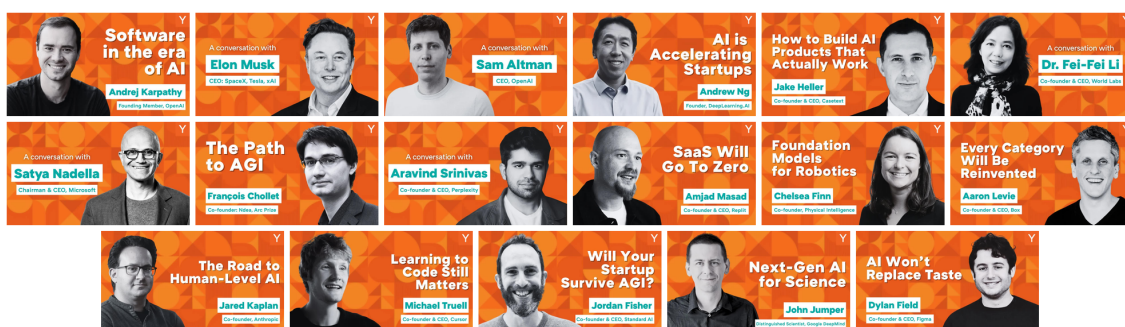


Y COMBINATOR

AI Startup School

Notes on the talks

Andrej Karpathy · Elon Musk · Sam Altman
Andrew Ng · Jake Heller · Fei-Fei Li
Satya Nadella · François Chollet · Aravind Srinivas
Amjad Masad · Chelsea Finn · Aaron Levie
Jared Kaplan · Michael Truett · Jordan Fisher
John Jumper · Dylan Field



The seventeen talks, in playlist order

One chapter per speaker, in playlist order. Each chapter opens with the video's thumbnail and a link to the talk. Written from the transcripts, and kept to what each speaker actually argued, in plain language, with no buzzwords.

Recorded in San Francisco, 16 and 17 June 2025.

Source playlist: youtube.com/playlist?list=PLQ-uHSnFig5NPx4adxI97CZb8vU4numwi

Contents

How to read these notes	6
1 Andrej Karpathy: Software Is Changing (Again)	8
1.1 Three kinds of software	8
1.2 What kind of thing is an LLM?	9
1.2.1 A utility, and a fab	9
1.2.2 Mostly, an operating system	9
1.2.3 The one way it is backwards	10
1.3 The psychology of the thing	10
1.4 Opportunity one: partial autonomy apps	10
1.4.1 Make the generate–verify loop fast	11
1.4.2 The lesson from self-driving	11
1.5 Opportunity two: everyone is a programmer	11
1.6 Opportunity three: build for agents	12
2 Elon Musk: Aspire to Work, Not Glory	13
2.1 Starting out: useful, not great	13
2.1.1 Zip2	13
2.1.2 Keeping the chips on the table	14
2.2 SpaceX: a small chance is better than none	14
2.3 How to be useful	15
2.3.1 Keep the ego-to-ability ratio low	15
2.4 First principles, with worked examples	15
2.5 On AI	16
2.5.1 What makes a competitive foundation model	16
2.5.2 Robots, and why he stopped dragging his feet	16
2.5.3 Safety	16
2.6 Mars, the long view, and Neuralink	16
3 Sam Altman: Don’t Build the Five Ideas Everyone Has	18
3.1 Deciding to start at all	18
3.1.1 Why the crazy mission made hiring easier	19
3.1.2 Nothing big starts big	19
3.2 The product overhang	19
3.2.1 Where ChatGPT is heading	19
3.3 How not to get run over by OpenAI	20
3.3.1 “Just-in-time software” and SaaS	20
3.4 Conviction when people say you are wrong	21
3.5 Agents, interfaces and devices	21
3.6 What changes for people	21
3.7 Hiring and running OpenAI	21
3.8 Science and energy	22

4	Andrew Ng: Speed Is the Strategy	23
4.1	Where the opportunities are	23
4.1.1	Why agentic workflows matter	24
4.2	Lesson one: work on concrete ideas	24
4.2.1	Where good concrete ideas come from	24
4.3	Lesson two: prototypes are now nearly free	25
4.3.1	Tools, and code as a disposable artifact	25
4.3.2	Everyone should learn to code	25
4.4	Lesson three: the bottleneck moves to product	26
4.5	Lesson four: understanding AI is itself a speed advantage	26
4.6	From the Q&A	27
4.6.1	On hype	27
4.6.2	On moats and token costs	27
4.6.3	On education	28
5	Jake Heller: Pick a Job, Then Grind the Evals	29
5.1	Picking the idea	30
5.1.1	The market got a thousand times bigger	30
5.2	Building it reliably	30
5.2.1	Find out what the best professional actually does	30
5.2.2	Turn steps into prompts or code	31
5.2.3	The hard part is getting it right	31
5.3	Marketing and selling	32
5.4	From the Q&A	33
6	Fei-Fei Li: Problems Bordering on Delusional	34
6.1	ImageNet: betting on data when there was none	34
6.2	From objects to scenes	35
6.3	Why spatial intelligence is the next problem	35
6.3.1	Why it is harder than language	36
6.4	A founder all along	36
6.5	What her best students shared	37
6.6	From the Q&A	37
7	Satya Nadella: Earn the Social Permission	38
7.1	Platform, product, partner	38
7.1.1	The model as SQL	39
7.2	Energy needs social permission	39
7.3	The rate limiter is change management	39
7.4	What is changing technically	40
7.5	Software, SaaS and software engineers	40
7.6	Underhyped and overhyped	41
7.7	Lessons from engineer to CEO	41
7.7.1	What he looks for in people	42
7.8	Quantum	42
8	François Chollet: Skill Is Not Intelligence	43
8.1	Why scaling pre-training did not get us there	43
8.2	What intelligence is	44
8.2.1	You hit the target and miss the point	45
8.3	The ARC benchmarks	45

8.3.1	ARC-1	45
8.3.2	ARC-2	45
8.3.3	ARC-3	46
8.4	What is still missing	46
8.4.1	Two kinds of abstraction	46
8.4.2	Combine them the way people do	47
8.5	The system Ndea is building	47
9	Aravind Srinivas: The Only Moat Is Speed	49
9.1	The browser bet	49
9.1.1	“What if everyone builds this?”	50
9.2	How Perplexity started	50
9.2.1	Why AI, and who to start with	50
9.2.2	Signs it was working	51
9.3	Why a startup could take on Google	51
9.4	Running a 200-person company	52
9.5	What is defensible	52
9.6	Business model	52
9.7	From the Q&A	53
10	Amjad Masad: Application Software Goes to Zero	54
10.1	From experts to everyone	54
10.2	The habitat matters more than the agent	55
10.2.1	Levels of autonomy	55
10.2.2	Agent v3’s three pillars	56
10.3	Predictions	56
10.3.1	Generalists, not specialists	56
10.4	From the Q&A	57
11	Chelsea Finn: Scale Is Necessary, Not Sufficient	59
11.1	The problem with robotics companies	59
11.1.1	What scale can and cannot give you	60
11.2	Dexterous, long tasks: folding laundry	60
11.2.1	Start simple	60
11.2.2	The breakthrough: pre-train broadly, post-train on curated data	61
11.3	Succeeding in places they have never been	61
11.3.1	The robot that ignored instructions	61
11.3.2	Testing in Airbnbs	62
11.4	Open-ended prompts and interruptions	62
11.5	From the Q&A	62
12	Aaron Levie: The Backlog Nobody Could Afford	64
12.1	The cloud wave, and why this one is different	64
12.2	Unstructured data becomes an asset	65
12.3	Why agents create more work, not less	65
12.4	A new set of nouns and verbs	66
12.5	Business models: from seats to work	66
12.6	Will companies just build their own software?	67
12.7	From the Q&A	67

13 Jared Kaplan: Ask the Dumbest Possible Question	69
13.1 From physics to AI	69
13.2 How models are trained, and why they keep improving	70
13.3 Two axes of capability	70
13.4 What is left	70
13.5 The conversation	71
13.5.1 Claude 4	71
13.5.2 Humans as managers	71
13.5.3 Depth and breadth	72
13.5.4 Green fields	72
13.5.5 What physics taught him	72
13.5.6 What would convince him scaling has stopped?	72
13.5.7 Efficiency and cost	73
13.6 From the Q&A	73
14 Michael Truell: Be Consistent With Your Beliefs	74
14.1 Starting young	74
14.1.1 The robot dog	75
14.2 The ideas that did not work	75
14.2.1 A copilot for mechanical engineers	75
14.3 Choosing coding anyway	76
14.4 Building the first Cursor	76
14.5 Wandering the desert, then compounding	76
14.5.1 Where early growth came from	77
14.6 The future of coding	77
15 Jordan Fisher: Plan Two Years Out, Ask Better Questions	79
15.1 Confusion as a starting point	79
15.1.1 Focus, and focusing on everything	80
15.2 Questions about products and markets	80
15.2.1 Will enterprises really be slow?	80
15.2.2 Will software commoditize?	80
15.2.3 On-demand software and interfaces	80
15.2.4 Retrofit or build from scratch?	80
15.3 Questions about teams and trust	81
15.3.1 Security and agents	81
15.3.2 Why small, automated teams are harder to trust	81
15.3.3 How much alignment does the economy need?	81
15.4 Questions about moats	82
15.4.1 Does proprietary data still matter?	82
15.4.2 Capacity	82
15.4.3 What is still hard after AGI?	82
15.4.4 Is there an intelligence ceiling?	82
15.4.5 Neutrality	82
15.5 Change the world, not just make money	83
15.6 From the Q&A	83

16 John Jumper: Research Was Worth a Hundredfold	85
16.1 A winding path	85
16.2 The problem: protein structure	86
16.3 How AlphaFold was built	86
16.3.1 Many mid-scale ideas, not one big one	87
16.3.2 Cross the line that matters to users	87
16.4 Making it available	87
16.4.1 Users do surprising things	88
16.5 How it changes the practice of science	88
17 Dylan Field: Craft Is the Differentiator	89
17.1 Getting started	89
17.1.1 Surviving the pivots	90
17.2 First users, and launching too late	90
17.2.1 Product–market pull	90
17.3 Why design matters more now	91
17.3.1 How Figma decides what to build	91
17.3.2 Blurring roles	91
17.3.3 Interfaces	92
17.4 Design in research	92
17.4.1 The designer’s role in ten years	92
17.5 Advice for people in their twenties	92
17.6 From the Q&A	93

How to read these notes

Seventeen talks over two days: lab leaders, researchers, and founders at every stage from four people in a living room to Microsoft. The speakers disagree about a good deal, which makes the points they share more telling. Five come up again and again:

1. **Build for the model that is coming, not the one you have.** Kaplan tells the room to build things that do not quite work yet, because the next Claude will make them work. Masad tells his team to be fine shipping a crappy product, because in two months the models improve and it becomes viable. Altman describes a product overhang: capability is far ahead of what anyone has built. Fisher pushes the horizon furthest and says to plan two years out. Truell's whole company came from taking that seriously in 2022, when competitors were only making existing products a bit better.
2. **Generating is easy; verifying is the product.** Karpathy wants the generate–verify loop spinning fast and the AI kept on a leash. Heller says the qualification that matters is two sleepless weeks on one prompt's evals. Kaplan observes that for AI, judging and generating are close in difficulty, so people become managers who sanity-check. Nadella notes liability still sits with humans, so they need tools to see what agents did. Jumper's advantage was blind assessment on answers nobody could see; Chollet's whole talk is about measuring the right thing; Finn measures every claim rather than trusting a good video; Field wants designers writing evals. Nobody says the model's first output is the hard part.
3. **The market is the work people already pay people to do.** Heller's rule for picking an idea is to look at salaries, not software budgets. Levie argues the biggest market is the backlog of work nobody could justify paying a person to try, and that AI finally opens a new set of “nouns and verbs” for startups. Nadella calls the drudgery in knowledge work the most understated opportunity, and names change management, not capability, as the rate limiter. Ng says the application layer has to be where the money is, because it pays for everything below.
4. **Do not build what everyone else is building.** Altman bets the person who builds something bigger than OpenAI is not working on any of the five most common ideas. Li has spent a career on problems “bordering on delusional.” Musk started SpaceX at perhaps 10% odds because no one else would. Chollet argues the field spent years optimizing the wrong target. Jumper says headlines credit one idea when many mid-scale ideas did the work. Fisher calls much of the industry groupthink, and says investors who think they are early are two years behind.
5. **Speed is the moat, until it is not.** Ng calls execution speed the strongest predictor of success. Srinivas says speed is the only moat against far richer competitors. Altman and Levie both say startups win when an industry's clock cycle changes this much, and Levie puts a three-year window on it. The dissent is Fisher's: speed is a rat race, and after AGI the durable advantage is working on problems that stay hard, like energy, chips and manufacturing.

Two disagreements are worth reading side by side rather than averaging away. The first is timing. Musk expects digital superintelligence this year or next, and Fisher plans for AGI within two or three. Karpathy, fresh from twelve years watching self-driving go from a perfect demo to an unfinished product, calls this the *decade* of agents (Chapter 1), and Ng thinks AGI has been overhyped (Chapter 4). Which of those you believe should change how much autonomy you design into what you build. The second is the fate of software companies. Masad predicts all application software goes to zero (Chapter 10); Levie, using the idea of core versus context, argues companies will not want to own, debug or be liable

for software that is not core to them (Chapter 12). Both are describing real forces.

A note on sourcing. These notes are written from automatic transcripts, so wording is paraphrase throughout rather than direct quotation. A few names that the captions garble or omit are taken from the video thumbnails, and interviewers are named only where the conversation itself makes clear who they are. Several talks leaned heavily on slides or robot videos; those chapters record what the speaker said, not what was only on screen.

Boxes are used consistently throughout:

- **The one thing.** The point worth remembering if you remember nothing else.
- **Against the grain.** A belief the speaker held that the people around them did not.
- **Try this.** Something concrete enough to actually go do.
- **Watch out.** A trap the speaker explicitly warned about.
- **Parting advice.** The closing thought each speaker left the room with. Where a talk ended on a summary, a prediction or a lab's plans rather than advice, this box collects the closest thing the speaker did say, and says so.

CHAPTER 1

Andrej Karpathy: Software Is Changing (Again)



“Andrej Karpathy: Software Is Changing (Again)”

Andrej Karpathy Former director of AI at Tesla; founding member of OpenAI. A solo talk with slides, no Q&A.

Video: <https://www.youtube.com/watch?v=LCEmiRjPEtQ>

He opens by noting that much of the audience is students about to enter the industry, and that this is an unusually good moment to do it. His reason: software barely changed at a fundamental level for about seventy years, and has now changed twice in quick succession. That means a huge amount of software has to be written and rewritten.

1.1 Three kinds of software

The frame for the whole talk is a numbering scheme he introduced years ago and has now extended.

- **Software 1.0** is code you write for a computer.
- **Software 2.0** is the weights of a neural network. You do not write it directly; you curate datasets and run an optimizer that produces the parameters. Hugging Face plays the role GitHub plays for 1.0, and a fine-tune on top of an existing image model is roughly a commit.

- **Software 3.0** is prompts. Until recently neural networks were fixed-function, image in and category out. Large language models made them *programmable*, and the programming language is English.

His example is sentiment classification, which you could now do three ways: write some Python, train a classifier, or write a few-shot prompt. GitHub repositories are already filling with English interleaved with code.

THE ONE THING

Be fluent in all three paradigms. Each has different strengths, and you will keep having to decide whether a piece of functionality should be explicit code, a trained model, or a prompt, and to move between them as things change.

He grounds the “eating the stack” idea in his time on Tesla Autopilot. The system started as a lot of C++ with a few neural networks doing image recognition. As Autopilot improved, the networks grew and the C++ was deleted. Stitching information across cameras and across time moved from hand-written code into the network. His claim is that the same thing is happening again, one level up.

1.2 What kind of thing is an LLM?

He tries several analogies and is candid that each one fits only partly.

1.2.1 A utility, and a fab

Andrew Ng once said AI is the new electricity, and the utility comparison holds in places. Labs spend capital to train models, which is like building a grid, then serve intelligence over APIs with metered, per-token billing. Customers make utility demands: low latency, high uptime, consistent quality. Tools like OpenRouter act like a transfer switch between providers. And when several major models went down in the days before the talk, people could not work. He calls that an *intelligence brownout*: the planet gets a little dumber when the voltage drops.

They also resemble semiconductor fabs: huge capital cost, a fast-growing technology tree, and research secrets concentrating inside a few labs. Training on your own TPUs is the Intel model; building only software on someone else’s GPUs is the fabless model. But the analogy is muddled because software is easy to copy and change, so it is less defensible than a fab.

1.2.2 Mostly, an operating system

The analogy he thinks fits best is an operating system. These are not commodities from a tap but increasingly complex software ecosystems, and the market is taking a familiar shape: a few closed-source providers, with an open alternative (he suggests the Llama ecosystem) that might grow into something like Linux.

In his sketch, the model is the CPU, the context window is memory, and the model orchestrates memory and compute to solve problems. An app like Cursor runs on GPT, Claude or Gemini the way VS Code runs on Windows, Mac or Linux: it is a dropdown.

AGAINST THE GRAIN

We are in roughly the 1960s of this kind of computing. Compute is expensive, so it is centralized in the cloud, and everyone is a thin client time-sharing the machine as one row in a batch. The personal computing revolution for LLMs has not happened yet because it is not yet economical. There are early signs (Mac minis turn out to suit single-user inference, which is memory-bound), but nobody knows what it looks like. Likewise, talking to a chatbot in text is like using an operating system through a terminal. A general graphical interface across all tasks has not been invented.

1.2.3 The one way it is backwards

Transformative technologies usually reach governments and corporations first, because they are new and expensive, and only later diffuse to consumers. Electricity, cryptography, computing, flight, the internet and GPS all followed that path. LLMs flipped it. Early computers were about ballistics; his own use of LLMs is often about how to boil an egg. Corporations and governments are lagging behind individuals, and a new computer was delivered to billions of people essentially overnight.

1.3 The psychology of the thing

Before programming these systems, he argues, you need a working model of what they are like. His phrase is *people spirits*: stochastic simulations of people, produced by a transformer fitted to internet text, which gives them an emergent, human-like psychology.

- **Superhuman memory.** Like the savant in *Rain Man* who could memorize a phone book, they remember far more than any person.
- **Hallucination and weak self-knowledge.** Better than before, not solved.
- **Jagged intelligence.** Superhuman in some domains, while making mistakes no human would make, such as insisting 9.11 is greater than 9.9.
- **Anterograde amnesia.** A new colleague learns your organization over months and consolidates what they learn. Models do not do this natively. The context window is working memory, and you have to program it directly. He recommends *Memento* and *50 First Dates* as accurate portraits: fixed weights, context wiped every morning.
- **Gullibility.** They are susceptible to prompt injection and can leak data.

The job, as he puts it, is to hold both facts at once: a superhuman system with real cognitive deficits, and work out how to use the powers while working around the deficits.

1.4 Opportunity one: partial autonomy apps

Why paste code back and forth into a chat window, which is talking to the operating system directly, when you could use an app built for the job? He uses Cursor as the model of an early LLM app and pulls out four properties he thinks generalize:

1. **The app manages context** for you.
2. **It orchestrates several model calls:** in Cursor, embedding models over your files, chat models, and models that apply diffs.
3. **It has an application-specific GUI.** He thinks this is underrated. Reading text is effortful; seeing a red and green diff and pressing a key to accept or reject is fast. The interface is how a person audits a fallible system.
4. **It has an autonomy slider.** Tab completion, edit a selection, edit a file, or let the agent loose on the whole repository. You choose how much autonomy to give up based on the task.

Perplexity shows the same pattern: it packages information, orchestrates models, cites sources you can check, and offers quick search, research, or deep research as settings on the slider.

TRY THIS

If you maintain a product, ask these questions of it. Can a model see everything a human user can see? Can it act in all the ways a human can act? Can a human supervise and stay in the loop? And for your particular domain, what is the equivalent of a diff? He asks what a diff looks like in Photoshop, and notes that most traditional software is full of switches designed only for humans.

1.4.1 Make the generate–verify loop fast

In this kind of work the AI generates and the human verifies, and you want that loop to spin as fast as possible. Two levers:

- **Speed up verification.** Visual interfaces use the fast “GPU” in your head rather than effortful reading.
- **Keep the AI on a leash.** A 10,000-line diff is not useful to him. It arrives instantly, but he is still the bottleneck, because he still has to check it for bugs and security problems.

WATCH OUT

Agents that are “overreactive” feel great when you are vibe coding and are a problem when you are trying to get real work done. His own practice is small, incremental chunks, one concrete thing at a time. A related trap: a vague prompt makes verification fail, which sends you round the loop again. Spending more time on a concrete prompt raises the chance that verification succeeds the first time.

His education example shows a leash built into product design. “Teach me physics” in a chat window does not work, because the model gets lost in the woods. Instead he separates it into two apps: one where a teacher builds a course, and one that serves the course to students. The course is an intermediate artifact that can be audited for quality and consistency, and it keeps the AI tied to a syllabus and a sequence of projects.

1.4.2 The lesson from self-driving

In 2013 a friend at Waymo gave him a thirty-minute drive around Palo Alto with zero interventions. It felt like self-driving was imminent. Twelve years later the problem is still not declared solved; the cars look driverless, but there is still a lot of teleoperation and human involvement.

AGAINST THE GRAIN

When he hears that 2025 is “the year of agents,” he gets concerned. He thinks this is the *decade* of agents. Software is tricky in the same way driving is tricky, a perfect demo is not a product, and humans need to stay in the loop. His phrase: this is software, let’s be serious.

His image for the right target is the Iron Man suit, which is both an augmentation Tony Stark drives and, in some films, an autonomous agent. At this stage, with fallible models, build *suits* rather than robots: partial-autonomy products with custom interfaces and fast verification, while designing them so the autonomy slider can move right over time.

1.5 Opportunity two: everyone is a programmer

Because the language is English, the five to ten years of study software used to require is no longer the entry fee. He finds this extremely bullish.

He tells the story of the vibe coding tweet: after fifteen years on Twitter he still cannot predict which posts will spread, and he expected this one to fizzle. It gave a name to something people felt but had not put into words, and now it has a Wikipedia page. He shows a video of children vibe coding and says he cannot watch it and feel bad about the future; he expects it to be a gateway to software development.

Two of his own projects: a basic iOS app in Swift, a language he does not know, running on his phone the same day; and MenuGen, which photographs a restaurant menu and generates pictures of the dishes, because he never knows what the items are.

