data and 3. Do R&D to invent the state of the art algorithms to get the perfect AI algorithm. If you are waiting for AI to evolve to the level when your product will be in production, before your first sale, I think you are just trying to be in your comfort zone. Customers are buying the AI you will deliver in the long run and not what you have when you speak to them for the first time. All you need to start working on making the first sale is some AI magic that you can show the customer (that might be AI working on couple of simple images or documents for our hypothetical Procuredocs AI), however you quantify ‘some’ and ‘AI magic’. These early AI magic results or Prototype as I call them in the Prototype → Demo → Annotate → Build → Feedback process is what you need to demo the AI to client first. Once you demo your product, you setup Annotate → Build → Feedback for the remaining round of rollouts and deployments, where you “annotate” more client data, “Build” that is train deep learning model and deploy and then taker errors made by the Deep Learning models as “Feedback” to create more training data and make the underlying models better. Remember, for now we know that no amount of clever training tricks and hacks can beat a Deep Learning model trained on large volumes of data. The more data you get feedback on and feed to the AI model, the more unbeatable your deployment for the client becomes. However, this is a journey you will have to take by setting expectations of the client and working towards increasing accuracy of AI models, you cannot have the full-blown AI ready on day 1.

  
Figure 5: Core Product Development Process

### The startupy terms : PMF/MVP and the product adaptability - efficiency tradeoff

You will come across the terms : Product Market Fit and Minimum Viable Product in traditional SaaS. How do these terms measure up in Enterprise AI ? According to me, MVP and PMF are not really points in startup timeline but rather continuums. You start developing a PMF the more of your MVP you develop. Enterprise products are just too complex and the threshold utilities so non-intersecting that thinking of developing a product that satisfies demands of all clients in the beginning before you start selling is [what I feel most people think of as MVP] is just hard to even think of. These continuums start from the Pivot cycle stage and continue to <figuring out repeatability in business> stage of the product. I would rather want to think of the product evolving from an adaptable product (in pivot cycle) which can cater to first customers for getting quick early revenue to an efficient one (if it graduates to the repeatable business stage from pivot cycle) around more common usecases for maximizing margins. You start getting closer to your MVP as the product starts getting more efficient and your Product Market Fit starts somewhere when you have achieved a certain level of efficiency and MVP. To be specific, PMF starts when you know that there are clients who will buy but you have to make product more efficient.

  
Figure 6: PMF, MVP, Product Adaptability and Product Efficiency Continuum [Qualitative Trends, exact values have no meanings]

## Selling the Product for the first few times

In this section, I will talk about some patterns I figured while selling product for the first few times to early customers. I think there will be disagreement/difference of opinion about initial sales based on founders’ personalities and how long they commit to try an experiment out in their pivot cycle. So disclaimer right now, your mileage may vary.

### There is no perfect Market Research [Or maybe just one way to do it]

Other alternative headings can be “Always be ready for surprises in early phase of business” or “A map is not the territory”- [https://en.wikipedia.org/wiki/Map%E2%80%93territory\_relation#%22A\_map\_is\_not\_the\_territory%22](https://en.wikipedia.org/wiki/Map–territory_relation" \l ""A_map_is_not_the_territory") . A beginner might ask the following questions if they have read the essay till this point: “Even before building a product you should have done market research and confirmed that clients should pay for it right ? Why do you need a Pivot Cycle at all then ?”. Also many people might have the image that your sales method should be sealed the day you take feedback from perspective clients because you have already figured out a way to build a pipeline by how you lined up these feedback interviews. “If I do perfect Market Research, I should basically be building an invincible product and sales methodology” might be another thought people would have while building a B2B/Enterprise product.

I personally believe that perfect market research is not possible (You can just avoid the Market Research step in case your team has lot of experience. I will talk about this scenario later). Let me try to convince you about my opinion before I start explaining you what you mostly cannot have it all figured out before you start. How many entrepreneurs do you think make the massive blunder of trying out an idea without proper market research ? It is definitely below 73% in my opinion. Try checking with your fellow entrepreneurs, most I know did market research to the best of their ability. However, it seems what people initially imagine and plan for their business to be rarely happens : <https://actu.epfl.ch/news/where-to-play-a-practical-guide-for-running-your-t/> . 73% of startups pivoted in a 10,000 startups study by EPFL. According to me there are three reasons for it : 1. Many facts you think you know about an industry when you are not as experienced, are actually just assumptions, not real facts and assumptions can be wrong. 2. People (both entrepreneurs and their contact points) can easily overestimate themselves and their capability to execute and 3. Many customers dont know what they want, so the initial feedback you collect about ideas has a low signal to noise ratio.

In enterprise AI, there are added factors :

1. AI as a technology has limitations. However, the status quo changes quickly and replaced by less-restrictive bounds. Its hard to estimate what progress will AI make in a limited timeframe. So when you are doing market research there is a very real possibility of under/overestimating AI capabilities.

2. There are no well researched and tried recipes to set AI businesses in motion. Many business people dont understand AI well and many AI enthusiasts dont understand business well.

3. Movies and MSM sensationalize news about AI, you will find extremely high expectations or paranoia more than you would expect.

So good market research and customer profiling will increase the probability of success of an idea, but you can never be sure. Except in one case : when you are copying an existing business model and your founders/important team members come from a competitor. If you are lucky enough to be in such a scenario, you dont even need market research. Just copy and build one advantage over the competitor.

### Early Customers

Probably the most important event for any B2B company is the time when they figure out who are going to be their early client(s). Good early customers are a must for an idea to move from a concept to some kind of business. Remember the figure above where there were too many open ends when you were trying to decide how to build and make your first sale. A good early client will help you make it slightly easy to navigate through that maze. We were lucky in our startup to get some extremely good early clients and thus could make the first few sales along with product development. Some points I would like to suggest about early clients in case of AI startups :

1. Start searching early for your first clients. Dont wait to build the entire product. In the Prototype → Demo → Annotate → Build → Feedback process, you should try to get early clients around demo step, maybe even during the time you are building the prototype.

2. Your early clients need to be senior people. Maybe a top position at a company, director of X or VP of Y type designations (in large organizations for an enterprise product). Dont fall in the trap of talking to even slightly junior people. You need a very capable and experienced person to A. Understand the product vision of a new product and more importantly B. help push a totally new technology without any references into a company, just on pure merit.

3. It helps if your early client is technologically aware. Even better if they are enthusiastic about AI. They can understand constraints and give accurate feedback about product and how to sell it.

4. Another very important thing is the willingness and patience of the client to review many iterations of the product and try to guide you with how much better your technology needs to be to make the cut. Very very few people apart from your early customers can help you with this type of data.

Infact, in some ways, if you just solve the step of getting good early clients, that is all you need to do to make your first few sales. Everything else you will be able to figure out along the way.

### Dont setup a sales team right from day 1

Another very important observation. Early sales hires are not going to be able to do anything. The first few sales the company makes are very different from how its salesforce will work later. Your company in the long run, should use outside sales (where your salespeople go and meet senior executives) or inbound sales (where you use marketing to create interest and get customers to approach you rather than you approaching the customer), as both the methods optimized to sell product at scale. But in the case of first few customers, the story will be different. The first few sales will be glacial, where you will be building different components of your products demoing and taking feedback from early clients until you have something good enough to sell. So selling to first few customers will not be optimized for scale but survival. Later, there will be a time where you will have to put thought into increasing the speed of sales after you are reasonably confident that you have satisfied the usecase of your early clients. Early sales are somewhat like how a door-to-door salesman sells, that is you will have to build trust, demo it and then only someone will buy your product, but with added complexity that you are creating components of product while making the sale. Examples of such inside sales process is abundant in IT services companies. However, IT services companies scale by using the trust earned in the first sale by upselling more services, which is a path a product startup cannot take, requiring them to make a switch to other sales methodsafter early customers. In fact, the indication of an idea graduating the pivot cycle and becoming THE product for a startup when the company starts thinking about scaling up sales and getting frustrated at how slow the initial inside sales process works.

All you need to take away from this section is that if you hire sales people day 1, they will not be able to help the business as startup sales professionals are typically trained to sell products at scale via outside sales or inbound sales. So founders will generally need to make the first few sales themselves. This will require them to resist the pressure to grow super fast in the pivot cycle and invest into understanding usecase deeply, solving it and getting the first $$ into the bank.