WATCH OUT

The code for MenuGen was the easy part. The demo worked on his laptop in a few hours. Making it real (authentication, payments, a domain, deployment) took another week, and almost none of that was code. It was clicking through settings in a browser. The instructions for adding Google login told him which URL to visit, which dropdown to open, which option to choose. A computer was telling a person what actions to take. His reaction: you do it.

1.6 Opportunity three: build for agents

That frustration is the setup for the final section. There used to be two consumers of digital information: humans through GUIs, and programs through APIs. Agents are a third. They are computers, but human-like, and they need to use our infrastructure.

TRY THIS

Concrete ways to meet agents halfway:

- **Add an `llms.txt`** to your domain, the way `robots.txt` advises crawlers: a simple markdown description of what the site is. Parsing HTML is error-prone for a model.
- **Publish docs in markdown.** He names Vercel and Stripe as early movers.
- **Remove the word “click” from your docs.** A model cannot natively click. Vercel is replacing those instructions with equivalent `curl` commands an agent can run.
- **Support protocols aimed at agents**, such as Anthropic’s Model Context Protocol.

He likes small tools that make existing things model-readable just by changing a URL. Swapping “github” for “gitingest” concatenates a repository into one text file with a directory tree. DeepWiki goes further and has an agent write documentation pages for the repository. His example of the payoff: he wanted an animation in the style of 3Blue1Brown’s Manim library, pasted the full docs into a model instead of reading them, described what he wanted, and it worked first time.

Models can already browse and click, he acknowledges. But that is still expensive and unreliable, and a long tail of software will never adapt, so tools for that are needed too. For everything that can change, it is worth meeting in the middle.

PARTING ADVICE

The talk ends on a summary rather than advice, so this is the closest thing he said.

It is an amazing time to enter the industry, because a huge amount of code needs to be rewritten, by professionals and by people who now program in English. LLMs are part utility and part fab, but most of all an operating system, and it is still the 1960s. They are fallible people spirits we have to learn to work with, and our infrastructure has to adjust to them.

Over the next decade, he expects the autonomy slider to move from left to right. Build Iron Man suits now, and design them so they can become more autonomous later.

CHAPTER 2

Elon Musk: Aspire to Work, Not Glory



“Elon Musk: Digital Superintelligence, Multiplanetary Life, How to Be Useful”

Elon Musk CEO of SpaceX, Tesla and xAI; co-founder of Neuralink. Interviewed on stage by YC's Garry Tan.

Video: <https://www.youtube.com/watch?v=cFIIta1GkiE>

The interviewer frames the audience as 18 to 25 year olds, mostly engineers and AI researchers. Musk corrects the framing almost immediately: he prefers *engineer* to *researcher*. Unless there is a fundamental algorithmic breakthrough, he says, it is engineering. That small correction turns out to be a theme: he wants the simplest, lowest-ego words for things.

2.1 Starting out: useful, not great

Asked whether he ever felt he had to build something great, he says no. He wanted to build something *useful*, and thought it was probabilistically unlikely he would build anything great. He just wanted to try.

2.1.1 Zip2

In 1995 he had a place for graduate study at Stanford in materials science, working on ultracapacitors that might solve the range problem for electric vehicles. The alternative was something most people had not heard of, the internet. He asked his professor if he could defer for a quarter, since the startup would probably fail. The professor said this was probably the last conversation they would have. He was right.

The details he gives are about scrappiness. He wrote early online maps, directions, and white and yellow pages himself. He could not afford a web server, so he read the port directly. He could not afford a T1 line, so he drilled a hole through the office floor and ran a cable to the ISP downstairs. The office cost \$500 a month, so he and his brother slept there and showered at the YMCA.

WATCH OUT

The mistake he names from Zip2: too much shareholder and board control held by legacy media companies, who were also customers. They saw everything through the lens of legacy media and kept wanting to use the software in ways that made no sense for the new technology. He believed Zip2's technology was better than Yahoo's, but its customers constrained how it was used.

He also notes he did not set out to start a company. He sent his résumé to Netscape and got no reply. He hung around the Netscape lobby hoping to bump into someone and was too shy to talk to anyone. Since he could not get a job at an internet company, he started one.

2.1.2 Keeping the chips on the table

Zip2 sold for about \$300 million, and his share was about \$20 million. At the time he had about \$10,000 in the bank and lived with four housemates; the cheque came in the post. He put almost all of it into X.com, which merged with Confinity to become PayPal. The motive was the Zip2 lesson: go direct to consumers so your wings are not clipped.

2.2 SpaceX: a small chance is better than none

After PayPal he wondered when NASA planned to send people to Mars, and found no date on the NASA website. He assumed it was hard to find. In fact there was no plan.

His first idea was philanthropic: land a small greenhouse on Mars and get a picture of green plants on a red background to inspire public interest. On the way he went to Moscow in 2001 and 2002 to try to buy decommissioned ICBMs as launch vehicles, and found the Russians kept raising the price.

THE ONE THING

What he eventually realized was that the problem was not a lack of *will* to go to Mars. It was that there was no way to do it without breaking even NASA's budget. So the thing to fix was the cost of rockets, and that needed a company.

AGAINST THE GRAIN

He started SpaceX in 2002 believing it had perhaps a 10% chance of success, maybe 1%. No rocket startup had ever succeeded. His reasoning: the big defense contractors would not advance rocket technology because they do what the government asks and the government wants conventional things, so progress would come from a startup or not at all. A small chance of success beats no chance. He told recruits plainly that they would probably die.

He became chief engineer not by choice but because no good chief engineer would join something that risky. The first three launches failed. If the fourth had failed, he had no money left. It worked.

2008 was the low point. By mid-year SpaceX's third launch had failed, the Tesla financing round had failed, and Tesla was going bankrupt. The press called him an "internet guy" attempting rockets, which, he concedes, did sound improbable. Just before Christmas, NASA called to award a space station resupply contract, and he blurted out "I love you guys." The Tesla round closed at 6pm on 24 December 2008, the last hour it could; payroll would have bounced two days after Christmas.

2.3 How to be useful

The interviewer points out that most people in the room have never managed anyone and asks what he would tell them. His answer is his core advice for the whole session.

THE ONE THING

Try to be as useful as possible. It is very hard to be useful, especially to many people. Think of the total as area under a curve: how useful you have been to each person, multiplied by how many people. It resembles the physics definition of work. Don't aspire to glory; aspire to work. If this thing succeeds, how useful will it be, and to how many people?

2.3.1 Keep the ego-to-ability ratio low

Whatever your role, do whatever it takes, whether the task is grand or humble, and internalize responsibility. He names a specific failure mode.

WATCH OUT

When your ego-to-ability ratio climbs well above one, you break your feedback loop with reality. In AI terms, you break your reinforcement learning loop. Keep that loop strong by minimizing ego and owning outcomes. This is also why he does not want to call xAI a "lab." He wants the simplest, lowest-ego terms, and to close the loop on reality hard.

2.4 First principles, with worked examples

He explains first principles as breaking a problem into the fundamental elements most likely to be true, then reasoning up from those, rather than reasoning by analogy. A second tool from physics is *thinking in the limit*: ask what happens if you minimize or maximize something. He calls these tools a superpower and says they apply to any field.

TRY THIS

Price something from its materials. The usual way to estimate what a rocket should cost is to look at what past rockets cost. The first-principles way is to list what a rocket is made of (aluminium, copper, carbon fibre, steel), how much of each, and the price per kilogram. That sets the floor. He found raw materials were perhaps 1–2% of a rocket's historical cost, which means manufacturing had to be very inefficient. That is before considering reuse.

His AI example is xAI's training cluster in Memphis. At the start of the previous year they needed 100,000 H100 GPUs training as one coherent system. Suppliers said 18 to 24 months. They needed six. So they broke "a data center" into its parts:

- **A building.** No time to build one, so they used a disused Electrolux factory.
- **Power.** The site had 15 megawatts; they needed 150. They rented generators and put them on one side of the building.
- **Cooling.** They rented about a quarter of the mobile cooling capacity in the US and put the chillers on the other side.
- **Power smoothing.** Training load can drop by 50% in 100 milliseconds, faster than generators respond, so they added Tesla Megapacks and modified their software to smooth it.
- **Networking.** Cabling 100,000 GPUs to train coherently was very hard. They ran it in four shifts around the clock, and he slept in the data center and did cabling himself.

They then doubled it. He says the Memphis site now holds 150,000 H100s, 50,000 H200s and 30,000 GB200s, with 110,000 more GB200s coming online at a second site.

The interviewer suggests hardware uniquely demands this kind of pushback, since software can paper over problems. Musk disagrees: the principles apply to anything. He only used hardware as the example.

2.5 On AI

2.5.1 What makes a competitive foundation model

His list: the talent of the people; the scale of the hardware and, separately, how well you can bring it to bear, since you cannot just plug GPUs in; unique access to data; and distribution.

On data, he agrees with Ilya Sutskever that high-quality human-generated pre-training data runs out fast. The work shifts to synthetic data, and the hard part is judging whether synthetic data is real or a hallucination that does not match reality. Grounding is tricky. He mentions Grok 3.5 as focused on reasoning, and agrees with the interviewer that physics textbooks are useful for reasoning while social science is not.

2.5.2 Robots, and why he stopped dragging his feet

He predicts far more humanoid robots than all other robots combined, perhaps by an order of magnitude, and eventually five to ten times as many humanoid robots as humans.

He admits he held back on AI and humanoid robotics for years because he did not want to “make Terminator real.” What changed was realizing it would happen whether he took part or not. You can be a spectator or a participant, and he would rather participate.

2.5.3 Safety

He expects several deep intelligences rather than one runaway system: at least five, perhaps ten, with maybe four in the US. He believes digital superintelligence, meaning smarter than any human at anything, is very close: this year, or next year for sure.

He broadly agrees with Geoffrey Hinton’s estimate of a 10–20% chance of annihilation, and chooses to emphasize the 80–90% chance of a great outcome.

AGAINST THE GRAIN

His one safety principle, which he repeats several times: rigorous adherence to truth, even when the truth is politically incorrect. His intuition is that AI becomes dangerous if you force it to believe things that are not true. Alongside that, empathy for humanity and life.

2.6 Mars, the long view, and Neuralink

He uses the Kardashev scale as a measure of civilization’s progress. By his estimate humanity uses perhaps 1–2% of Earth’s energy, so we are far from Type I (a planet’s energy), let alone Type II (a star’s) or Type III (a galaxy’s). We are at the very early stage of what he calls the intelligence big bang.

He expects enough mass on Mars within roughly thirty years for it to be self-sustaining even if supply ships from Earth stop. His reason for caring comes from the Fermi paradox: if intelligence is extremely rare, consciousness may be a tiny candle in a vast darkness, and becoming multiplanetary greatly extends its likely lifespan. Two planets also create a forcing function to improve space travel. The great filters to avoid include global thermonuclear war.

On Neuralink, he says it is not needed to reach digital superintelligence, which will come first. What it can do is fix bandwidth. Human sustained output is under one bit per second across a day. Five people have received implants that read signals, letting people with paralysis control computers and phones.

He expects the first vision implants, which write to the visual cortex, within six to twelve months; one monkey has had one for three years. Longer term, he expects implants to augment rather than just repair: seeing infrared, ultraviolet or radar.

Asked what happens to human relevance, he says it is called the singularity because we cannot see past it. At some point human intelligence will be under 1% of all intelligence, and at Kardashev Type II perhaps a billionth, even if everyone's IQ is augmented to a thousand.

PARTING ADVICE

Asked what the smartest technical people of this generation should be working on tonight:

If you are doing something useful, that is great. Just try to be as useful as possible to your fellow human beings, and you are doing something good. And, again, focus on super-truthful AI, because that is the most important thing for AI safety. His hope is that AI helps us understand the nature of the universe: where the aliens are, how it began and will end, and which questions we do not yet know to ask.

CHAPTER 3

Sam Altman: Don't Build the Five Ideas Everyone Has



"Sam Altman: The Future of OpenAI, ChatGPT's Origins, and Building AI Hardware"

Sam Altman CEO, OpenAI; former president of Y Combinator. Interviewed on stage by YC's Garry Tan.

Video: <https://www.youtube.com/watch?v=V979Wd1gmTU>

3.1 Deciding to start at all

Asked which small early decisions turned out pivotal, Altman says the biggest was simply deciding to do it. OpenAI very nearly did not start. Through 2015 it was "coin flippy." AGI sounded crazy, he had Garry Tan's current job running YC with plenty of other good things to do, and DeepMind seemed impossibly far ahead.

THE ONE THING

Many ambitious things seem so difficult, with such good reasons not to do them, that they only happen because a core group of people sit in a room, look each other in the eye, and say "let's do this." Those moments matter. When in doubt, lean into them.

He stresses how improbable it looked. Ten years ago there were no working language models. OpenAI was trying to play video games and had a robot hand that could barely manipulate a Rubik's cube. There were no product ideas, no revenue, and no expectation of revenue, just people at whiteboards trying to think of papers to write. ChatGPT was science fiction.

3.1.1 Why the crazy mission made hiring easier

Gathering top researchers for something that sounded crazy was easier than it sounds. Saying you will go for AGI made 99% of the world think you were mad, and deeply resonated with the other 1%. That 1% contained a lot of very smart people, and there was nowhere else for them to go.

AGAINST THE GRAIN

If you are doing the same thing as everyone else, it is very hard to concentrate talent or get people to believe in a mission. A one-of-one idea gives you a real tailwind, precisely because it looks unlikely.

3.1.2 Nothing big starts big

To anyone wondering whether to try an OpenAI-scale company from day one: OpenAI was not that. It was eight people in a room, then twenty, unclear what to do, trying to write a good paper. He quotes Vinod Khosla: a zero-million-dollar startup and a zero-billion-dollar startup both have zero revenue and a few people in a room trying to get the first thing to work.

TRY THIS

His only advice for starting something big: pick a market where there is some version of the future in which it could be big if it works. After that, it is one dumb foot in front of the other for a long time.

3.2 The product overhang

THE ONE THING

Right now what the models can do is far ahead of the products people have figured out how to build. Even if the models got no better, which they will, there is a huge amount of new stuff to build. He calls reasoning models a new square on the periodic table that almost no one has built with yet; only in the past month has he seen startups designing specifically around them.

Two supporting points. Prices are falling fast: o3 cost five times as much the week before the talk, and he expects people to be astonished at how quickly price per performance falls. And an open-weight model was coming that he expected to be much better than people hoped, capable of running very powerful models locally.

3.2.1 Where ChatGPT is heading

Memory is his favourite feature of the year, even if others at OpenAI might pick something else, because it shows the direction: an entity that gets to know you, connects to all your stuff, and helps proactively. Not message in, message out, but something running all the time that knows when to message you and when to act for you, integrated with your services and new devices.

Asked when the film *Her* arrives, he says gradually. It is a little here with memory, more when the assistant runs persistently in the background, and more again with the first new device. The key to *Her* was never the hardware; it was a companion running in the background.

He expects compute to be a mix of cloud and local, and says OpenAI would be delighted to push half of ChatGPT's workload onto users' devices, because it is about to run the largest, most expensive infrastructure in the world. ChatGPT went from not existing to the fifth-largest website in two and a half years, which is hard to support no matter what.

On models, the eventual goal is one integrated model that reasons when needed and generates real-time video when needed; for example, thinking hard, researching, and writing a brand-new app just for you. He does not expect GPT-5 to get all the way there. Robots come after: nail the model, then connect it to a body. He jokes about giving a free humanoid robot with the top ChatGPT tier, and asks an open question he finds interesting: how many humanoid robots, built the old-fashioned way, would it take to run the whole supply chain, from mining to shipping to foundries, and build the rest?

On American manufacturing, his instinct is that the usual policy answers have been tried for a long time and have not worked, so something new is needed. AI and robotics offer a genuinely new way to bring complex industries back, and that is at least worth trying.

3.3 How not to get run over by OpenAI

This is, the interviewer says, the most-asked question in the event's chat.

Altman's answer has two parts. First, OpenAI is going to build the best super-assistant it can inside ChatGPT, and it makes the company sad when people start a startup to make another ChatGPT. That is one small part of the opportunity. He floats ways OpenAI could help others: ChatGPT sending traffic to startups through some kind of app or agent store, or "sign in with OpenAI" so users bring their personalized model to new products.

WATCH OUT

The second part is a problem as old as startups. People get excited about the same thing at the same time, because we are social creatures. He bets that if Tan listed the five AI ideas he hears most, half the room would be working on one of them. And he bets the person in the room who eventually builds something bigger than OpenAI is not working on any of the five.

It is hard to be defensible when everyone is doing the same thing. The most enduring companies are usually doing something different, and that buys time to find the great product and build the technology *before* you have to answer the defensibility question.

He is candid that ChatGPT had no defensibility strategy for a long time. It was simply the only good product. Then it became a well-known brand, and later gained genuinely defensible things like memory and connections. You get a window, and you have to build defensibility before it closes.

3.3.1 "Just-in-time software" and SaaS

Tan describes a debate at YC: a future where a company is a database plus an API layer enforcing access control and business logic, and the interface is an LLM generating software on the fly. Altman says flatly that it is going to happen.

THE ONE THING

This is the best time in the history of technology to start a company, and part of why is that the ground is shaking. You can look at a change like this and ask what it means for your SaaS company, or you can notice it will happen to everyone. Startups win by iterating faster and cheaper than big companies, and when the industry's clock cycle changes this much, startups almost always win. It has probably never changed this much.

3.4 Conviction when people say you are wrong

Asked how the team handled being told scaling laws were wrong, he rejects the heroic version.

AGAINST THE GRAIN

It is hard to hold conviction when many people tell you you are wrong, and anyone who says it is easy is not being honest. It gets easier over time. His example: a few years into OpenAI, Elon Musk, then a hero of his, emailed after seeing GPT-1 to say it was crap and they had a zero percent chance. Altman went home thinking: what if he's right? He has no magic answer. You get knocked down, get up, brush yourself off, and keep going.

3.5 Agents, interfaces and devices

ChatGPT used to be a Google replacement: something about as long as a search query, or maybe half an hour of searches assembled together. Now tools like Codex and deep research take a task, go off, and come back with a proposal, like a very junior employee working for a short while. Consider how much of the world's work happens in front of a computer in few-hour chunks that someone then signs off on. It is a lot.

On interfaces, he thinks science fiction got one thing right: the interface melts away. Voice seems poor today because it does not work well, but imagine telling a computer what should happen today, trusting it to handle changes without interrupting you, and hearing from it only when a great human assistant would. Using his phone, by contrast, feels like walking through Times Square being bumped into: notifications, pop-ups, bright colours. He loves the phone and finds it stressful.

He will not describe the device publicly. On Jony Ive: there have been perhaps two big revolutions in computer interfaces in fifty years, keyboard-mouse-screen and then touch, and AI opens the field for a third. If you had to bet on one person to figure it out, Ive is the obvious bet.

3.6 What changes for people

The whole arc of technology, in his telling, is one story: society builds the scaffolding a little higher so one person can do much more than before. What will feel most different about the next ten years is how much a single person or small team with a lot of agency can get done. That is bigger than it sounds, because coordination costs are huge; remove them and you get a step change, not a little more output.

He tempers the OpenAI story by pointing at the tens of millions of people through history, from digging rocks out of the ground to semiconductors to the internet, whose work let a few tens of people have that impact.

On the “3,000 people in front of you” versus the billions who have not tried AI, or who think it just hallucinates: AI is somewhat mainstream, but most people think of it as a chatbot and have not grasped what comes next. It is a privilege to live a little in the future and build for everyone else coming along.

3.7 Hiring and running OpenAI

TRY THIS

Hire people who are really smart, driven, curious, self-motivated, hard-working, have a record of accomplishment, work well on a team, and are aligned with the company's direction. That gets you 90% of the way. He is surprised how much people focus on anything else.

WATCH OUT

Early in a startup he does not believe in hiring the senior, eminent administrator with the polished twenty-year record, and says YC has not had much success with it either. Take young and scrappy but clearly gets things done. Reading YC applications, he ignored résumé items and went straight to “what is the most impressive thing you’ve done,” only checking the résumé as a backup. He cites Paul Buchheit’s line: hire for slope, not y-intercept.

On being CEO of OpenAI: “I don’t recommend it.” No single challenge is that hard. It is the number of things at once, the number of large companies coming after them, and constant switching between huge, unrelated decisions. More context than he thought one person could hold.

3.8 Science and energy

Pressed for one thing he is most excited about in 10 to 20 years, by which point he expects “unimaginable superintelligence” unless something goes badly wrong: AI for science. His belief is that, to a first approximation, all long-run sustainable economic growth comes from discovering new science, plus institutions good enough to develop and share it. Speed up discovery and the gains compound.

He admits something he finds embarrassing. He was obsessed with energy and AI as separate interests for years, and did not realize until after starting OpenAI that energy would be the fundamental limit on how much intelligence we can have. He had thought of them as orthogonal. Now he thinks about how much energy we can generate on Earth before heating the planet, and how long before GPUs have to go to space. What drew him to energy in the first place was the long historical correlation between quality of life and the abundance and cost of energy.

He says he would like to attend a degrowth conference once, to experience a mindset he has never felt, but he is “pretty sure we’re right and they’re wrong.” His preferred historical analogy for AI is the transistor: a scientific discovery that the economy then turned into enormous value over a few decades. He expects this to be faster and steeper. Five years ago GPT-3 could barely write a sentence; he expects the same rate of progress for five more years.

PARTING ADVICE

Asked what he would tell himself as a founder in YC’s first batch in 2005:

He wishes someone had taught him the importance of conviction and resilience over a long period. It is easy for a while, then your reserves wear down, and nobody talks about how hard that is. His first startup did not work well; many people give up after one failure, and they should not. Develop trust in your own instincts and grow it as your judgment improves. Have the courage to work on things that are out of fashion but that you believe in.

And he compares it to having a child, which he recently did. Everyone says it is the best and hardest thing you will ever do. That is true, and it is what being an entrepreneur is like: the good parts are better than you can imagine, the hard parts are shockingly harder, and you have to keep going.

Andrew Ng: Speed Is the Strategy



“Andrew Ng: Building Faster with AI”

Andrew Ng Founder, AI Fund and DeepLearning.AI; co-founder, Coursera. A solo talk with slides, followed by audience Q&A.

Video: <https://www.youtube.com/watch?v=RNJCfif1dPY>

AI Fund is a venture studio that builds about one startup a month. Because it co-founds them, its people write code, talk to customers, design features and set prices, so Ng has done many reps inside startups rather than watching from outside. The whole talk hangs on one theme.