## Running the team

In this section, I will speak about a few things I observed while conducting the team. To be very truthful, when I started as an entrepreneur, I did not really believe in when other (very successful) ones said that building a good team is very important for the startup. That is because, in early phases of idea, a founder themselves manages lot of complexity, product, sales, tech etc. and they feel that they will be able to grow just using their personal ambition and force of commitment. Maybe that is why I felt earlier that all you need is good founders and some team which can do work. However, as the company grows and founders start becoming bottleneck, you start understanding why team building is extremely important. Like most other things in this article, this lesson was learnt the hard way. You need a motivated, good, solid team and also you need that strong founder instinct and then maybe you will succeed is what is my understanding now. You have 24 hours in your day and maybe you work with the highest efficiency x units/hour and still you will have upper limit on capacity. The work of a founder will quickly expand to fill all their time. When the company is scaling up faster than how fast you increase your productivity and you are all tied up and still more is required, there is no option but to have a team that can rise up to the occasion. My thoughts and lessons I learnt running an AI team in following headings :

### Adventure-Loving and Responsibility-Loving attitudes

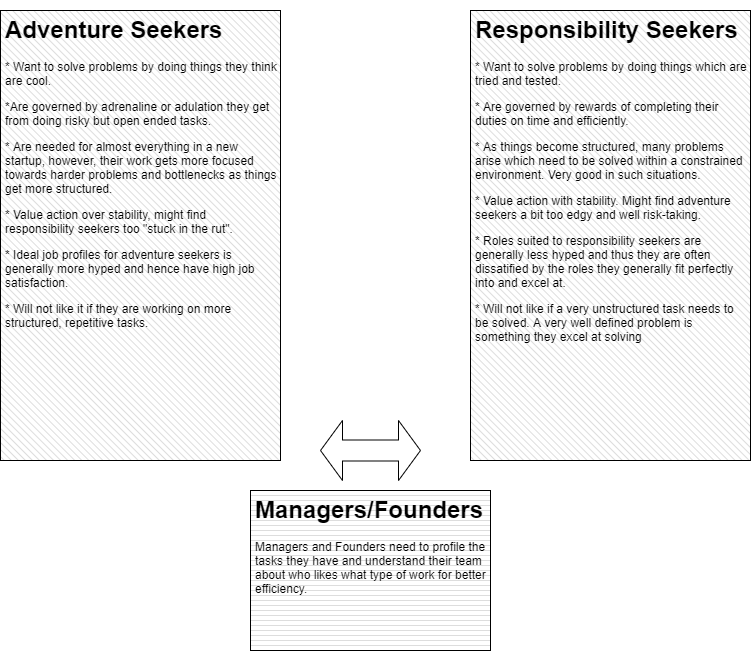
Tasks of every team in a startup has two components : Adventure and Responsibility. In product team, you need to make sure about reliability and test coverage and uptime and adding features which sales team demands, things you absolutely need for putting bread on the table. This component of work of product team I would call responsibility component of their job. Then there is scaling up of web services to serve twice the number of requests, making query on database 2X faster or making sure image are transferred faster to websevice from client are type of tasks I call as the adventure side of their job. In AI team, you have regular training of models to make sure newly discovered image categories are accounted for, updating NLP models with new upcoming text, updating different models for different customers, new annotations on data to increase accuracy of existing methods etc are what I think of as Responsibility side of their job. AI team can call thinking of methods to deploy models using less resources, developing new AI algorithms and training methods as adventure part of their job. Let me also extend this analogy to business team as well. Regular client calls, follow ups, making decks better, contacting and onboarding new clients, making sure delivery and client relation stays good etc. are part of Responsibility side of their job. On the other hand, getting on client calls to push for closures, exploring new products and a few other things can be thought to be adventure side of their job. I hope those examples help you intuit about the two type of tasks. To give it some kind of definition from my perspective: adventure tasks are totally open ended, that is the only thing you need to know about them is the goal needed to be achieved and then you figure out the way, while responsibility tasks have more structured paths with obstacles along the way to overcome. Responsibility tasks require handling some open ended complexities to sort out a well defined goal in a fixed time period, so it requires more of diligence and problem solving. Adventure tasks require more of risk taking and problem solving. So why such a long paragraph to convince you about this dichotomy ? In my limited experience, I have seen that people have preference towards one of these type of tasks. So if someone is trying to solve a problem, they will try to view it in the lens of Adventure like tasks or Responsibility like tasks. Because there are preferences involved, one type of people will view the behavior of others as too risky or stuck in the rut. *This is a good struggle to have in a startup.* You need both type of people who would use their respective forces to shape the journey, business, product and AI of startup into a halfway non-extreme path. Generally the best path is a combination of two thought extremes. When the pivot cycle begins, everything is an adventure like task and so adventurous people will basically be leading things more, but as the company figures things out and grows, adventurous tasks will reduce in number and increase in complexity and consistency tasks will take over the rest. There is never a dearth of adventurous tasks but you need to know that consistency needs to prevail in most parts of a successful organization for adventurous people disrupting smaller harder-to-crack bottlenecks of it. Adventurous-task loving people will have to thus learn to relinquish control over things which they controlled initially and move to harder problems to solve.

Now, humans have a bias towards admiring the risk taking dimension of character more than their consistency. So people who are risk taking get more psychological rewards than responsible ones. I feel that probably people become more risk taking because either they are conditioned to feed off this adulation or they love the adrenaline. So in AI team, there will be people who will always try to invent new tricks and publish new papers or write a deployment framework which makes deploying your algorithm 20% cheaper. This gets them somewhat more of a fame both within the team and sometimes even from outside.

Then there will be people who will get the day to day work done, run the algorithms through client studies, see where the algorithms were not working well and try to address the blind spot, train on newly data collected etc. These responsibility seeking rockstars are not praised enough. They thus might feel that they are working on irrelevant things and sometimes feel bad about their jobs even. Now of course if one is an adventurous type of person stuck in a role requiring too much responsibility, they should and might feel trapped, but I have seen people being really good at consistency/responsibility and not very willing to jump on open ended problems or less charted paths feeling bad about their job too. That is because we have probably built a world based on television and twitter and facebook where instead of encouraging adventurous type roles, we have started shaming responsibility type roles. If you dont work on an open-source project you are a bad engineer or the real salesperson is the one who closes the sales type of mentality makes our consistent folks feel incomplete for nothing. You are just different type, not inferior ! So many excellent responsibility-preferring people really good at their jobs live hating themselves due to groupthink.

The truth is both type of people are important in an organization, probably more responsibility seeking people than risk takers in established ones and the other way round in nascent ones. As the organizations become large, they will gather a larger number of responsibility-preferring people and thus will become averse to risky behavior allowing adventurous people to build a new B2B product.

That said, I am not profiling people into baskets. I just am saying most (I think all) people have a preference towards adventure type jobs over responsibility jobs or vice versa, but not saying people liking adventurous tasks dont have diligence or people preferring responsibility tasks cannot take risks. One example I can immediately recollect is Indian cricket captain Virat Kohli ( <https://en.wikipedia.org/wiki/Virat_Kohli> ) who was an explosive batsman taking lot of risks and scoring really fast in early phases of his career. However, now that he is captain of Indian team and looking from more of an organizational perspective, he has changed his game to focus on consistency. He doesnt get the super high scores of earlier, but now ensures he scores decent runs in most innings he plays. One is not better than the other, see ! One thing most employees and founders need to understand is which of these types of roles they are good at (and not which of them they like as most would like adventurous tasks due to human bias) and recognize/develop their circle of competence well rather than doing something sub-optimal. If you feel frustrated in uncomfortable situations, try using your love of order to develop your skills in responsibility tasks (No one likes a person whose comfort zone is doing nothing), if you feel trapped in day-to-day tasks, try combining your love of freedom with ability to achieve (No one likes an adventurer who gets nothing done). Also learn to recognize this diversity and appreciate people for their skills than dislike them for their differences from you. As a manager (you are a manager for free when you are a founder), you need to recognize the strength of people and use their help in tasks accordingly where they can contribute better.