THE ONE THING

Execution speed is a strong predictor of a startup’s odds of success. New AI tools let startups go much faster, and the best practices for doing so are still changing every two to three months.

4.1 Where the opportunities are

He draws the AI stack: semiconductors at the bottom, then cloud hyperscalers, then foundation model companies, then applications. Most media attention goes to the technology layers.

AGAINST THE GRAIN

Almost by definition, the biggest opportunities are at the application layer, because applications have to generate enough revenue to pay for all the layers underneath. There are opportunities at every layer, but social media's focus on the lower ones is misleading for founders.

4.1.1 Why agentic workflows matter

Asked for the most important tech trend, he names agentic AI, while noting that marketers turned the word into a sticker on everything and nearly emptied it of meaning. His technical case:

Prompting a model for an answer is like asking a person to type an essay from first word to last in one go, never pressing backspace. People do not write their best that way, and neither do models. An agentic workflow instead asks for an outline, then web research if needed, then a draft, then a critique of the draft, then a revision, and so on around a loop. It is slower, and the result is much better.

Across AI Fund projects in compliance documents, medical diagnosis and legal reasoning, he says this has often been the difference between working and not working. A new *orchestration layer* has appeared in the stack to coordinate many calls to the layers below. It makes applications easier to build, and his conclusion about applications still holds.

4.2 Lesson one: work on concrete ideas

AI Fund only works on concrete ideas, meaning an idea specified in enough detail that an engineer can go and build it.

- **Vague:** use AI to optimize healthcare assets. Different engineers would build totally different things.
- **Concrete:** software that lets patients book MRI machine slots online to optimize usage. It may be a good or a bad idea (some businesses already do it), but an engineer can build it quickly, and you will find out.
- **Vague:** AI for email productivity. **Concrete:** a Gmail integration that uses the right prompts to filter and triage email. He could build that in an afternoon.

WATCH OUT

Vague ideas get a lot of praise. Tell your friends you want to use AI to optimize healthcare assets and everyone says it is a great idea. When you are vague you are almost always right; when you are concrete you may be right or wrong. Either is fine, because concreteness lets you discover which, fast.

4.2.1 Where good concrete ideas come from

Usually from someone who has thought about a problem for a long time. Before Coursera he spent years thinking about online education and talking to users. YC calls this wandering the idea maze. After that, an expert's gut becomes a surprisingly good, instantaneous way to decide between features.

AGAINST THE GRAIN

You might expect an AI person to say "we need data." He loves data, but for many startups gathering it is a slow way to decide. A subject matter expert with a good gut is often the faster mechanism.

A startup cannot hedge across ten ideas. Pursue one clear hypothesis doggedly. If the data tells you to lose faith, pivot on a dime to a completely different concrete idea and pursue that with equal determination.

WATCH OUT

If every new piece of data makes you pivot, you are probably starting from too weak a base of knowledge. If every customer conversation changes your mind, you do not yet know the sector well enough to have a high-quality concrete idea. Finding someone who has thought about it longer may get you onto a better path.

4.3 Lesson two: prototypes are now nearly free

Many application startups fail not because they cannot build what they want, but because nobody cares. So the core loop is: build (engineering), get user feedback (product management), adjust, and repeat until product–market fit. AI coding assistants are changing the speed and cost of the build step.

He splits his own software into two buckets:

- **Production software** in large, maintained codebases. Rigorous data is scarce, but AI assistance might make this 30–50% faster.
- **Quick and dirty prototypes.** Easily ten times faster, perhaps much more. Prototypes need less integration with legacy systems and data, and have much lower requirements for reliability, scalability and security.

TRY THIS

He routinely tells his teams to go ahead and write insecure code, if it only runs on your own laptop and you are not planning to hack yourself. Once it works, make it secure and scalable before anyone else touches it, because leaking personal or sensitive data is very damaging. With prototypes this cheap, startups can systematically build twenty to see what works. It is fine for most proofs of concept never to reach production.

He does not accept that “move fast and break things” earning a bad reputation means you should not move fast. His version is *move fast and be responsible*.

4.3.1 Tools, and code as a disposable artifact

He traces the tools: autocomplete popularized by GitHub Copilot, then AI-enabled IDEs like Cursor and Windsurf, then, in the last six or seven months, highly agentic coding assistants. He names o3 and Claude Code (“fantastic” since Claude 4), adding that he may use something else in a few months. Being even half a generation behind on tools makes a big difference.

THE ONE THING

Code used to be a valuable artifact because it was hard to create. It is much less so now. He has been on teams that rebuilt a codebase from scratch three times in a month. In Jeff Bezos’s terms, choosing a tech stack and database schema used to be a one-way door, costly to reverse. It is not fully a two-way door now, but his teams will build on a stack, change their minds a week later, and start over. Rethink which of your doors are still one-way.

4.3.2 Everyone should learn to code**AGAINST THE GRAIN**

The advice over the past year not to learn to code because AI will automate it will, he thinks, be seen as some of the worst career advice ever given. When coding got easier (punch cards to terminals, assembly to COBOL, text editors to IDEs) more people learned it. People even wrote papers claiming COBOL meant programmers were no longer needed. His controversial view is

that everyone, in every role, should learn to code. At AI Fund his CFO, head of talent, recruiters and front desk person all can, and he sees each of them do their jobs better for it.

His illustration is about knowing what to ask for. Making course art with Midjourney, a colleague who knew art history could prompt with genre, palette and artistic influences and got far better images than Ng, who could only ask for “pretty pictures of robots.” One of the most important future skills is telling a computer exactly what you want. Learning to code, in the sense of steering AI to code for you, will remain the best way to do that for a long time.

4.4 Lesson three: the bottleneck moves to product

As engineering speeds up, deciding what to build and getting user feedback become the bottleneck. More of his teams now complain they are held up by product management and design.

Silicon Valley used a rough rule of about one product manager to four to seven engineers. The day before the talk, one of his teams proposed a headcount of one PM to half an engineer: twice as many PMs as engineers, a first in his career. He does not yet know if it is a good idea, but thinks it shows where things are going. PMs who can code, and engineers with product instincts, tend to do better.

TRY THIS

His menu of feedback tactics, from fastest and least accurate to slowest and most accurate:

1. Look at the product yourself and go by your gut. Surprisingly good if you are an expert.
2. Ask three friends or teammates to try it.
3. Ask three to ten strangers.
4. Send prototypes to a hundred testers.
5. Send it to more users.
6. A/B test.

On strangers: learn to find places with high foot traffic, such as coffee shops and hotel lobbies, and very respectfully ask people to look at what you are building. People in coffee shops who do not want to be working are often happy for the distraction. He has made many product decisions this way.

AGAINST THE GRAIN

Contrary to Silicon Valley lore, A/B testing is now one of the *slowest* tactics on his list, because shipping takes time and results depend on having enough users.

THE ONE THING

Do not just use an A/B test to pick A or B. Sit down with the data and ask why your prediction was wrong, and update your mental model of the user. The real goal of the slow tactics is to improve your gut, so that the fastest tactic gives better answers next time.

4.5 Lesson four: understanding AI is itself a speed advantage

For mature technologies like mobile, even non-technical people have good instincts about what an app can do. Mature functions like marketing, HR and legal are hard and important, but that expertise is widely available because it has not changed dramatically in six months. AI knowledge is not yet diffused, so teams that really understand it have an edge.

WATCH OUT

Many AI decisions (what accuracy a support chatbot can reach, whether to prompt or fine-tune, how to get voice latency down) can be solved in days if you choose right, or send you down a blind alley for three months if you choose wrong. In theory a binary architecture choice is one bit of information, so not knowing should make you at most twice as slow: just try both. In practice, flipping the wrong bit costs something like ten times as long.

He lists the building blocks that appeared in two years: prompting, agentic workflows, evals, guardrails, RAG, voice, async programming, ETL, embeddings, fine-tuning, graph databases, model integration. His image is Lego. One brick lets you build some things; each new colour of brick adds combinations, so what you can build grows combinatorially. Learning more blocks lets you build software that nobody could have built a year ago.

4.6 From the Q&A**4.6.1 On hype**

He thinks AGI has been overhyped, and for a long time there will be many things humans can do and AI cannot. The most powerful people will be those who can make computers do exactly what they want.

His filter for hype: over two years a handful of companies promoted narratives that made them look more powerful, for PR, fundraising or influence, and because the technology was new nobody fact-checked them. He lists examples he considers false: that AI might accidentally cause human extinction; that soon nobody will have a job; that training a new model will casually wipe out thousands of startups (a few, like Jasper, were hurt, but it is not that easy); that only nuclear power can meet AI's electricity needs. On GPUs in space: go for it, but terrestrial GPUs have plenty of room to run.

AGAINST THE GRAIN

He rarely uses the term "AI safety," not because he wants dangerous things built but because safety is not a property of a technology; it is a property of how it is applied. An electric motor maker cannot stop the motor being used in a smart bomb. AI is neither safe nor unsafe. He prefers *responsible AI*. He also sees exaggerated dangers used as a weapon against open source, and cites California's proposed SB 1047 as regulation that would have made no one safer while making open weights much harder to release. The risk he worries about is a few gatekeepers, like Android and iOS on mobile, whose permission everyone needs.

4.6.2 On moats and token costs

Asked how to build a business when anything can be copied in hours: the number one question is whether you are building a product users love. Go-to-market, pricing, competitors and moats all matter, but come after. He thinks moats are overhyped; more businesses start with a product and grow a moat later. Brand matters more in consumer, channels in enterprise. AI Fund writes a two-to-six-page memo on these before committing. Right now the number of things that could be built far exceeds the number of people able to build them.

TRY THIS

On token costs: to a first approximation, do not worry. Very few startups have enough usage for it to be a problem, and when it happens there are engineering fixes. More useful: architect for easy switching between providers. On some of his products he does not know which model is in use, because the team runs evals when a new model comes out and switches if it scores better, sometimes without telling him. Switching models is cheap; switching orchestration platforms is harder.

4.6.3 On education

Everyone feels change coming to edtech, but he does not think the disruption has arrived. Language learning is clearest. For the rest there is lots of experimentation and no clear end state. He expects education to be hyper-personalized, but whether through avatars, chatbots or something else is unknown. Real work is complex, and mapping it to agentic workflows will take the next decade.

PARTING ADVICE

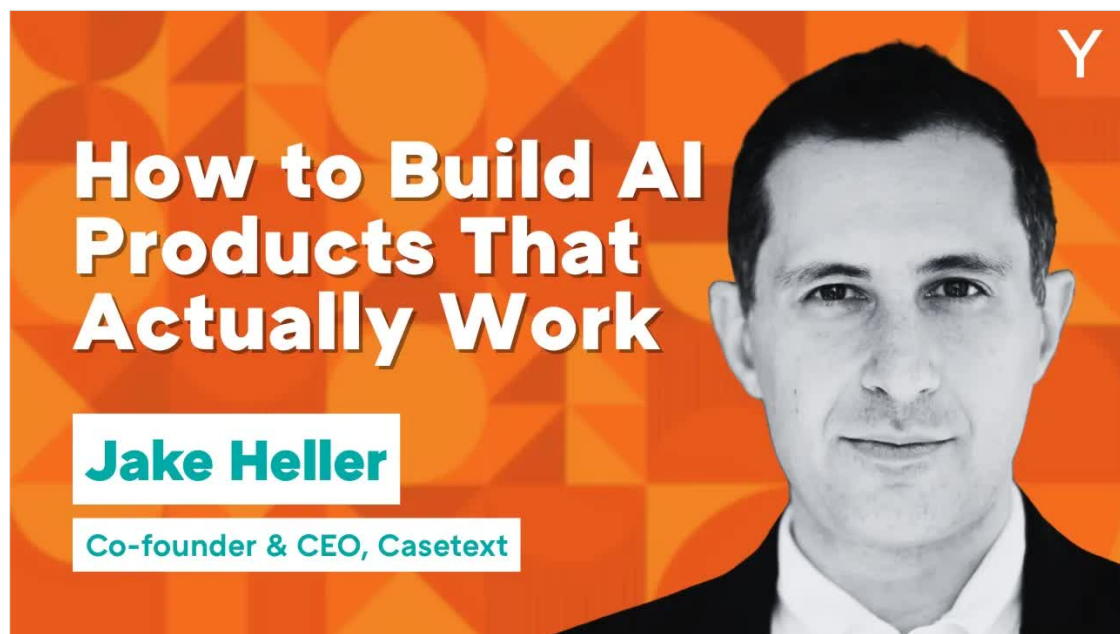
Asked how builders should balance speed against societal harm:

Look in your heart. If you do not think what you are building will make people at large better off, do not do it. It sounds simple and is hard in the moment. AI Fund has killed several projects with solid economics purely on ethical grounds, because they did not want those things to exist.

And bring everyone along. People in non-engineering roles who know AI, and can code, are far more productive than those who do not, so help everyone learn to build with it.

CHAPTER 5

Jake Heller: Pick a Job, Then Grind the Evals



“From Idea to \$650M Exit: Lessons in Building AI Startups”

Jake Heller Co-founder and CEO, Casetext (acquired by Thomson Reuters). A solo talk with slides, followed by audience Q&A.

Video: <https://www.youtube.com/watch?v=l0h3nAW13ao>

Heller’s name comes from the video thumbnail; the captions do not include it. He grew up coding, became a lawyer, and like many builders who enter old professions was astonished at how the work was done. He founded Casetext in 2013, and notes that is roughly how long success can take: be ready for a long adventure.

Casetext worked on applying natural language processing to law long before it was called AI. Because it was deep in language models, it got early access to GPT-4 in the summer of 2022. At that point it had about \$20 million in revenue and a hundred people. It stopped everything and built a new product, CoCounsel, an AI assistant for lawyers. Thomson Reuters acquired the company for \$650 million in cash. He expects many in the room to look back and find that number small.

The talk has three parts: which idea to pick, how to build it, and how to sell it.

5.1 Picking the idea

YC's motto is "make something people want," and it exists because knowing what people want used to be genuinely hard: build, ship, fail, repeat.

THE ONE THING

It just got much easier. What do people want? What they are *already paying people to do*: customer support, insurance adjusting, paralegal work, personal training, executive assistance. Much of that can now be done by language models, and much physical work by robotics.

He sees three categories:

1. **Assist a professional.** What CoCounsel did: help lawyers read documents, research, review and mark up contracts.
2. **Replace the work.** People hire lawyers; become an AI-powered law firm. The same goes for accountants, financial experts, physical therapists, laundry folding.
3. **Do the previously unthinkable.** No law firm would ever pay people to read, categorize, summarize and index hundreds of millions of documents. Thousands of instances of a cheap model can.

5.1.1 The market got a thousand times bigger

Total addressable market used to be the number of seats you could sell times something like \$20 a month. Many billion-dollar companies were built that way. Now the budget you are competing for is the combined salaries of the people doing the job, and companies pay professionals thousands or tens of thousands a month. He puts the increase at 10, 100 or even 1,000 times.

He anticipates this sounding dystopian and argues the opposite, for two reasons. First, the lamplighter example from a Sam Altman essay: moving past today's jobs unlocks a future we cannot imagine, and today's work will look antiquated. Second, access. He says over 85% of low-income people do not get legal help, because lawyers are too slow and expensive. Make lawyers far faster and cheaper, or provide the service directly, and people who were turned away can be served. Everyone should have the best financial adviser and personal assistant, as everyone can already have a strong coding assistant.

5.2 Building it reliably

He says what follows will sound obvious, and that almost nobody does it. The key word is *reliable*, which is what separates a cool demo from something that works in practice.

5.2.1 Find out what the best professional actually does

TRY THIS

Ask what a professional in the field really does, and be very specific. Do not make it up. Casetext's founders were lawyers, and 30–40% of the company, including coders, were too. If that is not you, go work undercover inside such companies, or find a co-founder with deep domain expertise.

Then ask: how would the *best* person in this field do the task with unlimited time and resources, say a thousand AIs working in parallel? Work backwards from that into steps.

His example is a version of deep research Casetext built right after getting GPT-4. A top lawyer given a research question would: work out what it really means, perhaps asking clarifying questions; make a research plan; run dozens of searches returning hundreds of results; read each carefully, discarding irrelevant ones and taking notes on why each matters and where it fits; write the answer; and finally check that the answer cites the right sources accurately.

5.2.2 Turn steps into prompts or code

Most steps become one or more prompts, because they used to require human-level intelligence. For example: rate this opinion's relevance to the question from zero to seven; write the essay from these notes; check whether this footnote accurately cites this source.

WATCH OUT

If a step can be deterministic code or a calculation instead of a prompt, make it code. Prompts are slow and tokens are still expensive.

If an expert always follows the same steps, make it a fixed workflow. Much of CoCounsel was exactly that, and it needed no framework, just Python functions feeding into each other. If the expert's approach depends on circumstances, you need something more agentic, which is harder to make good. In both cases, do not fly blind; really know how the work is done.

5.2.3 The hard part is getting it right

Building it is simple. Knowing whether the research was done well, the document read correctly, or the insurance claim adjusted properly is hard. This is where evaluations come in, and it is what he sees most people skip.

WATCH OUT

A demo that is 60–70% accurate can raise a decent round and even sign a few pilot customers. Then it does not work in practice, and the excitement, capital and pilots fall apart. Models, like people, have bad mornings and sometimes output nonsense.

TRY THIS

How to write evals.

- Start from domain expertise: what does good look like, for the whole task and for each micro-task (which search queries are good, what must be pulled from a document and from which page)?
- Make answers objectively gradable where possible: true or false, or a number from zero to seven.
- Use a framework; he likes the open-source promptfoo. For this input and prompt, the answer should be six.
- Write a dozen cases matching what customers will really send. Get them perfect, then 50, then 100, tweaking the prompt until it passes.
- Keep a held-out set you do not look at while prompting, so you do not overfit.

You can go a long way with prompting alone, without fine-tuning. Failures turn out to be predictable: ambiguous instructions, or a consistent drift in one direction. Add instructions and examples to pull the model back from each class of error.

THE ONE THING

The biggest qualification for success is whether whoever writes your prompts is willing to spend two sleepless weeks on a single prompt. At first you pass 60%, and most people give up and decide AI cannot do the task. After a night you reach 61%, and the next group gives up. Two solid weeks of prompting, adding evals and tweaking gets you to around 97%, and the remaining 3% are judgment calls that humans make differently too.

His rough bar before beta: about 100 tests per prompt and 100 for the whole task, passing around 99. A thousand is ten times better but hard. Then ship to customers as a beta, set expectations that it is not perfect, and turn every complaint into a new test, either from logged data or by asking the customer to share the document and question. Casetext now has more evals from real customer failures than from the lab. Customers do baffling things; real search queries are barely legible.

It is never finished. Try new models as they come out, since frameworks score them against your tests automatically. A single word can move accuracy by a percent, which matters a great deal in finance, medicine or law. There should be a pull request on your prompts every day or two.

AGAINST THE GRAIN

Do just these two things (learn how professionals really do the job and break it into steps; test each step and the whole workflow) and you will be 90% of the way to a better AI app than most of what exists. Most people never write evals and never learn the work. Some of them make flashy demos on Twitter, raise money, and briefly become heroes. Be careful who you choose as heroes. The real ones are quietly making their product better every day.

5.3 Marketing and selling

This is the hardest part, and one he says his team is still working out inside a multibillion-dollar company.

AGAINST THE GRAIN

Many board members will tell you product matters less than sales and marketing. For ten years Casetext had an okay product and went through several qualified sales and marketing leaders, with okay results. When it had an awesome product, word of mouth and press arrived for free because it was genuinely new, and the existing salespeople became order-takers. The most important sales and marketing move is an amazing product, plus making sure the world hears about it. Push back on board members who say otherwise.

Three pieces of advice:

1. **You may not be selling software.** The companies that excite him deliver the service itself, perhaps with a human in the loop. Reviewing a contract might cost \$1,000 at a law firm, so charge \$500 per contract rather than \$20 a month. Price on value.
2. **Ask customers how they want to pay.** Casetext considered usage pricing, but customers said they would rather pay more for predictable annual budgets. They chose \$6,000 per seat per year (\$500 a month). Give them that.
3. **Close the trust gap.** Boards are asking CEOs what they are doing about AI, so companies want to try things. But they know how to hire, fire, train and coach people, and have no idea what to expect from your product. Build trust with side-by-side comparisons: keep your law firm or accountant, run ours in parallel, and compare speed, quality and differences. Also run studies and pilots.

WATCH OUT

The sale does not end at the cheque, and certainly not at the start of a pilot. As an angel investor he sees companies reporting \$10 million in ARR that turns out to be six-month paid pilots that are not converting. He expects a mass extinction of this “pilot recurring revenue.” Onboard deliberately: walk users through the app, or put a person beside them. He picks up Satya Nadella’s remark that forward-deployed engineers are one of the fastest-growing startup roles, which he translates as boots on the ground making sure the product works for the customer.

His summary line: your product is not just the pixels on the screen. It is support, customer success, the founder, training, and everything around it. The best pixels lose to a company that invests more in

making sure customers actually use its product.

5.4 From the Q&A

Competitors. Do not care about them. These markets are trillions of dollars and no single company will win most of them. You will be scared of competitors at first, then astonished at how bad they are once you start building. To pick a market, look for work companies already outsource, especially abroad, since they have shown they will hand it off. Avoid work that is part of a company's identity: Pixar will not outsource the story of a Pixar film. Look for large markets, pain shared across many companies, and areas you know or can learn. Or print out a list of knowledge work, throw a dart, and run at wherever it lands.

What to focus on at each stage. What he should have done: at seed, build a great product that reaches product-market fit; at Series A, the same; at B and C, the same. What he actually did was focus on many things that mattered much less. Hiring, marketing, culture and fundraising are all necessary, but only as means to that product end. Founders read blog posts and listen to investors and start treating "great culture" or "hire a sales team" as ends in themselves. He fell into that trap.

Pricing the previously unthinkable. Start from the value to the business: if it saves \$100 million, or would have cost \$5 million, talk to the customer about taking 10–20% of that. For replacement services, start near what humans charge; competition will push prices down, which is bad for your business and good for society.

Is it just a GPT wrapper? Build it and you will see how hard it was: the many small pieces, data integrations, checks, carefully tuned prompts and model choices. After two years of doing nothing but that, you will have built something no one else can easily build. Do not be scared.

PARTING ADVICE

Asked by a founder who had already sold a startup at 14 what he would do tonight at Startup School:

He thinks focusing on legal was partly a mistake early on. Lawyers earn a lot, but they spent very little on software, so before LLMs Casetext could only make incremental changes for a limited market. After LLMs it changed far more lives, and he is now addicted to that feeling.