  
Figure 7: Adventure Seeking vs Responsibility Seeking people

### Distribute Complexity, Dont delegate it or fall for illusion of control

Second concept that I would I like to mention is around inter-team co-ordination. Your startup will form natural teams when you do your day to day work. So as an enterprise AI company you will coherent teams around : 1. AI/Deep Learning stack management 2. ETL and webapp/mobile app management, 3. Client Success team, 4. Data Annotation and ops team, 5. Salesforce and Marketing team and maybe more dependent upon your business idea. These teams have very good communication internally as they sit for scrum once or twice everyday and members are involved in solving similar problems. However, when it comes to inter-team communication, there will always be gaps and lags. You can avoid this till you have say a 6 – 8 people team by making sure the concept of sub-teams of people within startup doesn’t exist at all, but after a certain number of employees its inevitable. These gaps between different teams are hard to fill, due to two different reasons :

1. Lost in translation : Business related information originating at client-sales interface or at client-delivery interface is known second (or third or say fourth) hand to a person in tech team and they can decipher it very differently that what it is intended to convey. Information sometimes doesnt even reach the tech person because some manager thought they are not required to know about the business problem but only the tickets related to it. Similarly, Tech information originating at Technology-Problem interface might be visualized at business-client front as too trivial or a big blocker unlike what it actually is. People overestimate their part and underestimate others’ parts by a lot.

2. Open ended questions cannot always have answers : There are always issues for which you will not have answers when you start to work on it. This basically makes different teams work with different assumptions abstracting out what they cannot incorporate into their mental model. After completion of some work, you might find that two teams might have made progress in different directions.

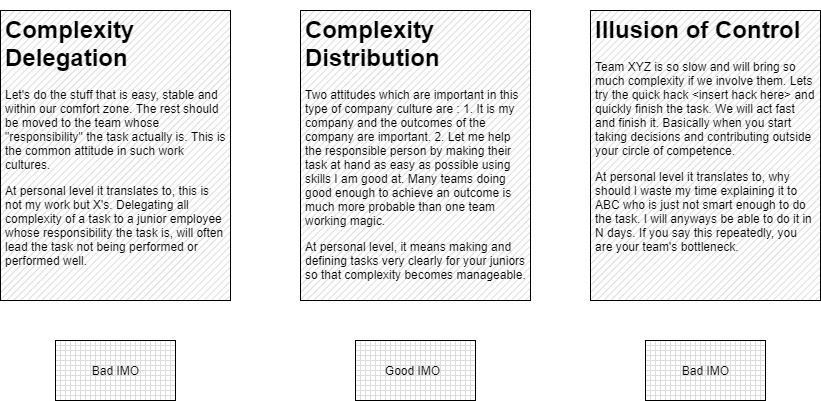
While one responsibility of the founders is to keep this inter-team communication gap minimal, let’s face it, you cannot make sure everyone has exactly the right information everytime in a dynamic environment as the organization scales up. Soviet Union tried to do that with their economy and they are history ! Let’s understand what happens when inter-team communication is weak and only after we understand how events unfold, we can understand how I propose to solve them “Distribute Complexity”.

So let’s say that there is a complex problem a sub-team within startup has to solve it. Say, business team has to convert a new and tough client or Technology team is working on a new feature which requires throwing away lot of code or AI team is starting to use a new Deep Learning architecture in production or Marketing team is thinking about a new blogging style. Everyone needs to admit that the problems they are trying to solve are not just theirs or someone else’s problems totally. So for example, Business team knows the client needs a very good AI demo to sell, so they cannot just say, the AI team should build a very accurate demo ad we will be able to sell. Similarly, when AI team introduces a new algorithm into the system, they cannot just say, let the Business or Client Engagement team wait for the feedback from client and tell us if there was any obvious issue. Or marketing team needs a set of technical blogs, they just send a mail to technology teams to deploy posts on the blog page. I have seen this attitude in many companies. Basically do not delegate all the complexity away to someone else who you think should do it as its their responsibility. Distribute the complexity, solve some parts yourself, let the other team solve some part of the problem. This attitude will only come if founders enforce it from the start, otherwise its very easy for one team to start delegating all the complexity away and working only on fun stuff from their comfort zone. So for example, for the hard client above, the business team can try to manage expectations about the demo to a level which the AI team can promise building given available resources, or the marketing team can take blog post from the tech people and try to make it better SEO wise rather than delegating everything. I feel if the culture of distributing complexity exists, startup will easily overcome unclear inter-team communication.