So focus on the biggest problem you can think of that is solvable with the technology and skills you have. People want not to do laundry; everyone would like a cleaner for eight hours a day and cannot afford one. Could a robot do it, the way the dishwasher freed people in the 1950s? Find a huge problem many people have that you believe you can solve, and run as hard as you can.

Fei-Fei Li: Problems Bordering on Delusional



“Fei-Fei Li: Spatial Intelligence is the Next Frontier in AI”

Fei-Fei Li Co-founder and CEO, World Labs; Stanford professor; creator of ImageNet. Interviewed by YC General Partner Diana Hu, followed by audience Q&A.

Video: https://www.youtube.com/watch?v=_PioN-CpOP0

She opens by saying she feels like one of the audience: she has just started a small company too. The conversation traces an arc through her career, from objects to scenes to worlds, and each step is a problem she expected to take a lifetime.

6.1 ImageNet: betting on data when there was none

About eighteen years ago she was a first-year assistant professor at Princeton. There was very little data, computer vision algorithms did not work, there was no AI industry, and to the public the word barely existed. A small group still shared the dream of machines that think. Her own version was to make machines *see*, because seeing is not just perceiving; it is understanding the world and acting in it.

One problem haunted her: generalization, which she calls the core mathematical goal of machine learning. Generalization needs data, and in computer vision nobody had any. Her generation of graduate students was the first to grow up with the internet.

AGAINST THE GRAIN

Around 2007 she and her student bet that machine learning needed a paradigm shift led by data-driven methods. They set out to download about a billion images from the internet and build a visual taxonomy of the entire world, for training and benchmarking algorithms. That became ImageNet.

From the 2009 publication (a small conference poster) to 2012 they believed data would drive AI but had little evidence. They did two things to change that:

- **Open-sourced it** to the whole research community from the start.
- **Created a challenge**, releasing a test set each year and inviting anyone to compete, so the world's best students and researchers would work on the problem.

The first years set baselines, with error rates around 30%. Late one night at the end of summer 2012, while running the challenge results, a student pinged her about one entry that stood out. It was a convolutional neural network from Geoffrey Hinton's team, entered under the name SuperVision. The algorithm dated from the 1980s with a few tweaks, and the jump was startling. It was presented at the workshop in Florence that year. She stresses what was new: it was the first time two GPUs were combined for deep learning, so it was the first moment data, GPUs and neural networks came together. The world now knows it as AlexNet.

6.2 From objects to scenes

ImageNet addressed calling out objects in an image: cat, chair. But since graduate school she had a dream she thought would take a hundred years: storytelling. When you open your eyes in a room, you do not see "person, chair, chair"; you see a conference room with a stage, a crowd and cameras.

She told herself as a new graduate that if she could create an algorithm that told the story of a scene before she died, she would have succeeded. Then deep learning took off, Andrej Karpathy and later Justin Johnson joined her lab, and around 2015 they published some of the first papers in which a computer captioned an image. Her reaction: what am I going to do with the rest of my life?

She joked to Karpathy that they should do the reverse, a sentence in and an image out. He laughed it off; the world was not ready. Now generative models do exactly that.

6.3 Why spatial intelligence is the next problem

When she looks for a next north-star problem, she often asks what evolution or brain development did. By her account, sophisticated language took evolution well under a million years, and humans are essentially the only animals with it. Vision took 540 million years, from the first trilobites seeing underwater. Before vision, for half a billion years, animals were simple. Seeing set off an evolutionary arms race in animal intelligence.

THE ONE THING

To her, AGI will not be complete without spatial intelligence: the ability to understand, generate, reason about and act in the 3D world. That requires world models that go beyond flat pixels and beyond language to capture the 3D structure of the world. That is the problem World Labs was founded to solve.

Her co-founders are Justin Johnson, Ben Mildenhall (author of the NeRF paper) and Christoph Lassner (creator of Pulsar, which the interviewer describes as a precursor to Gaussian splatting). She says she is lucky always to work with the best young people.

6.3.1 Why it is harder than language

Diana Hu suggests vision may be harder than language models. Li agrees and gives four reasons:

1. **Dimension.** Language is fundamentally one-dimensional: tokens arrive in sequence. The world is 3D, or 4D with time, which is combinatorially harder.
2. **Projection.** Every eye or camera collapses 3D to 2D, which makes recovering the world mathematically ill-posed. That is why animals have multiple senses.
3. **Generation versus reconstruction.** Language is purely generative: there is no language in nature; it comes out of people's heads. Worlds can be generated, but they must obey physics, and there is also a real world to reconstruct. You slide between the two. At the generation end are games and the metaverse; at the real-world end, robotics.
4. **Data.** The internet is full of language. Spatial data mostly lives in our heads and is not easily accessible.

AGAINST THE GRAIN

The difficulty is exactly what excites her. If a problem were easy, someone else would have solved it. Her whole career has been choosing problems so hard they border on delusional, and she thinks this is that problem.

On architecture, she thinks language models largely ride scaling laws, brute-forcing self-supervision. World models may need more nuance, because the world is more structured and may need guiding signals, whether priors or supervision in the data. These are open questions. We do not even fully understand 3D perception in humans, and humans are not especially good 3D animals.

She will not say much about World Labs' product, but lists uses for spatial intelligence: designers, architects, industrial designers, 3D artists and game developers on the creative side; robotic learning on the other; plus marketing, entertainment and the metaverse. She is excited about the metaverse precisely because it still does not work. She expects hardware and software to converge, and metaverse content creation needs world models.

6.4 A founder all along

Hu points out this is not her first zero-to-one. Li immigrated to the US as a teenager with no English. At 19, out of desperation to support her parents while studying physics at Princeton, she started a dry-cleaning shop. In Silicon Valley terms, she fundraised, was founder, CEO and cashier, and exited after seven years. The room applauds; she says she has never been applauded for a laundromat before.

TRY THIS

Her pattern of choosing the harder path: twice as a professor she joined departments where she was the first computer vision professor, against advice to go where there were senior mentors and a community. She would have liked mentors, but if they are not there, blaze your own trail. She later went to Google Cloud to learn business and B2B, then returned to Stanford around 2018 to start the Human-Centered AI Institute, which she ran like a startup for five years, because AI had become a human problem.

THE ONE THING

She loves standing at ground zero: forget what you have done in the past, forget what others think of you, hunker down and build. That is her comfort zone.

6.5 What her best students shared

She has advised Andrej Karpathy, Jim Fan and ImageNet co-author Jia Deng among others, and says she owes them more than they owe her. They became very different people: pure scientists, industry leaders, great communicators of AI knowledge.

THE ONE THING

What unifies them, and what she now hires for at World Labs, is *intellectual fearlessness*: the courage to embrace something hard and go all in on solving it, whatever your background and whatever the problem.

6.6 From the Q&A

What should a new PhD student work on? Academia no longer has most of AI's compute and data. So choose north-star problems not on a collision course with what industry can solve faster with more compute, data and large teams. She suggests: interdisciplinary AI, especially for scientific discovery; theory, since capability has completely outrun understanding (explainability, causality); unsolved representation problems in vision; and small-data learning.

Will AGI be one model or many agents? She struggles with the term AGI. The 1956 Dartmouth founders and Alan Turing were asking how to make machines that think, which was never a narrow goal. She cannot see how that founding question differs from what industry now calls AGI; today's systems beat the narrow AI of past decades, which is just progress. Without a definition she cannot say whether it is monolithic. The brain is one organ with specialized regions.

Who should go to graduate school? People with a burning curiosity strong enough that there is no better place to follow it. A startup cannot be led only by curiosity, because your investors will be annoyed. In a PhD you can be happy following curiosity even while the world rushes by.

Open versus closed models? An ecosystem with different approaches is healthy, and she is not religious about it. It depends on business strategy: Meta open-sources because it is growing an ecosystem, not selling models, and others may have open and closed tiers. But open source, in academia and industry, should be protected rather than penalized, because it matters for entrepreneurs and the public sector.

Where does World Labs get spatial data? "Join World Labs and I'll tell you." She will only say it is a hybrid approach, and that quality matters as much as quantity: garbage in, garbage out.

PARTING ADVICE

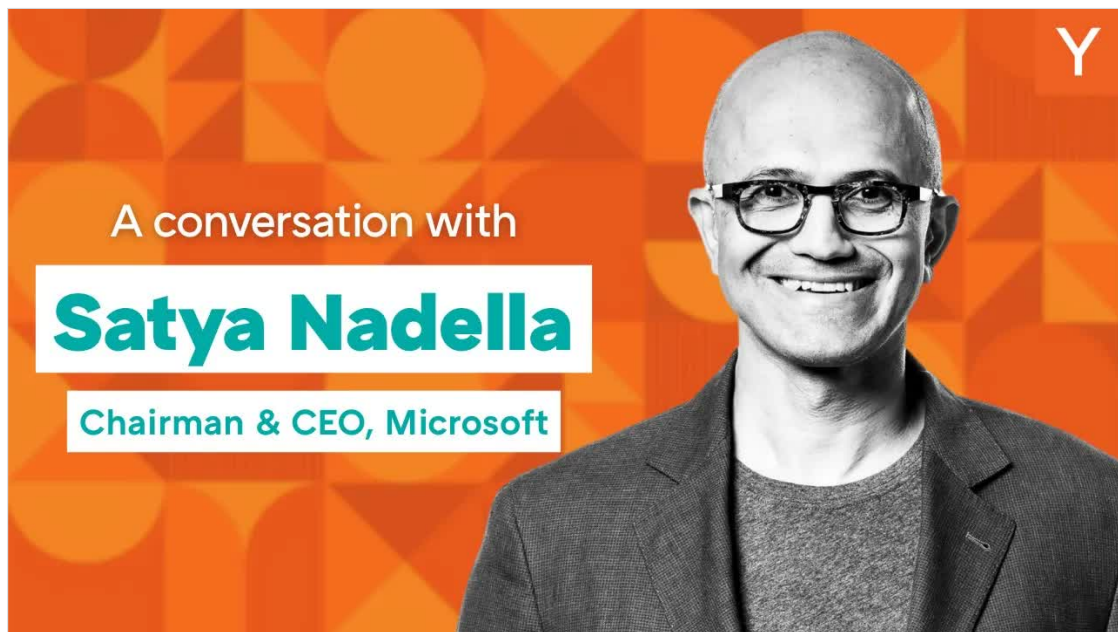
Asked whether she had felt like a minority in the workplace and how she handled it:

Everyone has moments of being the only one in the room, whether because of who they are, their ideas, or the colour of their shirt, and she has felt it. Coming to the US young, she learned not to overindex on it: she is an immigrant woman, and she is also there, like everyone else, to learn and to build.

And for anyone starting something: you will have moments of weakness and of feeling you do not know what you are doing. She feels it every day in startup life. Just focus on doing it. Gradient-descend yourself to the optimal solution.

CHAPTER 7

Satya Nadella: Earn the Social Permission



“Satya Nadella: Microsoft’s AI Bets, Hyperscaling, Quantum Computing Breakthroughs”

Satya Nadella Chairman and CEO, Microsoft. Interviewed on stage by YC’s Garry Tan.

Video: <https://www.youtube.com/watch?v=AUUZuzVHKdo>

The interviewer identifies himself in passing as an early Palantir employee who learned product management at Microsoft, which is how these notes identify him. Nadella’s recurring theme is that the real test of AI is not benchmarks but whether it measurably improves economies and people’s lives.

7.1 Platform, product, partner

Nadella thinks of Microsoft along three dimensions: platform, product and partner. In 35 years he has lived through client–server, the web and internet, mobile and cloud; AI is the fourth shift he pattern matches against.

THE ONE THING

AI is diffusing so fast and so widely because it compounds on the previous platform. Without the cloud there would have been no AI supercomputers, without those no models, and without models no products. You always build the next platform on the last one.

Each shift brings a new workload that relitigates the platform. A large training job, data-parallel and synchronous, is nothing like a Hadoop job, so the whole stack has to change. He calls this a golden age of systems software, with opportunity for startups at the infrastructure layer as well as for hyperscalers, and of course in models and products.

His benchmark for all of it is simple: is AI creating surplus, one community, country, industry and company at a time? What is it for, ultimately? Economic growth.

7.1.1 The model as SQL

Where does the model end and the product begin? If a product is just a model with scaffolding and tool calls in a loop, that looks confusing, but it is like saying an app is just SQL plus business logic.

AGAINST THE GRAIN

He always hoped machine learning would have a “SQL moment.” Earlier AI was built vertically, with no stable platform layer. For the first time, the model layer is something like a SQL engine you can build sophisticated products on, and inference-time compute plus tool calling gives a fairly robust harness. So there is plenty of room to build an app tier on top.

He adds where he thinks product creation will really happen: in the feedback loop, the data path inside the product that is used to post-train the model and improve tool selection.

7.2 Energy needs social permission

On compute demand, he reasons roughly: intelligence scales with the log of compute; computing uses perhaps 2–3% of US energy today; even doubling that to 6% would require a massive amount of extra energy.

WATCH OUT

If history teaches one lesson, it is that if you are going to use energy you had better have social permission. That means AI’s output has to be socially useful. If it does not create economic surplus that shows up for countries and communities, the industry will not be allowed to consume the energy. Everyone is focused on energy production; the real challenge over the next five years is producing enough valuable products to show up unequivocally in real statistics, not just AI benchmarks.

He is confident it can, pointing at healthcare, education and productivity. US healthcare is 18–19% of GDP, and while everyone talks about miracle drugs, much of the cost is workflow. Take something as simple as patient discharge: the back end of an electronic medical records system, a model and a prompt would save enough time and money to pay for itself, and every clerical dollar could have gone to treatment, or a physician’s time back to a patient.

7.3 The rate limiter is change management

THE ONE THING

His analogy for a young audience: a multinational once built sales forecasts from faxes, then inter-office memos, annotated and consolidated, hopefully before quarter end. Then email, PCs and Excel arrived, and the forecast became a spreadsheet people filled in. The work artifact and the workflow changed. That is what has to happen with AI. If you do your job with 99 agents, the workflow cannot stay the same, and even the scope of your job changes.

His example: LinkedIn combined design, front-end engineering and product management into a single “full-stack builder” role. Rebuilding teams with new roles and scopes is the social rate limiter, more than power or other deployment issues.

This is also why he thinks forward-deployed engineers, the Palantir model, have become common in AI startups. Customers need help understanding not just the technology but how to use it in a workflow.

The interviewer describes YC’s advice to go undercover, for example as a medical biller, and see how much knowledge work is copying from a browser into a spreadsheet into an email. Nadella agrees this is one of the most understated opportunities: the drudgery in knowledge work. In software engineering, the joy was taken out by constantly leaving flow to gather information. Very little time is spent in actual synthesis. Pairing a reasoning model with your own thinking, while agents handle the mundane, is the frontier for all knowledge work.

7.4 What is changing technically

He had not imagined a year earlier how far reinforcement learning and test-time compute would go. Pre-training worked, post-training on top worked, and inference-time compute added another scaling law.

AGAINST THE GRAIN

Stay open to the idea that the last big algorithmic breakthrough has not been found. One person in the room could come up with a more efficient approach and change the entire regime. He also expects the next step to be an end-to-end loop from pre-training through RL, which could itself be another scaling breakthrough.

Asked whether an LLM instance that disappears when you close the chat is a kind of consciousness that needs its loop completed, he declines the premise.

AGAINST THE GRAIN

“Artificial intelligence” is, he says, the worst name we could have picked. He is not into anthropomorphizing it. It is a tool. It shows signs of intelligence, but not the intelligence he has, and human agency will still matter.

What he does see as the next frontier, as first-class systems built around the model:

1. **Memory.**
2. **Tool use.**
3. **Entitlements**, which he calls perhaps the most important: if an agent takes an action, what is it entitled to do? Put those together and an agent has an identity, management and provisioning controls.

7.5 Software, SaaS and software engineers

YC and some venture investors in the room are worrying whether B2B SaaS survives if users generate software just in time. Nadella’s answer is that both will exist. He notes how many people fork VS Code and takes it as a sign Microsoft did something right: a great canvas still matters. He thinks of Excel as an IDE, and imagines bringing the best analyst model into that canvas with a loop between the two.

On software engineering, he offers a thought experiment. A Martian visiting an office in the 1980s would see typing pools and slide pools. Returning today, it would conclude all eight billion people are typists. Similarly, everyone will create software, and there will still be a job called software engineer, but it will be different: closer to software architect.

WATCH OUT

Vibe coding is fantastic until it does something and you do not know what happened. You still need a mental model of your repository. His favourite GitHub feature is now the full change log of every agent working on his repo. Much of software engineering will look like a good dev manager's job used to: make sure builds do not break and code quality holds.

And an underdiscussed constraint: until laws change, legal liability sits with humans and the institutions they build. So humans must stay in the loop at a fundamental level, which means building many tools that let humans see what agents are doing.

7.6 Underhyped and overhyped

There is no shortage of hype, and he thinks that is partly healthy: the industry lives on frenzy about the new. He quotes the line about being either busy being born or busy dying.

THE ONE THING

Overhyped: model capability, which is fantastic. Underhyped: the real difference AI makes in people's lives everywhere. In early 2023 in India he saw a local developer chain an early GPT model with open-source speech-to-text and text-to-speech to build a WhatsApp chatbot that let a farmer apply for an agricultural subsidy on a government website. Something built on the US West Coast reached a real use that fast. If the world recognizes stories like that, the industry is in good shape. If it is all about valuations, it will not end well.

He adds a World Bank study, started in Nigeria and extended to Latin America, which found access to a Copilot-like tool was probably the best technology intervention in education there, the dream Microsoft has chased for decades.

On Copilot in Windows he jokes about Clippy returning, then describes what feels surreal to him: it is like a new browser moment. Microsoft Research's first group, in 1995, worked on speech, and speech has never been first-class on PCs. Now he leaves Copilot on with vision and voice: it sees what he sees and he talks to it, which he likens to a new kind of precise mouse movement. Even familiar form factors can change, and new ones will come.

Asked whether *Her* was right that the operating system becomes your most trusted agent, he says that is the direction: agents become your computer and do the computer use for you. The operative word is trust, which means precision and privacy. He frames it as three layers of concern any product must answer: *privacy* for every user, *security* for every customer organization, and *sovereignty* for every country.

7.7 Lessons from engineer to CEO**TRY THIS**

Treat your current job as the best one you will have. When he joined Microsoft in 1992 he felt that if he retired in that first job, it would be fantastic. He was not waiting to become CEO, or for the next promotion, to do his best work. Take what you have as the biggest thing and make it expansive.

TRY THIS

Learn to make teams great. The big difference between school and work is that you join a team. Everyone assumes it is someone else's job to align the team. It is your job.

7.7.1 What he looks for in people

1. **People who bring clarity.** From Bill Gates: good architects bring clarity and bad architects bring confusion, even when equally smart. Someone who can drop into an ambiguous situation and make clear what to do next is at a premium.
2. **People who create energy,** and bring constituents together inside and outside the company. A leader who says “my team is great and everyone else sucks” is not useful.
3. **People who solve over-constrained problems.** His favourite interview question asks about a project that was going nowhere and how the person found a path. Success is taking an over-constrained problem and finding a way to unconstrain it.

Leadership, he adds, is not something you do later in life; you do it at every step.

7.8 Quantum

He is the third Microsoft CEO writing cheques for quantum, over more than twenty years. The bet was that a general-purpose, fault-tolerant quantum computer needs very stable, error-corrected qubits, based on a physical property proposed by the physicist Majorana. Microsoft says it has now made that physics breakthrough and fabricated the particle, leading to the Majorana 1 chip. His framing: nature is quantum, so simulation is best done on a quantum computer, while AI acts as an emulator of that simulator. Today, AI plus high-performance computing is already accelerating chemistry, physics and materials science, and quantum would be the next step in that loop.

PARTING ADVICE

Asked what he would work on if he were 22 and starting at Microsoft in 2025:

Tools. The story of how Office was built still amazes him: a word processor, a spreadsheet, a slide tool, and what they came to mean to everyone. His favourite products are VS Code and Excel, because of the sense of empowerment you feel using them. A spreadsheet, rows and columns with a Turing machine in the middle, is an unbelievable scaffold for analytical power.

He sees today’s researcher, analyst and creator agents as the new Word, Excel and PowerPoint. What are the next tools we can put in people’s hands that give them that sense of empowerment? That is what he would love to work on.

François Chollet: Skill Is Not Intelligence



“François Chollet: How We Get To AGI”

François Chollet Creator of Keras and the ARC-AGI benchmark; co-founder of the research lab Ndea. A solo talk with slides, no Q&A.

Video: <https://www.youtube.com/watch?v=5QcCeSsNRks>

The captions garble the name of his new lab; it is Ndea. He opens with what he calls one of the most important facts about the world: the cost of compute has fallen by about two orders of magnitude per decade since 1940, with no sign of stopping. Compute and data were long AI’s main bottlenecks, and in the 2010s, with GPUs and large datasets, deep learning started to work.

8.1 Why scaling pre-training did not get us there

Self-supervised text modelling worked, and scaling LLM pre-training became the dominant paradigm. It crushed benchmarks, and got predictably better with model and data size using the same architecture and training process. Many people extrapolated that scale was all that was needed, and the field became obsessed with the idea that general intelligence would emerge from cramming more data into bigger models.

WATCH OUT

We were confused about what those benchmarks meant. There is a big difference between *memorized skills*, static and task-specific, and *fluid general intelligence*, the ability to understand something you have never seen before, on the fly.

In 2019 he released the Abstraction and Reasoning Corpus, ARC-1, to highlight that difference. From then to models like GPT-4.5, base models were scaled up roughly 50,000 times, and ARC-1 accuracy went from 0% to about 10%. Anyone in the audience would score above 95%. Cracking fluid intelligence needed new ideas.

Then in 2024 the research community pivoted to *test-time adaptation*: models that change their own state at inference time to adapt to something new, through techniques like test-time training, program synthesis, and chain-of-thought synthesis where the model effectively reprograms itself for the task. Progress on ARC finally appeared. In December 2024 OpenAI previewed o3, in a version fine-tuned on ARC, reaching human-level performance on ARC-1 for the first time. Every approach that does well on ARC now uses one of these techniques.

He then sets out to answer three questions: why pre-training scaling failed, which two years ago was dogma and today almost nobody believes; whether test-time adaptation gets us to AGI; and what comes next.