There is an exactly opposite problem to “Complexity delegation” and that is “maintaining illusion of control”. One very common example here is one team or person trying to do tasks inefficiently beyond their circle of competence. So software executives trying to setup a framework for Artificial Intelligence team, or business team deciding what database to use in the backend are quite common examples from such a culture. While there is some merit in people going beyond their ability and doing things not expected from them, doing it inefficiently always will not work. Founders need to advice good control and let people solve things and discourage them depending upon what they feel is optimal or suboptimal for the company.

### Problem Solving Hierarchy

Everyone solves one or the other problem in a startup. However, one needs to recognize that they have 24 hours in their day and can only work as much per day on an average. Basically, what I am describing here is the intra-team component of my thought about “Distributing Complexity”. Just like within teams you see these two extreme attitudes of delegating everything and containing everything, similar dynamics plays out within teams as well. There are some teams where the leader delegates a bit too much and junior team mates have to look into complexity issues they probably are not as well prepared to solve and then on the other end of spectrum there are leaders who want to solve all complexities themselves and delegate only what they think is “simple” becoming a bottleneck for the team and company. Try not to be these people! Try and understand that you are one person, can do one person’s work and you will need a team to achieve larger goals than what one person can build. So as a founder, you need to do 100% work as an individual contributor plus strategist and manager and also need to make sure the right amount of complexity is assigned to members of your team based on their profile and experience. Thoughts of perfection around “Only me can get all work done with least amount of failure risk” is misplaced as its costs time and makes work slow. Rather recognize teams potential for failure and focus on setting up processes that can detect failures early and before they explode. If you delegate a bit too much, you will see your team not making progress, that is a sign that you need to roll up sleeves and lead the team into accomplishments and then delegate back again.



### Last Mile Delivery is Messier than you think ! And its Very Important.

This probably applies more to Enterprise startups. Enterprises want data in some consumable format, which they can plugin to their existing systems and custom dashboards which their executives find helpful. That means while there is a product which applies AI on their data, analyzes it and converts it into an Excel file for all data they shared during an hour say, they dont just want this concise output your AI pipeline creates for them. They would want it in specific format, or maybe integrated into their reporting system, or as a custom dashboard. This output format might often be as important as the product’s utility itself, so you cannot say no! This thus gives rise to need of customer support team(or in their alternative role can also be called customization ninja team) whose aim is to figure out how to quickly solve these one time problems. As I mentioned above, this team is what helps you makes your first sales fast by figuring out quick customizations and helping the product focusing on core concepts. Its basically like you have one airport in the city to/from where you transport all air travelers than building an airport in each locality. Customer Support team figures out a way to make data to/from the team. So unlike in software services, your customer team might need analysts (of course), developers, analytics professionals and BI developers. I have not seen counterpart of such teams in other startups I know (even in the other experiments we run in pivot cycle), so thought it would be worth mentioning about this team. What you need to understand here also is the concept of complexity distribution. Its not always simple to integrate with proprietary systems like how client may expect. Dont just delegate whatever client says to this Customer support team, try setting client expectations and helping the team to close and automate the delivery process faster. Remember, this data to/from movement to your product is sizable work and if it gets slower for a client, the money in bank from this client might get delayed and also would suck time of team who need to manage delivery of the client next in line.

### Managing Expectations and Saying NO

Remember that the most important point of business is fulfilling the expectations of clients. However, you need to understand and talk more and more with your client to understand which of the expectations they cannot live without and which of them are good-to-have. Maybe say No in the short term to the good-to-have stuff to make good progress with the clients ? Remember you aim at least in the start is to get money in bank fast to prove the business idea. Developing too many features for a client might make you miss that goal. Another thing you learn to say no are other AI projects not related to your product. If you are trying out an idea to check its potential in pivot cycle, it will already be hard to prove it in a fixed timeframe. I you add more projects into the mix, you are diluting your attention and that increases the probability of a false negative (something that might have worked but you discarded it as something you couldn’t pull off). Trust me on this, an AI project you are doing without building a product to solve complexities of data pipeline, annotation and client integration around it is going to be extremely hard, learnt this the hard way.

## Research in your AI startup

Cool, now you have decided to work on an AI startup of yours. There will be some Deep Learning R&D required as you cannot use off the shelf models forever and your dataset will have its own peculiarities. Here are a few points I would like to share about the R&D.

### Start with a Real World problem, not an AI model

The first thing I would like to remind you again about is do not start thinking about what problems can you solve with the new cool algorithms you have around you. This leads to a bad cycle of sub-optimal products. Your task as an entrepreneur is to do research about real world problem statement you are trying to solve and then decide two things : A. Is it feasible to solve it using the AI methods available ? B. What method(s) would be required to solve the problem ? A is very important because there are people who get into solving things that AI algorithms cannot really solve like predicting social phenomenon, like say interview success or bullshit like Phrenology: <https://twitter.com/ylecun/status/1276147230295166984>. If you are an AI enthusiast joining an AI startup, please do give a thought if the company you are going to join plans to do something nonsensical. B basically is the process of breaking down the usecase into real world Deep Learning (or Machine Learnig problems). In images you can have a combination of Semantic Segmentation and GANs or in the example ProcureDocs AI we have been using, it will be a set of entity extractors and classifiers. Once you are able to figure out B, you have the blueprint to start working on the AI algorithms and hence the product around it.

### Don’t start expecting a Research Wonder

The only way I think you can start building your product with original research is if you were working on something in your university and you discovered something which you think is commercially viable. In all other cases, you need to start with off the shelf Deep Learning (or Machine Learning models) for quick first iteration and sales. In my opinion, you already should be demoing and talking to your clients by the time you think of new research, as that is when you realize what parts of the AI problems you are trying to solve carry more weight-age. Unless you are working on a very obsolete use case, some grad student would have recognized the problem you want to solve and would have published papers on it. Find out existing research for the problem you are trying to solve and build both basic product and baselines before you start anything innovative around new algorithms. Research can never be time bound and the experiment you are doing to evaluate your idea is. If you don’t find any research on the problem you are trying to work on, that is a signal too. It might be a really hard problem you are trying to solve or you might have not performed step B of previous topic well, that is breaking down the real world usecase into solvable Machine Learning problems.

### I dont think that publishing interesting results is bad in anyway

I have come across a few opinions like if you publish the methods you invent, aren’t you running the risk of people copying your IP ? The answer to that is we are. However, I believe publishing your findings has many important benefits which far outweigh someone using your methods after reading your papers. The first benefit is that while your method goes out into open, you get in touch with many enthusiasts who give you valuable feedback you can incorporate into your research. This feedback when cleverly combined with your existing research gives you even better results. Remember, while someone might be trying to implement the methods you publish in your papers, you will be implementing newer algorithms based on feedback you have received on your publication. The second gain is attracting good HR, who are attracted to your firm by looking at your work. As an early stage startup, it will be hard to hire expensive AI engineers and the only way to tell enthusiasts that good, promising and innovative projects are done in your team is to show them what type of results you publish. You might attract people who buy your tech mission when they read your publications and having a motivated innovative team is a big advantage. The third benefit is that your AI engineers and researchers get rewards which are very important to keep the team morale high during hard times of startup when seemingly no progress is being made. When you are running a pivot cycle and your experiments fail, techies will feel dejected as all the efforts they put into building something will mostly be trashed to make way for a new experiment. While you cannot help doing multiple experiments in pivot cycle, the publications keep the team’s morale high and they feel they have done something substantial building tech even if the business could not scale.

### Having a Research Vision is important after product graduates the Pivot Cycle

The startup needs to have a Research Vision about what the AI team specifically needs to develop in a quarter or year and measure progress with respect to that. Its hard to do time bound research but its not hard to look at the tasks done and estimate if any on-ground progress is being made. A very common mode I have heard a few AI startups go into is Data Scientists working in random directions and not achieving anything concrete. Its very easy to speak a few technical terms in front of non-AI people to overwhelm them and avoid having a discussion about real progress made. *So the AI lead needs to* :

1. Have regular review meetings about what are the goals the AI team is striving to achieve in say next quarter or half year.