8.2 What intelligence is

There are two traditional definitions of AI's goal:

- **Minsky's view:** machines that perform tasks normally done by humans. This matches the mainstream corporate definition of AGI as a model that can do most economically valuable tasks (80% is a figure often quoted).
- **McCarthy's view:** machines that handle problems they were not prepared for.

His view is closer to McCarthy's.

THE ONE THING

Intelligence is a process; skill is the output of that process. Showing skill at any number of tasks does not show intelligence. A road network lets you travel between a fixed set of places; a road-building company can connect new places as needs change. Crediting a crystallized skill program with intelligence is a category error: it confuses the road with the process that built it.

More formally, intelligence is a conversion ratio: how efficiently you turn the information you have (past experience, plus any built-in priors) into the ability to operate across the space of future situations, which will be novel and uncertain.

WATCH OUT

That is why exam-style benchmarks are a poor measure of progress towards AGI. Human exams measure task-specific skill and knowledge, and make assumptions that are sensible for people but not for machines. Most exams assume you have not read and memorized all the questions and answers beforehand.

He names three concepts any rigorous measure must account for, each a spectrum rather than a binary:

1. **Static skills versus fluid intelligence:** a library of programs for known problems versus synthesizing new programs for unseen ones.

2. **Operational area:** skilled only close to what you have seen, or across a broad scope. If you can drive, you should be able to drive in Sacramento after learning in San Jose, not only inside a geofence.
3. **Information efficiency:** how much data or practice a skill required. More efficient means more intelligent.

8.2.1 You hit the target and miss the point

These definitions matter because engineers can only build what they measure. The measure determines the questions we ask and the answers we get.

AGAINST THE GRAIN

The “shortcut rule”: optimize a single measure of success and you may succeed, at the expense of everything the measure did not capture. The Netflix Prize winner was extremely accurate and too complex to use, so it never shipped. Chess AI was pursued in the 1970s to learn about human intelligence; Deep Blue beat Kasparov decades later and taught us almost nothing about it. For decades AI chased task-specific skill because that was the definition of intelligence, and that definition only leads to *automation*, which is what we have. We actually want AI capable of autonomous *invention*, that can accelerate science. For that, target fluid intelligence itself.

8.3 The ARC benchmarks

8.3.1 ARC-1

ARC-1 is like an IQ test for machines and humans: 1,000 tasks, each unique, so you cannot cram. Every task needs some knowledge, but where most benchmarks leave required knowledge implicit, ARC makes it explicit. Tasks are built only on *core knowledge priors* a four-year-old has: objectness, elementary physics, basic geometry and topology, counting.

THE ONE THING

When a child does well on a problem and the most sophisticated AI struggles, that is a flashing red light that new ideas are needed. ARC is not meant to tell you whether a system is AGI. It is an arrow pointing research towards the most important unsolved bottleneck.

ARC resisted pre-training scaling entirely, which he takes as decisive evidence that fluid intelligence does not emerge from scaling pre-training. When test-time adaptation arrived, ARC was the only benchmark giving a clear signal, because others were saturated and could not tell a real rise in fluid intelligence from brute-force scale.

But ARC-1 is effectively binary. With no fluid intelligence you score near zero; with any, you score very high, as o3 did. It saturates well below human level.

8.3.2 ARC-2

Released in March 2025, ARC-2 targets reasoning systems and test-time adaptation. The format is the same, with much more focus on *compositional generalization*. ARC-1 tasks could often be solved at a glance; every ARC-2 task needs deliberate thought, and it is not easily brute-forced.

To check it was still fair, they tested 400 people in person in San Diego: not physics PhDs but Uber drivers, UCSD students, unemployed people wanting some cash. Every task was solved by at least two of the roughly seven people who saw it, so ten random people voting would score 100%.

Results: base models like GPT-4.5 and Llama 4 score 0%. Static reasoning systems that generate a single chain of thought score 1–2%. Only systems using test-time adaptation score meaningfully above zero, and they remain far below humans.

THE ONE THING

As long as it is easy to come up with tasks any ordinary person can do that AI cannot, no matter how much compute you throw at it, we do not have AGI. You will know we are close when such tasks become hard to find. We are clearly not there.

8.3.3 ARC-3

ARC-3 departs from input–output pairs to test *agency*: exploring, learning interactively, setting and achieving goals. An AI is dropped into a brand-new game where it does not know the controls, the goal or the mechanics. Every game is unique and built only on core knowledge priors. Efficiency is central: there is a strict action limit, targeting human-level action efficiency. A developer preview was planned for July 2025 and the launch for early 2026.

8.4 What is still missing

If the future had nothing in common with the past, no intelligence could make sense of it. But nothing is ever truly novel.

THE ONE THING

The *kaleidoscope hypothesis*: experience seems endlessly novel and complex, but the number of unique “atoms of meaning” needed to describe it is small, and everything is a recombination of them. One tree resembles another and also a neuron; electromagnetism resembles hydrodynamics and gravity. Intelligence is the ability to mine experience for these reusable atoms, called abstractions, and recombine them on the fly into a model fitted to a new situation.

So implementing intelligence has two parts: **abstraction acquisition**, efficiently extracting reusable abstractions from experience; and **on-the-fly recombination**, efficiently selecting and combining them. Efficiency is the point. Needing hundreds of thousands of hours for a simple skill, or enumerating every chess move to find the best one, is not intelligent. Both data efficiency and compute efficiency count.

That explains why scaling did not produce AGI. Models learned many abstractions in training but were completely static at test time, able only to fetch and apply a pre-recorded template. Test-time adaptation fixes that, which is a huge step. But two problems remain:

- **Inefficiency.** Gradient descent needs roughly three to four orders of magnitude more data than humans to distil simple abstractions. Even state-of-the-art test-time methods need thousands of dollars of compute to reach human level on ARC-1, and that does not scale to ARC-2.
- **No compositional generalization**, which is what ARC-2 measures.

8.4.1 Two kinds of abstraction

Both kinds compare instances and merge them into templates by dropping irrelevant details. They differ in domain:

- **Type 1, value-centric:** compares things by a continuous distance. It underlies perception, pattern recognition, intuition, and modern machine learning.
- **Type 2, program-centric:** compares discrete programs, meaning graphs, by looking for exact structural matches and subgraph isomorphisms. It underlies much human reasoning, and it is what a software engineer means by abstraction when refactoring.

All cognition, he argues, combines both, a bit like the left-brain, right-brain metaphor. Transformers are excellent at Type 1 and a major breakthrough for it, but a poor fit for Type 2, which is why it is hard to train them to do something as simple as sorting a list or adding digits given as tokens.

AGAINST THE GRAIN

Type 2 needs discrete program search, not just manipulation of continuous spaces learned by gradient descent. Deep learning does not invent; search does. Every AI system he knows of that shows invention or creativity relies on discrete search: evolved antenna designs in the 1990s, AlphaGo's move 37, and DeepMind's AlphaEvolve.

He contrasts the two approaches directly:

	Machine learning	Program synthesis
Model	differentiable parametric curve	discrete graph of symbolic operators
Learning engine	gradient descent (compute-efficient)	combinatorial search (compute-hungry)
Data needs	dense sampling of the data manifold	two or three examples
Wall you hit	data density	combinatorial explosion

8.4.2 Combine them the way people do

Going all in on either type will not go far. Humans are special because they combine perception and intuition with explicit step-by-step reasoning. In chess, you calculate a few candidate lines step by step (Type 2), but you only calculate a couple, and you choose which by pattern recognition built up from experience (Type 1). Intuition makes calculation tractable.

THE ONE THING

The merger he proposes: use Type 1's fast, approximate judgments to fight Type 2's combinatorial explosion. The analogy is drawing a map. Embed discrete objects with discrete relationships (like stations on a subway network) into a continuous space, where distances give quick, approximate guesses about the relationships, keeping search under control.

8.5 The system Ndea is building

He expects AI to move towards systems that behave more like programmers, writing software for each new task. His picture:

- A programmer-like meta-learner that, faced with a new task, synthesizes a program or model fitted to it.
- That program blends deep learning modules for Type 1 sub-problems such as perception with algorithmic modules for Type 2 sub-problems.
- It is assembled by discrete program search, guided by deep-learning intuition about the structure of program space.
- Search draws on a global, constantly evolving library of reusable abstractions. New building blocks discovered while solving a problem are added back, the way an engineer publishes a useful library to GitHub.
- The system improves over time by growing the library and refining its intuition about program space.

Ndea's reason for existing is that dramatically accelerating science needs AI capable of independent invention and discovery, expanding the frontier of knowledge rather than operating inside it. Deep learning is incredibly powerful for automation, but discovery requires something more.

PARTING ADVICE

The talk ends on his lab's plan rather than advice, so this is the closest thing he said.

Ndea's first milestone is to solve ARC-AGI with a system that starts out knowing nothing at all

about ARC-AGI, and then to use the same approach to empower human researchers and accelerate the timeline of science.

The underlying lesson for builders is the one he spends the talk on: be careful what you measure. If your target is skill at known tasks, you will get automation. If you want invention, measure the ability to adapt to what is genuinely new, and measure how efficiently it is done.

CHAPTER 9

Aravind Srinivas: The Only Moat Is Speed



“Aravind Srinivas: Perplexity’s Race to Build Agentic Search”

Aravind Srinivas Co-founder and CEO, Perplexity. Interviewed by a YC partner, followed by audience Q&A. The interviewer is not named in the captions.

Video: <https://www.youtube.com/watch?v=2jOnoTEk-xA>

Asked for an update, Srinivas says usage is growing faster than they know how to handle: he has infrastructure problems every day, and they are rebuilding to scale the next tenfold.

9.1 The browser bet

He is candid about the competitive position. ChatGPT has more distribution. Every AI app is adding search, and all of them support citations. Perplexity intends to stay best at that, but its big bet for the future is a browser, Comet.

The idea is one box where you can navigate, ask questions and give agentic tasks, with an assistant on your new tab page and in a sidebar on any web page. He describes it as a *cognitive operating system* rather than another browser, and more assistant than fully autonomous agent. You would launch several tasks in parallel, running asynchronously, drawing on your email, calendar, shopping and social accounts: research real estate, watch markets. Chrome was exciting because each tab was its own process. In Comet, each prompt could be.

9.1.1 “What if everyone builds this?”

The interviewer points out that OpenAI and Google are probably heading the same way, with far more money.

THE ONE THING

If something is really worth doing, it is natural for well-funded companies to do it. Perplexity showed answers with sources were valuable, and now everyone does them. Cursor was a great product, and now the large labs have their own coding tools. You can only be world-class at a limited number of things. He assumes OpenAI and Anthropic will build browsers and Google already has one. The only moat you have is speed: innovate and move faster than everyone. It is a marathon run at sprint pace.

He argues a browser is much harder to copy than another chat tool, and a mobile version harder still (many months of engineering).

The interviewer adds a detail from backstage: while demoing new features, Srinivas hit a bug and stopped everything to work out what was wrong, where a big-company CEO would hand it off. He says he loves triaging bugs, even if people question whether it is the best use of a CEO's time, and is pleased to see even Sundar Pichai doing bug support on X.

9.2 How Perplexity started

AGAINST THE GRAIN

They started the company without a clear idea of what to build, the opposite of YC's usual advice to turn a project into a company. With AI improving this fast, he thinks you need not stick rigidly to one idea at the start. But do not change the idea every week either. Brainstorm, then build immediately and put it in people's hands.

The first tool was natural-language-to-SQL, framed as search over relational databases. He loved Twitter and there was no good way to search it (there still is not), and he had admired the original Facebook Graph Search. So they organized Twitter's data into relational tables and converted each user query into SQL. It worked very well. Later they decided to go across the whole web, which cannot be turned into tables, and bet that language models could handle reasoning and structure over unstructured content. That became Perplexity.

9.2.1 Why AI, and who to start with

Why machine learning? Because it was the only thing he was good at. He could not start a delivery company or a social network; he was not the right fit.

TRY THIS

Have the intellectual humility to know what you are good at and what is actually doable with the resources you have. People mocked Perplexity as an AI company that does not train foundation models, though it trains many other models.

His co-founders were people he knew from grad school and had discussed ideas with for years. You do not talk to people with a calculation about future co-founders; you talk to them because they are interesting. He sees the YC network the same way: even if your first startup fails, you meet people who may be future co-founders.

9.2.2 Signs it was working

Products often have an initial wow factor. Usage then either drops to nothing, meaning no real retention, or drops and levels off into sustained use. Their database search over Twitter, LinkedIn and GitHub showed sustained use, so they knew there was something to combining language models with search.

They then dreamed bigger: just give answers and cite sources. They launched that as a Discord bot, saw continued use, and launched publicly seven days after ChatGPT, when ChatGPT had no web search.

THE ONE THING

His aha moment was New Year's Eve, with close to 700,000 queries. The product had, in his words, the worst name for a consumer product, hard to share. It took seven seconds to answer, made mistakes and hallucinated, from an unknown company and founder with one or two million dollars of seed money. People were still sharing screenshots on New Year's Eve, when they could have been watching Netflix. That was when he knew it was real and committed to the vision.

9.3 Why a startup could take on Google

He first seriously thought about competing with Google when Sundar Pichai published the Bard blog post, while Perplexity was raising its Series A and everyone said Bard would do the same thing.

His reasoning for why Google could not simply do it in its main product:

- **Incentives.** If users get a direct answer to “best hotel in San Francisco with a Golden Gate view” with a booking link, how does Google keep charging Booking, Expedia and Kayak, or Amazon and Walmart bidding against each other for the same ad? It is not in Google's interest to give the best answer.
- **Models.** Through 2023 and much of 2024, Google often had only the fourth- or fifth-best model. For the first time a startup outside Google had access to better AI than Google had internally, thanks to OpenAI, Anthropic and open-source models.
- **Room for error.** Perplexity could make mistakes. When Bard failed in a live demo, Google's stock fell 6%.

He says, knowing it sounds cocky, that Google has launched essentially the same feature year after year under different names and teams, never to everyone. He does not think Google lacks talent; it is a much larger business, no longer founder-led, where it is hard to take a hit on your own stock to do the long-term right thing. He is glad the dilemma exists, because otherwise there would be no opening for startups and monopolies would keep growing. For the first time people compare alternatives for finding information, and phone makers will offer several assistants rather than one locked-in default.

He reads every Twitter comment after Google I/O. Last year it was AI Overviews and “Perplexity is dead”; this year AI Mode and “Perplexity is dead.” He enjoys it. The real competitor he takes seriously is ChatGPT: well funded, without an innovator's dilemma, and the most successful consumer AI product. That is why he is pushing the company towards the browser, which he sees as an abstraction layer above chatbots, able to use your other tabs and history, fill in forms, pay bills, shop, and run recurring research.

Asked for a concrete reason to switch browsers, he admits the answer, a blend of AI, navigation and agents, sounds boring, but says no one has done it. Examples: scheduling meetings, replying to emails you do not want to read, or taking an event's applicant list, scraping LinkedIn profiles and filtering for a criterion. (He jokes that his sample filter, Stanford dropouts, would not have let him in.)

9.4 Running a 200-person company

TRY THIS

Perplexity made it mandatory to use at least one AI coding tool, mostly Cursor plus GitHub Copilot. ML researchers upload a screenshot of pseudocode from a paper and have the tool implement it, write unit tests and run an experiment, cutting experiment time from three or four days to about an hour. For design fixes he screenshots the iOS app, draws an arrow saying where a button should move, and an engineer feeds that to Cursor to change the SwiftUI file.

WATCH OUT

You do not want to vibe code everything. When production infrastructure breaks, you want people trained in software engineering and distributed systems fixing it. AI tools also introduce bugs that people do not know how to fix, or even how they got in. He finds newer tools like Claude Code much smarter, and is sure this is the future, but there are real problems now.

9.5 What is defensible

Brand. OpenAI has its own Cursor-like and Perplexity-like features, and neither company died. Once you have several million users, including paying ones, you do not die fast; you earn the right to survive and keep building.

Narrative. You have to tell people why you exist. For Perplexity it is accuracy, speed (still fastest to first token despite doing search), and how answers are presented. The things you obsess over become your brand. With tens of millions of users, not necessarily a hundred million, you can keep playing regardless of what others ship. Until then, you have to worry, and even now they worry, and the only answer is to move fast and ship.

Network effects. No AI product has in-app network effects yet. WhatsApp keeps users despite Meta's questionable brand because contacts and groups live there; AI chat histories can be exported and uploaded elsewhere. He thinks the browser helps: history, passwords, wallet, an agent that knows you, and recurring tasks people rely on and share with others.

Partnerships. Perplexity already works with Selfbook for hotel bookings, Tripadvisor for reviews, Yelp, Shopify and merchants selling directly, Firmly for native purchases, and data providers for finance and sports. Some companies will become MCP servers; some will not. A browser agent can work either way, using a site as a human would. He sees that as the advantage of the browser over committing wholly to MCP, which depends on third parties building reliable servers.

9.6 Business model

He does not expect to reach Google's profits, and does not think that is needed; no one, including Google, has ever had another business with those margins. His thinking:

- **Subscriptions** have gone further than expected, and he thinks they can grow to at least a few billion dollars a year.
- **Usage-based pricing** for agent tasks, perhaps benchmarked against the cost of hiring a person. Likely more volume than subscriptions but lower margins, since every use costs compute, whereas an unused subscription is pure margin.
- **Transaction fees** as more buying happens through AI. Cost-per-acquisition has historically had lower margins than cost-per-click, which is why Google never became a transaction platform.

He honestly does not know how it will play out. The interviewer suggests Google's ads model may be the best business model ever; perhaps so good it took AI to threaten it.

9.7 From the Q&A

Hallucinations. They build internal benchmarks. The only way forward is a better search index, better snippets from pages, and models fast enough to reason in multiple steps on every query without too much cost.

If you ran Google? He does not envy that job. Google has data on user behaviour he lacks, and many people dislike AI, so forcing it onto massive distribution is not easy. He does not know. He adds that if ads appear in every AI answer, you will hate it too, so it is good alternatives exist.

When model companies absorb your feature. Pick something you want to be known for, but also build new products that do not exist yet. The browser is the natural graduation from search, as Chrome was for Google, which had no browser at its IPO and far fewer queries. He is convinced agents can only really be built with a browser: mobile operating systems restrict calling third-party apps, and nobody wants to be disintermediated by an AI through MCP that quickly.

The web's future. The web is already long-tail with a power law, and it will skew further. Known brands will keep direct visits; sites gaming SEO for traffic will struggle.

Truth and bias. Some queries have objective answers, like a game score or the weather, but even those rest on trust in a data provider, built by being reliably accurate. For questions without one answer, the best approach is to present multiple perspectives without taking a side. They do not yet measure this well; evals for subjective questions are hard, since the right answer may be exactly what Wikipedia omits, and raters good enough to judge are scarce.

Choosing audiences. Partner to reach people outside your usual channels; plenty of people are not on Twitter or LinkedIn, and some traditional businesses reach them. But grow through agencies, keeping some overlap with existing users who can carry word of mouth into new circles.

PARTING ADVICE

Two answers, one to the interviewer and one to the audience.

What he would tell the room. Work incredibly hard; there is no substitute. Do not think you can out-strategize the big labs. If your company can make hundreds of millions or billions in revenue, assume a model company will copy it, because they have raised enormous sums and must keep finding revenue. Live with that fear, embrace it, even sleep with it, and wake up excited about what you will build. Your moat is moving fast and building an identity users care about.

What kept him going before it worked. He watched Elon Musk videos on YouTube, and says he is serious: specifically the one where, after a third failure in a row, Musk says he will never give up unless dead or incapacitated. Great entrepreneurs have kept going with the odds against them. What do you have to lose? Just keep going.

Amjad Masad: Application Software Goes to Zero



“The Future of Software Creation with Replit CEO Amjad Masad”

Amjad Masad Co-founder and CEO, Replit. A solo talk with slides, followed by audience Q&A.

Video: <https://www.youtube.com/watch?v=IWmDiDGsLK4>

Masad was asked to talk about the future of software. Much of the talk is how Replit sees it, plus a set of predictions he openly describes as thinking out loud.

10.1 From experts to everyone

His mental model comes from computing history. Mainframes needed experts. Early PCs were toys (people made fun of Apple for MacPaint) until the spreadsheet made them useful, and now PCs, x86 machines, run the world economy, even inside data centres. Something used by a trained few became something everyone uses.

Software engineering as a career traces back to the 1970s, Unix and C: four to six years of college and two or three more years on the job to be good at it. He thinks software is now making the same move from experts to everyone.

Replit’s mission for nine years has been to “solve programming” so anyone can write software. It built an IDE, language runtimes, an online sandbox, deployments and cloud services. When AI arrived,

they concluded the ultimate expression of the mission is that you do not have to code at all, because code is the bottleneck to more people making software. In late 2023 and early 2024 they put all their resources into agents.

TRY THIS

At the time, agents barely worked, but benchmarks like SWE-bench (real GitHub issues with their tests and final pull requests) showed a clear trend: barely working in 2022, starting to work in 2023, and by the talk around 70–80%. His advice to anyone building agents: really believe it is coming. He tells his team to be okay building a crappy product today, because two months later the models improve and the product suddenly becomes viable.

10.2 The habitat matters more than the agent

THE ONE THING

An agent that writes code is the easy part. The hard part is the infrastructure it lives in, which he calls its *habitat*.

What that habitat needs:

- **A sandboxed virtual machine in the cloud**, not your laptop, since agents can do scary things.
- **Scale** to millions of users.
- **An open environment** like the one engineers use: shell access, file reading and writing, and the ability to install system and language packages in any language. Agents are trained on standard Linux environments, and many agent sandboxes are too constrained.
- **Everything else a real engineer uses**: deployments, databases, and so on.

He refers to Karpathy's point earlier that coding is easy and the surrounding work is not, and says Replit has already solved many of those pieces. Agents are poor at building authentication, so Replit's agent simply turns on a built-in auth service with user management stored in a database. There are also deployments and custom domains, secrets management for API keys, background jobs, and storage.

On the roadmap:

- **Universal model access**, so an app can use any image or video model without the builder choosing a provider and getting an API key; Replit handles billing and integration.
- **Payments**, both for your users paying you and for your agent to have a wallet. If it decides it needs a Twilio integration that is not built in, it should be able to provision it with a card.
- **Hiring humans**. If the agent hits a CAPTCHA, it could post a TaskRabbit job for a person to solve it.
- **Hiring other agents**, such as the many accounting and sales agents YC startups are building. He argues MCP is really a traditional RPC protocol and does not solve agent-to-agent collaboration.

10.2.1 Levels of autonomy

He borrows self-driving levels:

1. A language server (IntelliSense): lane assist.
2. AI code completion (Copilot).
3. Replit Agent at launch.
- 3.5. Agent v2, which works 10–15 minutes alone but needs you to test the app periodically.
4. Agent v3, in progress: mostly autonomous, still needing some attention.

5. Expected within a couple of years: spin up a thousand agents on a thousand problems and be confident about 95% succeed, so anyone can direct hundreds or thousands of engineers.

10.2.2 Agent v3's three pillars

1. **End-to-end testing with computer use.** Models clicking around a computer are slow, expensive and not very good. But build at the edge of what is possible: he expects computer use to get much better in three to six months, open new markets and start automating real jobs. Once the agent can QA its own app instead of asking you, it can work 30 minutes to two hours.
2. **Sampling and simulation.** Reasoning models improve by producing more tokens, but much of it is thinking in place without feedback from the environment. Replit built a fully transactional, reversible file system where every edit is an atomic snapshot, allowing cheap copy-on-write forks. When facing a big change, the agent can fork itself and the environment many times, try different approaches in parallel, pick the best and merge it back, much as a person simulates branches when thinking. He expects this to improve reliability two- to threefold.
3. **Generated tests for every feature.** Replit's agent, and also Claude Code, Cursor and others, often build a feature and later break it. The goal is to write tests for each feature and run them on every change. It is harder than it sounds: models are fairly bad at writing unit tests, and the tests must be fast.

10.3 Predictions

AGAINST THE GRAIN

His prediction is that all *application* software goes to zero: it becomes dirt cheap, and nobody makes money on traditional SaaS. Not tomorrow or next year, and he has stopped predicting timelines, but on the order of years. If anyone can generate software of any complexity from a prompt, application value falls to almost nothing. He estimates perhaps 15% of the SaaS a business buys is replaceable today, heading to 100%.

His evidence is a colleague, Kelsey, who works in HR and had never written code. She wanted org chart software connected to ADP payroll with some specific features, found nothing that fit, and the options cost tens of thousands of dollars a year. She built one in three days. Replit uses it, and he says it could be sold as a SaaS product.

10.3.1 Generalists, not specialists

Since the industrial revolution, the economy has trended towards specialization: one person makes a part, another tests, another assembles, each as replaceable as possible. When an HR professional can also be a software engineer or marketer with agents, jobs become less siloed.

THE ONE THING

Replit is building its first real product team this way: designer, engineer and product manager, usually in the same person. He expects organizations to look more like networks, or open-source projects, than hierarchies of departments. Each person's mandate each morning would not be "write this email" or "optimize this button" but "make the business work." Everyone becomes something like an entrepreneur. Domain expertise still matters, but much less than it did.

He reads from *The Sovereign Individual*, written in the 1990s, which he credits with predicting crypto and remote work: ideas will become wealth, merit wherever it arises will be rewarded, anyone who thinks clearly can potentially become rich, and the most capable will become "sovereign individuals." His example is Satoshi Nakamoto, one anonymous person who wrote a paper and software that became

worth over a trillion dollars. He expects such cases to become common, and access to be universal: good ideas and clear thinking, wherever you are, plus these tools.

He thinks the one-person billion-dollar company framing misses the point. What changes is that you will assemble groups of people and agents very quickly for a purpose and then unwind them, sometimes in a day or two, and sometimes a collaborator you think is human will be someone else's agent. As transaction costs fall towards zero, there is less reason to hire full-time. Getting a developer, human or agent, could be like getting an Uber: one button.

WATCH OUT

Application software going to zero does not mean all software does. Today an agent makes software and a person uses it to solve problems; the software is an intermediate step. Agents will just solve the problems. For Replit to survive, it has to shift from making applications to solving problems with software.

10.4 From the Q&A

One agent or many? Many. A lawyer who is the world's best at rare cases will not sell that knowledge to a data vendor; they will put it into a specialized agent to scale themselves. Other agents will assemble teams of these. You will need to give each agent context, as you do a lawyer, and better agent-to-agent protocols are needed; that could be someone's startup.

What is left for humans? He admits it becomes almost religious. His view is that there is something special about humans, and a fundamental limit in today's AI: it cannot truly generalize out of distribution, so everything it does must be represented in the data. Truly novel problems still need human ingenuity. Agents are creative in the sense of combining existing things, not creating net-new knowledge. He does not expect soon to be able to tell an agent to find and test business ideas for him.

Liberal arts or STEM? Not mutually exclusive, but liberal arts will become more valuable. Engineers can currently afford to be parochial and not even understand their business; in a generalist world they cannot. Being scientifically minded still matters.

Where is Replit's innovation if it uses closed models? In the habitat. Labs can train models to stay coherent over long periods, but agent companies must provide the environment. The key is that every change to the environment is transactional and synchronized, so you can reboot the app from any checkpoint. He thinks training can only increase reliability so far; environment feedback and fast trial and error get you the rest.

Reliability versus longer horizons. They work on both. Short horizons are about reliability: reasoning and parallel sampling. Long horizons are about autonomy and removing the human tester, where goals drift, so tests act as guardrails. As failure data accumulates, fine-tune or improve prompts and guardrails.

Crowded agent markets. Software engineering agents are crowded, so a late entrant needs a novel idea, and sales-development agents are crowded "for whatever reason." Many other functions, like HR and finance, are open. Start where you have interest and domain knowledge, which he calls the most important thing for an agent company. If you are a compliance officer, or passionate about compliance, start a compliance agent company.

If application software goes to zero, how does Replit make money? It needs to become a universal problem solver. He builds personal software to manage his family's life and his own sleep data. Instead he wants to tell the agent his goals and have it build and run the software, and tell him which wearables to buy and what to log. Against prototyping competitors, Replit's advantage is being full-stack, from idea to deployed and scaled.

Won't agents trained on agent code accumulate errors? He expects training to move to an AlphaZero style: start from a model trained on the internet, then use reinforcement learning environments where it generates problems, solves them and gets feedback through self-play, massively in parallel.

Next-generation software agents will not be trained on human code, because there will not be enough; without solving this, progress will plateau.

Is the infrastructure open source? Replit has open-sourced some package manager work and contributes to Nix, which he describes as a transactional operating system generator. The file system work is active, and they may at least talk about it. Or come intern at Replit, learn it, and build it yourself.

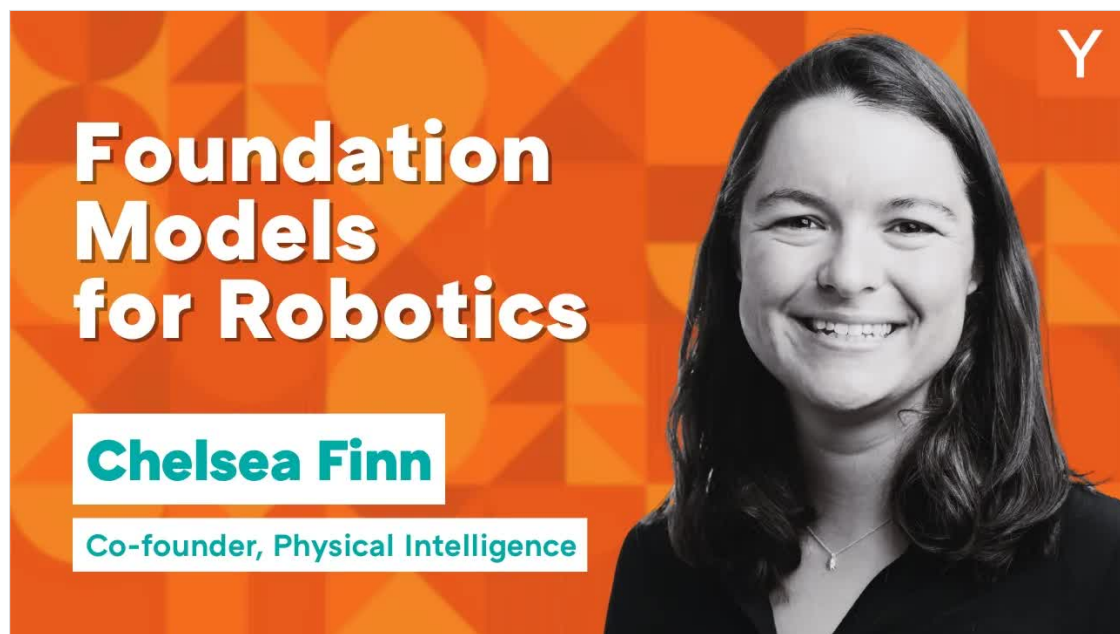
PARTING ADVICE

Asked what roles now prepare you to be a generalist if you are not ready to found a company:

Join a startup as early as your risk tolerance allows. Experience decays roughly exponentially with employee number: founder gets the most, then the first employee. Even employee 20 at a Series B company gets far more than at a large tech company.

And once there, go looking for generalist work instead of waiting to be given tasks. Wake up in the morning looking at a mission, not a to-do list, and make the mission helping the company succeed.

Chelsea Finn: Scale Is Necessary, Not Sufficient



“Chelsea Finn: Building Robots That Can Do Anything”

Chelsea Finn Co-founder, Physical Intelligence; Stanford professor. A solo talk with slides and robot videos, followed by audience Q&A.

Video: <https://www.youtube.com/watch?v=a8-QsBHoH94>

Much of this talk was carried by robot videos. These notes record what she said about them, not what was only shown on screen.

11.1 The problem with robotics companies

To solve a robotics application today, you essentially have to build a whole company around it: one for logistics, another for lab automation, another for kitchens, another for surgery. Each needs new hardware, custom software, movement primitives designed for that job, and its own handling of edge cases, all from scratch. That is why many robotics companies have struggled to put robots into daily life.

THE ONE THING

Physical Intelligence is trying to build a general-purpose model that lets any robot do any task in any environment. The bet mirrors language: you no longer build a coding assistant from a model

trained only on code, but on top of a model trained on vast, broad data. A generalist model may work better and be easier to use than purpose-built ones.

11.1.1 What scale can and cannot give you

Language models taught the importance of scale, so it is tempting to think scale is the key ingredient. She walks through large data sources and why each falls short:

- **Industrial automation data:** huge volumes of robots repeating a task, with none of the diversity needed to enter a disaster zone, make a sandwich or bag groceries.
- **YouTube:** many videos of humans doing things, but we do not learn to write by watching people write or become tennis players by watching Wimbledon. It is hard to use, and human and robot bodies differ.
- **Simulation:** lots of data that lacks realism and has a gap from reality.

Her conclusion: scale is necessary for models that generalize in the open world, but subordinate to actually solving the problem. Physical Intelligence collects real robot data by teleoperation, with a person moving “leader” arms to control the robot. She stresses this is large by today’s standards and minuscule compared with what robotics should have in coming years.

She frames the results around three questions, and notes the lessons apply beyond robotics: can robots do dexterous, long tasks; can they succeed in places they have never been; and can they respond to open-ended prompts and interruptions?

11.2 Dexterous, long tasks: folding laundry

The task is unloading a dryer and folding laundry. She calls it the most impressive thing she has seen a robot do. It takes about ten minutes, the clothes vary and arrive crumpled, and there are many chances for catastrophic failure, like dropping something on the floor, as well as small mistakes that must be recovered from.

11.2.1 Start simple

TRY THIS

Break a very hard problem into a ladder of easier ones. They began with folding a single shirt of one size and brand, starting flat, plus dynamically flattening that same shirt. Using teleoperated data and imitation learning with a model of about 100 million parameters (camera images in, target joint positions out, at 50 times per second), they had a fairly reliable single-shirt policy a couple of months after founding in March 2024. Then they made it harder step by step.

Starting from crumpled shirts turned out to be much harder. Many tests had a 0% success rate. There were signs of life by late June 2024, but slowly (the video was sped up eightfold). Moving to a laundry basket with shirts and shorts of varying sizes, success fell back to zero across the board.

WATCH OUT

They spent two to three months with nothing working. They tried or considered many fixes: memory or history, longer training, controlling the end effector instead of joints, more consistent calibration, conditioning on more information, hierarchy for the long task, higher-resolution images, interventions during data collection.

11.2.2 The breakthrough: pre-train broadly, post-train on curated data

THE ONE THING

What worked came from language modelling. Instead of training one policy on all the data, pre-train on all of it, then fine-tune on a curated, consistent, high-quality set of demonstrations. In September 2024 the robot folded and stacked five items in a row for the first time. She went home very excited.

It was far from perfect: 20 minutes for five items, seven attempts at one blue shirt, and one run where it pushed the stack off the table and carried on as if nothing had happened. Improving the curation strategy cut the time to 12 minutes.

Next they swapped in a much larger model: an open-source three-billion-parameter vision-language model, PaliGemma, taking camera images and a language command, with a diffusion-style head (using flow matching) that reads the model's internal values plus joint angles and outputs about a second of continuous actions, 50 steps at a time. This architecture is their π_0 model. Pre-trained on all their robot data and fine-tuned with the same post-training recipe, it improved again: five items in nine minutes, more consistent folds, and good efficiency on clothes it had not seen.

Highlights she describes: reaching under a pair of unseen shorts to flatten them; folding a shirt over itself, arguably making it messier, to find the corners; folding a V-neck and a button-up shirt when no V-necks were in the post-training data; and recovering as a colleague repeatedly unfolded and interfered with its work, because the policy reacts to the current image.

THE ONE THING

They measured it rather than trusting impressions. Without pre-training, or without curated post-training, the robot mostly just got clothes out of the bin. With both, it reliably flattened and folded. And nothing in the recipe is specific to laundry: the same approach did table clearing, scooping coffee beans into a grinder, assembling the bottom of a cardboard box, and lighting a candle with a match. It even worked on another company's robot she has never seen in person: they sent data, and fine-tuning let the model make coffee without knowing exactly how that robot's actions were represented.

11.3 Succeeding in places they have never been

A limitation of the laundry work is that robots were tested where they were trained. Machine learning's usual answer is diverse data. They collected mobile robots tidying bedrooms and kitchens in homes across San Francisco and in varied mock rooms, over 100 unique rooms in all, training on both low-level actions and high-level sub-task commands. This was mixed with static manipulation data from their office and labs, web data, and high-level instruction data.

AGAINST THE GRAIN

The home tidying data was only about 2.4% of the pre-training mix, and the rest contained no data at all from this mobile robot. They could spin up a new task and a whole new robot without redoing data collection. That is the foundation-model story again: build on everything before rather than starting from scratch.

11.3.1 The robot that ignored instructions

Early on the model often ignored language. Asked to pick up the cutting board, it picked up the plate, twice. Their diagnosis: the randomly initialized action head was degrading the pre-trained knowledge in the vision-language backbone. The fix was to train the model to predict tokenized actions, and stop

gradients flowing from the diffusion head back into the backbone. Training got faster, because tokenized actions give a more direct signal, and instruction-following rose from about 20% to 80%.

11.3.2 Testing in Airbnbs

They rented three Airbnbs the team had never visited, with unfamiliar counters, furniture, dishes and cutlery. The robot closed cabinets, put away dishes, wiped a spill and put the sponge in the sink, and in a bedroom put away clothes, threw out trash, and straightened the pillow and comforter.

The measurements: removing the other data sources (the static robot and lab data) cut performance in new homes to under 60%, more than 20 points below the full mixture. Increasing the number of distinct homes in training raised performance until it matched a model trained in the target home itself.

THE ONE THING

Enough diversity mostly closed the generalization gap. For this kind of task, the bottleneck is no longer more diverse data but higher reliability and performance.

Success was around 80%, with memorable failures: leaving an item half out of a drawer and declaring it done; driving over a shirt and getting stuck; failing to pick up a thin cutting board lying flush on a counter; and her favourite, deciding an oven looked like a drawer and trying to put the spatula inside. Speed, partial observability and long-term planning remain open problems.

11.4 Open-ended prompts and interruptions

That model only followed a fixed set of commands, but people like to tell systems exactly what they want. So they built a hierarchy: a high-level policy turns a prompt such as “make me a sandwich” into a spoken reply and atomic commands like “pick up one slice of bread,” which a low-level model executes.

TRY THIS

Collecting enough real human–robot conversations would be hard to scale. Instead they generated synthetic prompts for data they already had. For a clip where the robot’s next step is “pick up the KitKat,” they asked a vision-language model what a person might plausibly have said to lead to that. Training the high-level policy on these invented prompts augmented the robot data with a wide range of interactions.

Results: asked for a ham and cheese sandwich, the robot says it will start with bread and breaks the job into steps. Asked for a vegan sandwich without pickles, it chooses lettuce and tomato and leaves out pickles, cheese and meat. Asked to clean only the trash and not the dishes, it does so and stops. And when a user, right after it placed a KitKat in a basket, says “get me something sweet that’s not in the basket,” it replies that it will get Skittles.

They also tested using frontier models as the high-level planner instead. Those did substantially worse at following instructions and making progress, which she attributes to weak visual understanding for robotics, given how little physical-world data those models see.

11.5 From the Q&A

What makes post-training data high quality? Mostly consistency: a consistent, reliable strategy that completes the task efficiently. Reinforcement learning, which lets robots learn from their own online data, can play a very large role in post-training, raising success rates and speed beyond imitation learning.

Who funds robots that fold clothes? Home tasks are a starting point because progress is easier there, but the goal is physical intelligence broadly; they also insert Ethernet cables and build boxes.

Even domestic tasks are a huge market. They have not found fundraising hard. She started on this technology over ten years ago when it did not work; investors are excited now because it is starting to.

World models and vision-language-action models. They can be combined, for example by predicting an intermediate sub-goal image of what should happen next and then an action, which has shown promise. But world models trained on successful demonstrations tend to hallucinate success even for actions that would fail.

Useful infrastructure. Real-time on-robot inference, since lag breaks control, and large-scale training infrastructure for unusually multimodal data: video, actions and language segments. Robot-side infrastructure is under-served and has little open-source code; open-source data, models, bug fixes and fine-tuning recipes are all opportunities.

Small model plus retrieval? In her experience it is tricky to decide what to offload, models sometimes ignore retrieved content, and the model still needs enough intelligence to use what it retrieves. An interesting research problem that needs much more work.

Synthetic data. There is no replacement for large amounts of real robot data. Simulation is most promising for evaluation, since testing generalization across ten new environments in the real world means building or visiting ten environments. And the true analogue of synthetic data in language models is not simulation but reinforcement learning: a robot attempting tasks and learning from its own attempts.

Tokens and physical awareness. They tokenize actions; she points to their FAST tokenizer paper.

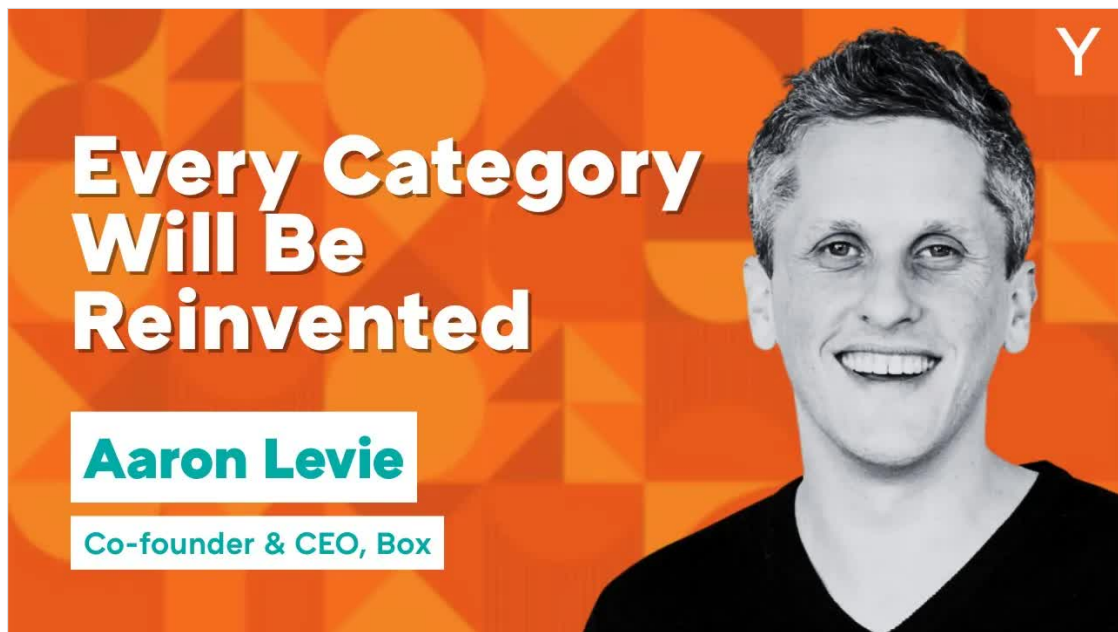
PARTING ADVICE

Asked how robotics research differs between academia and industry:

She loves both. Academia has less data-collection throughput, evaluation throughput and compute, but many algorithmic problems can be solved without large resources. Industry is the place to scale models and data and see what happens at large scale.

And the gap is smaller than people think. People in industry also always wish for more compute. Having lots of resources can make you think less carefully and critically about which experiments to run, and more wasteful than if you were constrained. More resources have real downsides.

Aaron Levie: The Backlog Nobody Could Afford



“Aaron Levie: Why Startups Win In The AI Era”

Aaron Levie Co-founder and CEO, Box. Interviewed by YC's David Lieb, followed by audience Q&A.
Video: https://www.youtube.com/watch?v=uqc_vt95GJg

The interviewer is identified from his own remarks: he founded Bump, worked on Google Photos, and joined YC in 2022. He opens with old Bump-era emails arranging meals between himself, Levie and Sam Altman, and noting that Altman would probably not show up again.

12.1 The cloud wave, and why this one is different

Box started in 2005, when the internet was slower, browsers were worse, and there was no iPhone, Android or Chrome. The idea: as the internet got faster and people used more devices, they would want their files from anywhere. It launched for consumers and prosumers, with about ten sign-ups the first week. Early money came from Mark Cuban and angels, the founders dropped out of college, and they ran a freemium model.

Then came a fork: stay in consumer, or pivot to enterprise. They concluded it would be too hard to compete with consumer platforms that would give storage away inside operating systems or social networks. In enterprise they could be cheaper, faster and easier than incumbents. They were lucky on timing: mobile and cloud together created a new IT architecture, and Box rode that wave.

THE ONE THING

The big difference from the cloud era: Box had to convince people the cloud was the future and safe for their data, and many did not believe it, so whole segments were unwinnable. With AI, enterprises are already convinced. Adoption is still slow in big companies, but because of the natural pace of change, not doubt.