2. Setup a Research Vision and all projects of AI team must fall in line with this research vision. Research Vision must take in account big pain points of clients and customer support team and the type of datasets company is planning and collecting.

3. Make sure there is progress. Data Scientists have their own comfort zone where they read and discuss more and more about possible methods and implement less. An idea means nothing, implement it and prove it works or doesn’t. More ideas you try out, more are the chances of success. A lot of applied AI research is clever and well informed trial and error.

# Moats for an AI startup

For new entrepreneurs, I would like to define what a moat is. A moat is essentially the world asking you how would your startup survive if say the richest and most innovative person in the world (Elon Musk maybe ? ) thinks of starting a competitor to your startup. The arguments you have in response to this question are guarantee to your investors that their money is safe and to your employees that they are not working on something that can become redundant fast. Its a hard question to answer and people will have answers like Network Effect etc, but we dont really know how this moat phenomenon will shape up for AI startups: <https://a16z.com/2020/02/16/the-new-business-of-ai-and-how-its-different-from-traditional-software/> . I have a few expectations about how a moat for an AI startup might look like in future :

### The usual Enterprise-as-a-client Lock :

It takes a really really long time to onboard an enterprise. You do demo/PoC, maybe multiple pilots, integrations, phased rollouts, many customizations once they are onboard and start understanding their internal lingo. Needless to say, this takes a lot of effort on both the teams, the startup and its clients. Inertia thus becomes your friend. Unless you really screw it up or are quite costly, the client will not want to replace you. With an effective pricing and good delivery, you can build a non-fragile business.

### Preferential Attachment :

A comparatively weak moat which can be achieved by very early movers/winners into a business idea. The idea is to use the first movers advantage to shape the market. A first movers gets the first client credentials, creates the first technology and inventions, creates the first wave in the industry, figures out the first pricing models and creates the first set of experts. Once your people are experts, you can shape narrative of the industry you work in. Unless you miss out on some very important aspect on which the late arrivals cash on (the entire gameplan OTOH for late entrants is to learn from mistakes of the early bird), the early mover is hard to defeat. Most businesses around AI are relatively new context and are waiting to be explored. Good time to build preferential attachments.

### Infrastructure Moat :

This is possibly the first moat which you see playing out in the real world right now. The trend with respect to AI models that if you can supply large amounts of data, the larger the model, the better it works. GPT3 for example is a huge AI model that can only be deployed on specialized hardware and not commodity hardware available to everyone else in the world. <https://news.developer.nvidia.com/openai-presents-gpt-3-a-175-billion-parameters-language-model/> . Now the fact that the NLP predictions by GPT3 are better than anything we have seen before and cannot be replicated by others due to the lack of hardware which can hold the GPT3 model, gives a unique advantage to OpenAI. Basically anyone using OpenAI APIs will get better results than someone trying to keep the model in their own premises. Very interesting economics would shape up here as maybe this is not going to be a monopoly but can become an oligopoly with other big AI players having huge and super accurate AI models they can only host. Such type of an advantage where a new entrant will have to invest substantially in hardware to get parity is achievable by large firms who are incorporating AI into their products or AI startups that have been able to raise a lot of money. That said, infrastructure moat also gives rise to the incentive to bringing huge models back to commodity hardware (say through discretization) and many startups will try (and might succeed) to democratize GPT3 and other huge models again.

### Data Moat :

While very large models trained on special hardware might be something that not all companies can afford, even relatively smaller models function better when more annotated data is available with higher variety for them to learn. So when two companies use exactly the same AI algorithms, one having more data to train them will have a better service than the other. So if you can setup a loop in your business which consistently annotates training data for AI making your training dataset bigger and better, you will always stay ahead of the curve. It doesnt just stop here, recent Deep Learning methods have introduced regimes where availability of unannotated data gives better AI performance. So you can be ahead of a newcomer just by having lots of data, even if you dont annotate it. Just having a lot of data can make you stronger than your competitors.

I think I have covered all topics I wanted to. That said, always open to any questions, you can tweet to me at @muktabh or ask a question on Quora: quora.com/Muktabh-Mayank .