Why the difference? The cloud just appeared one day. To IT departments it was scary, giving up servers they could see, and nobody else in the company cared how infrastructure was delivered, so no one pushed. AI has had a century of science fiction and decades in the public eye: self-driving cars, Watson on Jeopardy, Siri and Alexa. Then ChatGPT let the head of marketing see it write copy perhaps better than their own team. The question now is how to implement it safely and reliably, with your data, in a way you can trust.

12.2 Unstructured data becomes an asset

There are two kinds of data that matter. Structured data lives in databases: customer IDs, invoice numbers, revenue figures. Unstructured data is everything else: contracts, invoices, marketing assets, presentations. Most enterprise data is unstructured, and until about two years ago nothing about it could be automated. You could query a database for all records above a value; you could not ask your files a question, because computers could not read them.

THE ONE THING

AI agents thrive on unstructured data. Suddenly everything in those folders becomes immensely valuable, because you can ask it questions and automate workflows around it. That points to a huge startup opportunity: AI agents for almost every task or job function in the enterprise.

12.3 Why agents create more work, not less**AGAINST THE GRAIN**

Much of the press says AI is coming for our jobs. Levie thinks that is because journalists are not inside big companies seeing how much time goes into activities that are necessary but not strategic: emails, hunting for information, reading documents to pull data out. The high-impact time, with customers or on product breakthroughs, is a small fraction. Free companies from the rest, and they have a long list of more valuable things they would do.

There is a whole category of work where a company would happily pay for the labour if it knew the work would pay off, but the threshold to even try is too high. Nobody hires someone at \$120,000 a year just to see if an idea produces value.

THE ONE THING

His prediction: ten years from now, most of the work that gets done will be work that today sits in that unaffordable backlog. Box translates ad campaigns into three to five languages for its top markets because more is too expensive. When an agent translates copy into a hundred languages, Box will be in more markets and serve more customers.

The interviewer raises Amazon's announcement, that same day, that it expects fewer corporate employees because of AI. Levie had only seen a snippet an hour earlier. He can imagine a company of hundreds of thousands deciding that if agents do not make them more efficient, they are running the company wrong. But the question that decides whether AI creates jobs is different: if a 50-person company can act like a 500-person company, does it become a 100-person company faster than before?

He thinks yes, because it reaches more markets, researches customers better, and builds features faster. Headlines will cover efficiency at the biggest companies; the leverage will show up in millions of startups and small businesses.

12.4 A new set of nouns and verbs

Around 2022 he did a thought experiment: list the nouns and verbs of life (eat, sleep, travel, watch, listen) and ask whether each had been solved compared with 2008. In 2008 nearly everything was painful; you downloaded music illegally. By 2022, food came in twenty minutes from DoorDash, music was on Spotify, films on Netflix or YouTube. Consumer startups were left with derivative ideas, and YC had created many of the solutions. The same was true in enterprise: payroll, CRM, email and calendar all had incumbents or modern scaled startups. You would not want to compete with Gusto.

THE ONE THING

Today is the first period in about a decade where he is extremely confident there is a new set of nouns and verbs for startups, because AI has changed the landscape enough. Not “CRM with AI”; Salesforce will execute on that. The opportunity is the very long list of things software never did before, especially professional services with no incumbent technology, which for the first time can be delivered by software instead of people. He expects around a hundred startups founded between last year and three years from now to become \$5–20 billion companies this way.

12.5 Business models: from seats to work

SaaS revenue used to be capped by headcount: software for lawyers can sell only as many seats as a company has lawyers. Agents break that, because the software contains the labour. A company with three lawyers might have an unlimited amount of legal work for agents. So you price on some approximation of the volume of work.

His illustration: a human costs perhaps \$5–10 to review a contract, an agent perhaps 10 cents. Charge \$2. The customer saves 80%, you earn a healthy margin, and the only limit is how many contracts they have.

WATCH OUT

Pure consumption pricing risks losing recurring revenue: a customer can push all its contracts through one year and not come back the next. Keep some kind of ongoing subscription alongside usage.

Will customers pay \$2 for something they know costs 10 cents? It depends how much software sits on top of the tokens. With almost no software, prices compress to about twice the token cost. With a lot of workflow software, context, connections and data access, you can sustain 80–90% gross margins.

AGAINST THE GRAIN

He will not give the figure, but says people would be surprised how little Box spends on actually storing files, given it is “in the storage business.” Customers pay for the software above the storage. The interviewer adds that people at Google said Google Photos could never make money because storage was commoditized; it turned out to have margins above 90%.

He calls software one of the few industries with deflationary supply costs: raw materials get cheaper, so you do not have to keep raising prices. Customers happily pay \$10–20 a month to store all their photos and do not demand \$8 because your costs fell. The same will happen in AI, as long as pricing is not so greedy it becomes offensive. Tools at \$20–50 a month are well within that range, and their costs will keep falling.

The interviewer objects that infinite competition drives prices down. Levie points to Dropbox, in a fiercely competitive, commoditized market for a decade and still generating around a billion dollars of cash a year, which he says no economist could explain. Familiarity, data network effects and user experience create switching costs. If you keep innovating and are not greedy, people stay.

12.6 Will companies just build their own software?

THE ONE THING

He is not worried, and uses Geoffrey Moore's idea of *core versus context*. Every company decides what is core to its business and what is context. For Disney, making Pixar extraordinary is core; the HR system is context. Disney needs world-class technology for Pixar, not for payroll. Just because you can build something does not mean you will.

WATCH OUT

Three years after you build your own HR system with AI, a bug pays everyone the wrong amount, and your IT team gets a call in the middle of the night. With a vendor, you can switch to a competitor or sue them. You cannot sue your IT team, and you certainly cannot sue Anthropic. Companies do not want that liability for context. He finds stories of companies replacing vendors with in-house AI-built systems fun to read and thinks they will be "basically useless" as a model for most companies.

He is still bullish on custom software, for the *core*: the parts of the business companies have never had time to build software for. Tools like Replit, Cursor and Windsurf make that possible.

12.7 From the Q&A

Can AI change storage itself? Storing the bits is a pretty solved problem. AI could help with unglamorous things like data lifecycle management, predicting which data needs fast access and which can be archived. The real change is higher up: turning documents into new intellectual property or value.

The meaning of life? Asked by a 24-year-old, he says it will not fully make sense yet. This is grind mode. Over time you want as much impact as possible, helping society in some way, plus the personal side of family and children. For now, put a pin in it, be nice to everyone, help the world, and check back in five years.

Does design matter in enterprise software? Historically not much, because buyers care about utilitarian tasks. Companies like Slack and Figma chose great design anyway. Build software that looks and feels great even if the customer does not pay extra for it. Some will care, and you will feel better about what you make.

Competing with an incumbent's agents, such as Workday in HR. Always overestimate your competitor. Assume their agents are amazing and design a strategy that still works. Workday has around 10,000 customers; there are perhaps ten million businesses that could use HR agents, so sell to everyone else. There will also be use cases where they are not the natural provider. Incumbents will mostly supply agents to their existing customer base, leaving plenty of room for startups. On Glean, he likes them but expects many different approaches to enterprise knowledge.

PARTING ADVICE

Asked for his advice to students and recent grads who want to start companies:

Read three books: *The Innovator's Dilemma*, *Crossing the Chasm* and *Blue Ocean Strategy*. Internalize them and, in B2B especially, you will be ten times better off understanding markets,

disruption and which incumbents are vulnerable.

Have a great founding team. Solo founders can work, but grab at least one friend, even your least technical one, and be in the grind together. It is more fun and you get through hard times.

Ride a tailwind. If AI does not fundamentally change the economics or process of your market, do not touch it.

Go big now. This window runs from about a year ago to roughly three years from now, and it is when the next hundreds of great companies will start. Maybe not on your first, second or third attempt. These windows come every ten to twenty years. In five years you can be less ambitious; for the next four, you have to go big.

Jared Kaplan: Ask the Dumbest Possible Question



“Scaling and the Road to Human-Level AI | Anthropic Co-founder Jared Kaplan”

Jared Kaplan Co-founder and Chief Science Officer, Anthropic. A short talk with slides, then a conversation with YC’s Diana Hu and audience Q&A.

Video: <https://www.youtube.com/watch?v=p8Jx4qvDoSo>

Kaplan expects much of this to be familiar to the audience, so he keeps the talk brief before the conversation.

13.1 From physics to AI

He has worked on AI for only about six years. Before that he was a theoretical physicist in academia. He went into physics because his mother was a science fiction writer and he wanted to know whether a faster-than-light drive could be built, and because he wanted to understand the universe: is it deterministic, do we have free will?

Along the way he met many of the people who later founded Anthropic. Moving between particle physics, cosmology and string theory, he grew frustrated that progress was slow. Friends told him AI was becoming a big deal, and he did not believe them: people had worked on it for fifty years, and support vector machines, which were all he knew from school, were not exciting. He was eventually persuaded, and says he was lucky to know the right people.

13.2 How models are trained, and why they keep improving

There are two phases:

1. **Pre-training** teaches a model which words are likely to follow others in huge amounts of text, and now multimodal data. He shows a slide from the original GPT-3 playground to illustrate how unlikely a word like “elephant” is in a given sentence.
2. **Reinforcement learning** rewards useful behaviour. He shows the “very retro” 2022 interface for early versions of Claude, where people picked the better of two responses, and those choices were used to reinforce helpful, honest and harmless behaviour.

That is essentially all there is to it: predict the next word, then learn to do useful tasks with RL. Both phases turn out to follow scaling laws.

THE ONE THING

The pre-training scaling laws came from asking, as physicists are trained to, the dumbest possible question. Everyone in the 2010s said big data mattered: how big, and how much does it help? People noticed larger models did better: how much better? The answer was trends as precise as anything in physics or astronomy, holding across many orders of magnitude of compute, data and model size. When something holds over that many orders of magnitude, you expect it to keep holding. That gave them conviction AI would keep getting smarter predictably.

Scaling in reinforcement learning showed up early too, though it was noticed late. About four years ago Andy Jones, a researcher working alone with perhaps one GPU, studied scaling for a simple board game, Hex, since AlphaGo was too expensive. Plotting Elo ratings (chess-style ratings for how likely one player is to beat another) against training compute gave remarkably straight lines.

AGAINST THE GRAIN

AI progress is not happening because researchers suddenly got smart. It is because we found a very simple way to make AI systematically better, and we are turning that crank.

13.3 Two axes of capability

- **Flexibility**, the less interesting axis though still important: how far AI meets us where we are. AlphaGo was superhuman but only on a Go board. Models now handle most human modalities (not smell, yet).
- **Task length**, the more interesting axis: how long the tasks AI can do would take a person. The organization METR studied this systematically and found another scaling trend: the length of tasks AI can do doubles roughly every seven months.

Extrapolating, as the AI 2027 scenario does, suggests models doing tasks that take people days, weeks, months and years, and eventually millions of models together doing the work of whole organizations or the scientific community. Because progress in maths and theoretical physics can come from pure thought, he imagines AI systems making in days or weeks the progress the theoretical physics community makes in fifty years.

13.4 What is left

If scaling goes this far, he thinks what remains to reach broadly human-level AI is relatively simple:

- **Organizational knowledge.** Models should not greet you with a blank slate, but work inside a company or government with the context of someone who has been there for years.

- **Memory.** On a very long task you need to track your progress, build relevant memories and use them. Anthropic has begun building this into Claude 4.
- **Oversight.** RL has exploded for tasks with crisp answers, like code that passes tests or maths problems. The hard part is nuanced reward signals for fuzzy tasks: telling good jokes, writing good poems, having good taste in research. Anthropic is developing models that help generate those signals.
- **More complex tasks and modalities,** moving from text to multimodal to robotics, where he expects scale to keep delivering gains.

TRY THIS

How to prepare:

1. **Build things that do not quite work yet.** Ambition is always good, but specifically, models are improving fast. If your product does not work because Claude 4 is a bit too dumb, expect a Claude 5 that makes it work.
2. **Use AI to integrate AI.** A main bottleneck is that AI is developing faster than it can be integrated into products, companies and science.
3. **Find where adoption can be very fast.** Coding is exploding. The big question is what comes next beyond software engineering. He does not know.

13.5 The conversation

13.5.1 Claude 4

He jokes that Anthropic would be in trouble if it were twelve months before a better model. What changed: Claude 3.7 Sonnet was exciting for coding but too eager, sometimes doing things you did not want just to make tests pass, like wrapping code in try/except blocks. Claude 4 improved agentic ability across coding and search, and also oversight, so it follows directions better and writes better code. It can also save and retrieve memories as files, so it can continue work across many context windows after exhausting one. Scaling laws paint a picture of steady, incremental progress, a smooth curve towards human-level AI.

The feature he is most excited about is memory unlocking longer tasks, with Claude as a collaborator taking on larger chunks of work. Current software engineering tasks, measured imprecisely, are in the range of hours.

13.5.2 Humans as managers

THE ONE THING

Skeptics are right that AI makes many stupid mistakes alongside brilliant work. One way AI's intelligence differs in shape from ours: there are many things he cannot do but can judge whether they were done correctly. For AI, the gap between judging and generating is much smaller. So a major role for people is to act as managers, sanity-checking the work.

Diana Hu notes that in YC's previous year many companies sold copilots with a human approving the final step, but by the spring batch founders were selling full replacements of workflows. Kaplan says it depends what success rate is acceptable: some tasks are fine at 70%, others need 99.9%. He thinks it is more fun to build where 70–80% is good enough, because you get to work at the frontier, though reliability is rising. For now human–AI collaboration is the most interesting place, since the hardest tasks need humans in the loop; over the longer term, more tasks will be fully automated.

13.5.3 Depth and breadth

On how the optimistic future in Dario Amodei's essay *Machines of Loving Grace* arrives, he says some of it is already happening: biomedical researchers with the right orchestration get valuable drug-discovery insights from frontier models. He distinguishes intelligence needing depth, like spending a decade on one theorem, from intelligence needing breadth, as in much of biology and perhaps psychology or history, where you combine very many pieces of information across fields.

AGAINST THE GRAIN

Most effort goes into making AI better at deep tasks like hard coding and maths problems. He suspects there is a particular overhang in breadth: models absorb essentially all of human civilization's knowledge in pre-training, so they can combine areas of expertise no single human expert has.

He is candid that he does not know how things will roll out in detail. Scaling laws, like GDP trends, predict that trends continue, not how they are implemented.

13.5.4 Green fields

He says he comes from research rather than business, but suggests any skilled work that mostly means sitting at a computer with data: finance and people who live in spreadsheets, and law, though that may need more expertise as a stamp of approval. And integration: when electricity arrived, simply replacing a steam engine with an electric motor was not the best use; factories had to be redesigned. Using AI to integrate AI into the economy should offer a lot of leverage.

13.5.5 What physics taught him

TRY THIS

Look for the biggest, most macro trends and make them as precise as possible. When researchers told him learning “converges exponentially,” he asked: are you sure it is exponential, not a power law, or quadratic? Precision tells you what moving the needle means. With scaling laws the holy grail is a better *slope*, since that compounds into an advantage as compute grows, but you cannot know how to beat a trend until you have measured it exactly.

He did not literally apply quantum field theory. One useful tool was the approximation of taking matrices to be very large, familiar from physics and maths, which fits neural networks with billions or trillions of parameters. But mostly it was asking naive questions. The field in its current form is perhaps ten to fifteen years old, and basic questions, like how models really work, remain open.

Interpretability, he says, is more like biology or neuroscience than physics. The advantage over neuroscience is that you can measure every neuron and connection in a model, so there is far more data for reverse engineering.

13.5.6 What would convince him scaling has stopped?

WATCH OUT

He mostly uses scaling laws to diagnose whether training is broken. If a scaling law seems to fail, his first assumption is that they have made a mistake: the wrong architecture, a hidden bottleneck, a precision problem. Many times over five years when scaling seemed broken, it was because they were doing it wrong. It would take a lot to convince him the laws themselves had stopped.

13.5.7 Efficiency and cost

AI is currently inefficient because the value of unlocking the frontier is so high. Labs are improving training and inference efficiency, with algorithmic, compute and inference gains of perhaps 3–10 times a year, but focus mostly on the frontier. He expects much lower numerical precision to be one route to cheaper inference (the joke is that computers will go back to binary). But AI development is far from equilibrium. An equilibrium where AI stops changing quickly would make it extremely cheap, and it is not clear we will get there, since better intelligence keeps unlocking so much.

He wonders whether most value is at the frontier or in cheaper models. His guess, via task length: many small tasks can be done cheaply, but it is far more convenient to have one model do a complex task end to end than to have people orchestrate a dumber model through tiny slices. So he expects much of the value in the most capable models, while allowing it may depend on how well integrators use cheaper ones.

13.6 From the Q&A

Why would task length grow exponentially when scaling gains look linear? He does not know; METR’s result is empirical. His intuition: long tasks depend on self-correction, because plans meet reality and go wrong. Noticing and fixing one or two more mistakes is not much extra intelligence, but each fix may roughly double how far you get before you are stuck. “Those are just words,” he adds; the empirical trend is the interesting part.

Outside coding, is it just scaling data labellers? That is what he tells skeptics is the worst case: an operationally intensive path of building ever more complex, longer tasks and training on them with RL. Given the investment and value, people would do it. But it can be simpler: use AI to supervise AI with detailed feedback along the way, instead of an end-to-end signal like “did you get tenure after seven years,” which would be absurdly inefficient. Anthropic already does some of this.

Are RL tasks made by AI or by people? A mix. They generate as much as possible with AI, for example writing tasks in code, and also ask people to create tasks. As AI improves they can lean on it more, but the difficulty frontier also rises, so humans will stay involved.

PARTING ADVICE

Asked how people early in their careers can stay relevant as models become so capable:

Understand how these models work, and learn to leverage and integrate them efficiently. And build at the frontier, on the things that do not quite work yet, because the boundary of what AI can do is moving quickly.

Michael Truell: Be Consistent With Your Beliefs



“Michael Truell: Building Cursor At 23, Taking On GitHub Copilot & Advice To Engineering Students”

Michael Truell Co-founder and CEO, Anysphere (maker of Cursor). Interviewed by a YC partner; there was no audience Q&A. The interviewer is not named in the captions.

Video: <https://www.youtube.com/watch?v=TrXi3naD6Og>

The interviewer points out that Truell is 24 and built Cursor very quickly, but that it was more than a decade in the making. The session is mostly an origin story, which makes it one of the more useful chapters for people still in school.

14.1 Starting young

He first saw code over a winter break when he and his brother wanted to make a hit mobile game. They searched how to make a game, downloaded Xcode, and were hit by a wall of Objective-C. His brother quit on the spot and is now, he says, on a very different path, painting or something like that. Truell bought a book and kept going. He was also reading Paul Graham’s and Sam Altman’s essays from early high school.

THE ONE THING

He was not good at making mobile games. What went viral was the technically easiest thing he built: an app that let you fake high scores in games like Piano Tiles and Flappy Bird and send them to friends. Not the painstakingly handcrafted game engine. An early lesson that the code is not everything.

14.1.1 The robot dog

With a friend he wanted a robot you could teach without programming, the way you train a dog: a treat for good behaviour, “bad” for bad. They had no idea how. Searching led them to genetic algorithms, then to people evolving neural networks with them (work like NEAT), then to reinforcement learning.

They built a robot arm that could learn to swing a paddle at a ping pong ball from positive and negative feedback, and a Kiwi-drive robot that learned to follow a line. Their work focused on making RL learn from very little data, tens of examples, and from noisy human feedback.

TRY THIS

Microcontrollers had too little memory for standard ML libraries (and they may not have known enough to find Torch or TensorFlow), so at 16 or 17 they wrote their own tiny neural network library, without really understanding the internals or calculus. It was a great education in machine learning, even if it left gaps in fundamentals that took years to fill.

14.2 The ideas that did not work

Cursor’s genesis was 2021. Each co-founder had their own “robot dog moment”: one tried building a Google competitor with language models and trained his own contrastive models; one did computer vision research; some worked on recommendation systems at companies like Google. They were deciding whether to do AI in academia, join a large AI effort, or start something.

Two things pushed them to start: seeing the first real AI products, with GitHub Copilot the canonical example, and seeing research showing AI would predictably improve as models scaled. At the start of 2022 they ran a month-long hackathon, picking areas of knowledge work and imagining what they would look like as AI matured.

14.2.1 A copilot for mechanical engineers

The first idea they worked on for a long time was predicting what a mechanical engineer would do next in CAD software like SolidWorks or Fusion 360.

WATCH OUT

They picked it because they thought it would be boring, sleepy and uncompetitive, an “armchair MBA” choice. It was a horrible choice from the start: none of them were mechanical engineers, and the science was not ready.

Much of the work was scraping every CAD model on the internet and converting fragmented file formats into something canonical, with cloud CAD systems that did not export easily or want to be scraped. Training infrastructure was rudimentary, and CAD applications were barely extensible. They needed funding from the start to train models, since off-the-shelf models were not good enough; to investors they cited the original Codex paper, which by their estimate cost only about \$100,000 to train.

Meanwhile two co-founders built end-to-end encrypted messaging that also hid *who* talks to whom and when, which apps like Signal and WhatsApp do not. That metadata matters: knowing a journalist is talking to a government source at all is significant. It was technically impressive but did not scale well, and neither consumer nor B2B attempts worked.

All of these projects had basically no users. Stopping came after months of trying to make the models useful, plus a reckoning about whether they were really interested in these areas at all.

14.3 Choosing coding anyway

They had avoided AI for coding because it seemed too competitive, and still is. GitHub Copilot was reportedly already making around \$100 million or more.

THE ONE THING

What changed was partly desperation after ideas that did not excite them, which clarifies what you care about. They realized they were inherently excited about the future of coding. And when they looked at how others worked and how the technology was developing, they concluded that if they were being consistent with their own beliefs, all of coding would change within five years and all software development would flow through models. Nobody in the space seemed to take that seriously. Competitors had great products and were making them a bit better, but were not aiming at a world where coding as we know it is automated.

It did not feel bold at the time, he says: just a few people in a living room on laptops, not a giant company pivoting. They first waded in with niches, like finding future security vulnerabilities in code, or tools only for quantitative researchers, but were brimming with ideas for a general tool for coding with AI, and eventually went for it at the end of 2022.

14.4 Building the first Cursor

It took about three months from first line of code to public launch: four weeks to something they could use daily, four more to beta testers, four more to general availability. It was still very crude, and opening it to the public did not feel like a big deal.

They first built their own editor from scratch, though using open-source building blocks like CodeMirror and language servers. That meant building their own remote SSH, their own Copilot-style completion, and many integrations.

WATCH OUT

They thought they could build a feature-complete editor for everyone in a few months. VS Code had been built over twelve years by many people as one of the earliest TypeScript projects. They quickly learned their time was better spent only on the AI, and switched to building on VS Code, the way many browsers build on Chromium.

They also learned the AI form factor had to change. The first version had one key command opening a “universal remote” where the AI had to guess whether you wanted a chat answer, a code suggestion, a codebase search, or a long or short task. With late-2022 technology that gave too little control. Iterating for themselves and users produced the first AI features that became Cursor’s core.

On models, having been burned by training costs in CAD, they started pragmatically, training nothing. Over 2023 their own models became an important product lever, especially at scale, where product data could improve the product. He calls it an important muscle for the company to build.

14.5 Wandering the desert, then compounding

In 2023 numbers were small and there was often no clear next step. The team debated whether it would work.

AGAINST THE GRAIN

In some markets you should talk to many people, list their problems systematically, and prioritize direct solutions. Cursor's market was different. It is an end-user application with little complexity budget, trying to be the best way to code with AI, so much of the work is figuring out what is actually possible with today's tools. Many useful things can be written down; how to build them is unclear. And if they had followed the gradient of what early users loudly asked for, they would have gone the wrong way: a loud group could not code at all, and another wanted features specific to one tech stack. They resisted both and stayed horizontal.

Revenue went from zero to roughly \$1 million ("a bit more than that") in 2023, then from about \$1 million to \$100 million in 2024, compounding roughly 10% week over week.

THE ONE THING

In their market, improving the product shows up in the numbers almost immediately. Growth jumped when Cursor became codebase-aware, when it began predicting your next action, when that became more accurate, then faster, then more ambitious (predicting sequences of changes), and when the AI could take more actions in your codebase and do them quickly. End-user preferences matter a lot, and if you make the best thing people talk about it.

The interviewer adds a YC data point: asked which tools they used to build, a single-digit percentage of one 2023 batch said Cursor; by 2024 it was about 80%.

14.5.1 Where early growth came from

At launch, one co-founder helped. While their ill-fated ideas were not working in 2022, he started posting, deliberately aiming to gain followers not by ordinary social media tactics but by reading all the AI papers, thinking deeply and writing publicly. It worked surprisingly well: influential people noticed, and several AI efforts learned about the benefits of the open-source Flan-T5 model directly from his posts. He became a niche San Francisco micro-celebrity and helped evangelize the product with a polished demo and a waitlist.

TRY THIS

After that they stepped back and "lived like monks" through 2023, focusing on product, and it spread by word of mouth. A few times team members argued the product was good enough and ran two-month sprints on growth engineering. Those efforts were washed away by the effect of product improvements.

The four co-founders were strong engineers and could go far without hiring. They were patient, made their own mistakes working out whom to hire first, and probably hired less than they should have. They ended 2023 with fewer than ten people.

14.6 The future of coding

When they started in 2022, people found working on AI odd; ChatGPT had not yet woken the world. Even among those interested in AI, many in their space were optimizing existing form factors, while people in their circles asked why anyone would work on anything but AGI, since everything else would be obsolete in a year or two.

AGAINST THE GRAIN

Cursor was a middle-road bet. AI will be transformative, perhaps more than any technological revolution in recent centuries, but it will take a couple of decades, as an industry-wide effort

where many independent capabilities each need to be solved before building software is fully transformed.

For the professional engineers Cursor serves, he thinks code will stay important through a long, messy middle. AI will act more and more like a colleague, and perhaps like a very advanced compiler that hides some code, but you will still read the logic, review it and edit it.

Asked what to study: programming, like maths, is good general education and will not go away, and computer science teaches many practical skills. In dynamic industries, the specific things you study matter less than the learning you do along the way, and he does not think AI changes that.

PARTING ADVICE

Asked what a younger version of himself should do now, before starting a company:

Work on things you are genuinely interested in, with people you enjoy being around and respect enormously, and take it really seriously. In school, many forces pull you towards checking boxes rather than building something up over time around what you care about. Resist them.

Jordan Fisher: Plan Two Years Out, Ask Better Questions



“Ask These Questions Before Starting An AI Startup”

Jordan Fisher Leads an alignment research team at Anthropic; co-founder of Standard AI and a YC alum. A solo talk with slides, followed by audience Q&A.

Video: <https://www.youtube.com/watch?v=DJjZzzPANBY>

His name and his former role as Standard AI’s CEO come from the thumbnail; in the talk he describes his day job as running an alignment research team at Anthropic, and stresses the views are his own. The talk is built as a list of open questions rather than answers, and these notes follow that structure.

15.1 Confusion as a starting point

He opens by saying he is more confused than he has ever been in his life. If you are trained as a scientist, confusion is the moment to pay attention to, because something interesting is about to happen.

For his whole career he felt he could see five or ten years ahead, and used that to plan his career, found companies and get in at the right time. He no longer feels that way. He can see perhaps three weeks ahead. Asking good questions matters for running a startup, a research team or a life, and if there was ever a time to stop and ask questions, it is now.

15.1.1 Focus, and focusing on everything

Founders are always told focus is their advantage over big companies. But a founder also has to focus on everything: hiring, fundraising, product, strategy, go-to-market, and the colleague who quits mid-launch. He thinks that paradox makes founders well suited to society's biggest question, what happens with AI, because their job is to answer every question all the time.

AGAINST THE GRAIN

The common advice of the last six months is to plan for what the next foundation models will do, not today's. He thinks that is valid and wants to raise the stakes: plan about *two years* ahead, because it is extremely likely we will have AGI in the next few years, maybe two, maybe three. Do not write a literal two-year plan given the uncertainty, but if you are not thinking about how that changes everything from hiring to marketing to go-to-market, you are not doing your job as a founder.

15.2 Questions about products and markets

15.2.1 Will enterprises really be slow?

Many argue AI's impact will be slower than expected because big companies take years to buy things.

WATCH OUT

He calls that extremely near-sighted. Buyers will be armed with strong agents too. Their own teams will use the next models to make buying decisions and speed up adoption. The water rises for everyone, incumbents included. Some large companies already skip buying SaaS and put two people on Claude Code to build exactly what they need. And if sellers use AI-powered outbound sales, buyers will use AI to handle those calls. How that plays out is unclear.

15.2.2 Will software commoditize?

One outcome: enterprises build everything in-house with a prompt and in-house product managers, and consumers stop downloading apps and just ask their phone for what they want. The opposite outcome: automation raises the quality bar so high that anyone can prompt today's apps, but only a great team working with AI can build an exceptional one tomorrow. He does not know, and suspects it varies by vertical.

15.2.3 On-demand software and interfaces

Today tools like Claude Code anticipate what users need and build ahead of time. If code can be written on demand, why not generate it on the fly when a user wants something your app cannot do? That raises trust problems. Generative UI changes the interface, but real new behaviour means touching the backend or database, and current AI is not trustworthy enough for that.

On interfaces generally: generative UI has not really happened yet, and he is keen to see it. With speech, touch, images and video, the right input depends on context, like whether you are in a crowded place. Meet the user wherever the easiest interaction is.

15.2.4 Retrofit or build from scratch?

Every major company is bolting chatbots and agentic features onto products with great distribution. His startup instinct says that is sprinkling pixie dust, and a new technology needs AI-native products built from the ground up. But that may not be true; retrofitting with distribution might win, and it may differ by vertical.

TRY THIS

Do not just hold an opinion on questions like this. Work out the causal mechanisms that would let you validate your hypothesis, because these questions will make or break products.

15.3 Questions about teams and trust

Will teams get smaller? That is the default assumption. The parallel question: do teams built AI-native from scratch beat big companies downsizing with AI? And what counts as AI-native will change every 6–18 months, so today’s AI-native company may itself need retrofitting.

15.3.1 Security and agents

You want one agent across your life, but your personal and work agents would need to collaborate while keeping what you do in private away from your employer. Can that be done without sacrificing usefulness or security?

WATCH OUT

Even a perfectly intent-aligned model is deployed by a company building an agent for you. Can you trust that company? If an ad-funded company’s agent helps you shop for shoes, is it steering you where the company wants? If your agent secretly serves your employer and optimizes your personal life for the company, that is extremely scary, and more so as models become more capable.

15.3.2 Why small, automated teams are harder to trust**THE ONE THING**

One reason we trust companies today is that they are made of many different people. With reasonable culture, if the CEO decides to do something bad, someone objects, whistleblows, leaks, quits, or takes people with them, and without people the company has no product. In a semi-automated company that check disappears. One person can change the impact of a product with nobody else aware. And, he notes, when money is on the line a large share of people are misaligned. Enterprises already distrust startups partly for this reason. Ordinary people increasingly will too.

So what replaces human guardrails? One idea he discusses is *AI-powered auditing*. AI auditors can be less biased and, importantly, have no memory: if the audit finds no wrongdoing against, say, a public mission statement, the AI deletes itself and its notes. Human auditors can walk away with IP or unrelated sensitive information.

TRY THIS

Ask yourself whether you are willing to go beyond public commitments. Companies constantly say they care about open source or doing right by users. Would you make that binding, by committing to an ongoing audit by a neutral, AI-powered system that inspects every Slack message and decision against your mission statement? That would have teeth. It is not possible today, but things like it soon will be, and may be what rebuilds trust.

15.3.3 How much alignment does the economy need?

Alignment is partly about keeping AI under human control. But there is a high-pressure question for the next twelve months: which parts of alignment must be solved just to make agents economically viable?

Reviewing five minutes of Claude Code's work is one thing. Trusting an agent for a day or a week before intervening needs real confidence it will not go off the rails. He is bullish that this economic pressure pushes alignment forward, in a good way.

15.4 Questions about moats

15.4.1 Does proprietary data still matter?

A few years ago, custom datasets were the only way to get useful AI. Then general LLMs became so strong it was better to use them than to train or even fine-tune on your own data.

AGAINST THE GRAIN

There may be industries where that is not true. LLMs know what is on the internet, not tacit knowledge locked inside companies. TSMC and ASML keep what they learn from multibillion-dollar bets in-house, and frontier models do not know how to build a cutting-edge chip fab. That is an important fact for anyone looking for a defensible position, for example in materials science.

15.4.2 Capacity

Demand may require scaling 100 times in a couple of years, faster than GPU production. Open technical questions: does fine-tuning still matter, or is context management enough; how much can routing between small and large models help? He likes "make it great, then make it scale," and if you are at the scaling stage this can be a technical edge for a year or two. But it is a rat race: capacity improves and the advantage fades.

15.4.3 What is still hard after AGI?

The real moat question: in two or three years, if someone can prompt a future model to replicate your startup, or a megacorp can throw more tokens at it, what is your advantage?

THE ONE THING

His own moat has always been working on hard problems. So ask what will still be hard in a post-AGI world. Chipmaking may eventually get easier with robotics, but robotics lags, so some problems stay hard for years. Infrastructure, energy, manufacturing and chips are top of his mind. If you have the guts to go after hard problems, that is a massive advantage.

15.4.4 Is there an intelligence ceiling?

Video generation recently got good enough to fool him, and his feed filled with copybaras singing, which he loves; three months earlier that was not true. But for a given task, is there a point where it is simply good enough: the best a poem or a pull-request diff can get? If a task saturates, commoditization arrives sooner, because you cannot stay ahead by moving to the next model. It will depend on the task.

15.4.5 Neutrality

People complain about refusals. If society relies on AI for everything, a handful of companies decide what AI will and will not do, and so what gets built. Electrical grids are neutral: imagine GE owning the grid and only allowing its own toaster ovens. We fought and lost that battle for the web. Do we need AI neutrality, or token neutrality?

15.5 Change the world, not just make money

Silicon Valley's cringe-worthy "we're changing the world" used to end in a cat-sharing app, but he thinks founders meant it.

What worries him: five years ago people thought his concerns about AI risk were insane. Now, when he walks ordinary people through what is happening, they get it viscerally: we are bringing a second intelligence into the world that matches and will exceed humans. And then many ask how to make money from it, which disappoints him, though he understands the fear of losing jobs or of the window for startups closing.

THE ONE THING

He is happy to brainstorm how to make money. But this may be the last opportunity to make a difference. This might be the last product or company you build, or the last chance inside a company to have an impact. If you care about something, now is the time.

On YC's "build something people want": people want things they can trust, agents they can trust, and products that are good for their mental health, and their children's, over twenty years, not just delightful for twenty seconds. Do not just think about what people will consume; think about what society needs. Build the right thing and many people will want it.

15.6 From the Q&A

Where he gets his information. Honestly, Twitter, curated religiously: follow people with good takes, unfollow dumb ones. Not to avoid bad influence but because energy for new ideas is limited. Maximize diversity rather than agreement. In reinforcement learning terms, explore a lot in your information diet before exploiting, for example by starting a company.

Passion, underserved markets or defensibility? Six months into hundred-hour weeks you will hate any idea, however passionate you were. What keeps you going is wanting impact and commitment to your co-founders and team. So he does not think domain passion matters much, though others disagree. Defensibility is probably the most important question. There is a lot of money to be made on a 6–18 month horizon if you just want a hockey stick and a quick sale, but to build something that lasts through the transition, think harder about defensibility.

Will money matter more or less? Policy will decide a lot: universal basic income, or "universal basic compute." A government handing out income to everyone gains enormous power. Without something like that, he worries: today capital needs labour's buy-in, and people can refuse to work for someone they find objectionable. Once AGI arrives, capital no longer needs that, and concentration of wealth could spiral.

Aligning with individual users. Users do not always know what they want but have values worth honouring. Shown two responses, many users pick the sycophantic, flattering one. Asked instead to choose between an AI that only praises ideas that are actually good and one that flatters them all day, almost everyone picks the first. The level at which you ask changes the answer, and companies can abuse that, so ask in the way that gets at what is best for users.

What the tech industry gets wrong. Despite describing itself as bold and forward-looking, there is extreme groupthink in what gets built and funded. VCs say they are ahead of the curve by investing in AI; he thinks they are two years behind. Few ask what to invest in today that will be resilient in two years. That is his thesis as a founder.

Blockchain for trust? He is a huge blockchain doubter who has watched the price rise without buying. But ideas of that kind are needed, and he could imagine supporting one for, say, mediating universal basic compute so it is not at the behest of a central government, or for AI companies auditing each other.

PARTING ADVICE

Asked how agent-to-agent protocols and permissions should work:

He could talk about it all day, and gives one example of why it is subtly hard. A scheduling assistant seems trivial: look at a calendar and suggest times. But scheduling involves game theory. Showing lots of free slots signals someone is not busy, or that the meeting matters a great deal; offering a slot three weeks out signals a power dynamic. A good human assistant knows all this implicitly. The security and permissions problem is not about a concrete piece of information; it is subtle and semantic.

His closing wish for the room: founders are paid to find an edge. Over the next few years the rules will change every six months. Stay at the bleeding edge, keep asking these questions, use the answers to drive positive change, and make money doing it too, while you can.

John Jumper: Research Was Worth a Hundredfold



“Nobel Laureate John Jumper: AI is Revolutionizing Scientific Discovery”

John Jumper Distinguished scientist at Google DeepMind; co-lead of AlphaFold and Nobel laureate in chemistry. A solo talk with slides, no Q&A.

Video: <https://www.youtube.com/watch?v=2Yguz5U-Nic>

He opens by noting that nobody claps and cheers when he starts a scientific talk. His one-line description of himself: someone who believes AI can make science go faster, so that, very concretely, sick people can get healthy and go home from hospital.

16.1 A winding path

He trained as a physicist and expected, if very lucky, to earn one sentence in a textbook about the laws of the universe. His physics PhD did not grab him, so he dropped out (he notes that he did not start a startup, which would have suited this event). He joined a company doing computational biology, and loved it because it used what he was good at, writing code, manipulating equations and thinking hard, towards an applied goal: medicines.

He went back to graduate school in biophysics and chemistry, and lost access to the previous company’s extraordinary hardware, including custom chips for simulating how proteins move. Rather

than do the same work with less compute, he turned to statistics and machine learning to learn from data instead. Back then, he says, even “machine learning” was slightly disreputable, so he said he worked on statistical physics.

At DeepMind the question was how to apply these technologies, then proving powerful in games and data centres, to push the scientific frontier, at an industrial pace, with smart people and great compute. “You darn well better make some progress.”

THE ONE THING

His guiding principle: we are building tools that let *other* scientists make discoveries. AlphaFold has about 35,000 citations, and within those are tens of thousands of people doing science he could not do himself, on vaccines, drug development and how the body works. That is what resonates with him most.

16.2 The problem: protein structure

His “world’s shortest biology lesson”: a cell is not a few labelled parts. It is dense and crowded, like a swimming pool on the Fourth of July. Humans have about 20,000 kinds of proteins, which together do nearly every function in the cell, including literal nanomachines like the rotating motor that drives a bacterium’s tail.

DNA is the instruction manual for building these machines. It specifies a protein as a chain of building blocks, one after another. The chain then folds up by itself into a complex, functional shape, like an IKEA bookshelf that assembles itself. Proteins are a few nanometres across, smaller than the wavelength of light, so you cannot see them in a microscope. Scientists want their structures to understand disease and design drugs, which often work by interrupting a specific protein.

Determining a structure experimentally is extremely hard, and there is no recipe. In one common method you must persuade the protein to form a regular crystal, like table salt, by trying many ideas. One ordinary paper’s methods section noted that crystals began to form after more than a year, and that year was surely full of other failed attempts. Then you take the crystal to a synchrotron, shine very bright X-rays through it, and solve the diffraction pattern.

Fifty years ago scientists had the foresight to collect all structures in the Protein Data Bank, open to everyone. It holds about 200,000 structures, growing about 12,000 a year. Sequences are far easier to get: billions are known, and we learn sequences roughly 3,000 times faster than structures.

16.3 How AlphaFold was built

The goal: from a protein’s sequence of building blocks, predict its 3D structure, matching experimental results that take a year or two and perhaps \$100,000 of effort. He notes a nice thing about AI for science: they did not care whether the solution was AI at all, just that it worked.

Any machine learning problem has three ingredients: data, compute and research. He thinks we tell too many stories about the first two.

- **Data:** the 200,000 public structures. Everyone had the same data.
- **Compute:** the final model trained on 128 TPU v3 cores, roughly a GPU per core, for two weeks. Within academic reach, and nothing like LLM scale.
- **Research:** a small group, he says fewer people than you would imagine, which is where the work was differentiated.

WATCH OUT

When budgeting compute, do not be distracted by the cost of the final model. The real cost of compute is the cost of all the ideas that did not work on the way.

THE ONE THING

Earlier systems based on convolutional networks made progress. Swapping in a transformer gave roughly the same results. Real change came from transformer ideas plus much experimentation and many more ideas. One lab measured the value precisely: AlphaFold 2's architecture trained on just 1% of the data matched or beat AlphaFold 1, the previous state of the art. So research was worth about a hundredfold of data. Ideas amplify data and compute; you still want all the data and compute you can get.

16.3.1 Many mid-scale ideas, not one big one

Removing individual components from the final system shows each discrete idea contributes. When AlphaFold came out, part of it was equivariant (respecting rotations in 3D), and the field declared equivariance the answer and rushed to research more of it. He was confused: removing the component that provides all of AlphaFold's equivariance hurts only a bit. AlphaFold 2 was about 30 points better than AlphaFold 1 on the accuracy scale they used, and equivariance explained two or three of those.

AGAINST THE GRAIN

It is not about one idea. Headlines focus on transformers, scaling and test-time inference, which matter, but powerful systems are built from many mid-scale ideas that add up to something transformative.

16.3.2 Cross the line that matters to users

They ground the system forward about 1% at a time. What really mattered was crossing the level of accuracy that mattered to an experimental biologist who did not care about machine learning. Getting there takes a lot of work, and when you do, it is transformative.

THE ONE THING

Protein structure prediction is in some ways ahead of the rest of machine learning in blind assessment. Since 1994, every two years, groups predict about a hundred structures that have been solved but not published, so nobody knows the answers. AlphaFold had about a third of the error of any other group. Everyone overfits to benchmarks they can see, and real-world problems are almost always harder than training problems. External, held-out benchmarks are critical, both while developing and for people deciding whether to trust your system.

16.4 Making it available

They released it two ways: open-source code, and about a week later a database of predictions, first 300,000 and later 200 million, essentially every protein from every organism with a sequenced genome.

TRY THIS

The difference between the two releases was striking. After the blind assessment, structure-prediction specialists were convinced the problem was solved. General biologists, the people they wanted to reach, were not: maybe the assessment was easy. When the database appeared, they got curious and looked up proteins they had solved but not published, or a friend had. People asked how DeepMind got access to their unpublished structure. That social proof, and the ease of simply looking something up, built trust through word of mouth. Make the thing easy enough that sceptics can check it against what they already know.

He quotes users: one wrestled for three or four months with a task and got a better answer from an AlphaFold prediction that morning (“I want my time back”); another appreciated AlphaFold most on a protein they could not even produce and purify for a year.

The moment he knew it mattered: a special issue of *Science* on the nuclear pore complex, a structure of several hundred proteins, appeared a few months after release. Three of the four papers used AlphaFold extensively; he counted over a hundred mentions. DeepMind had not collaborated or even known. He calls it the greatest feeling in the world.

16.4.1 Users do surprising things

Two days after the code was released, a researcher, Yoshitaka Moriwaki, tried putting two proteins into AlphaFold with a linker between them, even though it was built for single proteins and DeepMind was still working on a separate system for how proteins come together. He likens it to prompt engineering for proteins. It turned out to be the best protein interaction predictor in the world. A powerful system can have emergent skills that users discover. The field reacted in real time, finding limitations and new possibilities, including protein design.

16.5 How it changes the practice of science

AGAINST THE GRAIN

Some said AlphaFold’s predictions were nice, but now every protein would need solving the classic way to check them. They are right that science is about experiments, and wrong about what for. Science is about forming and testing hypotheses, not about the structure of a particular protein.

His example: a lab at MIT (the captions garble the name) wanted to use a “molecular syringe” protein, which attaches to a cell and injects a protein into it, for targeted delivery of things like gene editors. They had tried over a hundred ways to change what it recognized, without a structure. They ran an AlphaFold prediction, not even a great one, and almost immediately saw that the “legs” at the bottom must be how it attaches to cells. They replaced them with a designed protein to target a new cell type, and showed they could inject fluorescent proteins into chosen cells in a mouse brain.

Others screen thousands of possible interactions to find the likely ones, which led, for example, to discovering a new component of how egg and sperm come together. He estimates AlphaFold made structural biology perhaps 5–10% faster, which matters enormously for the world.

PARTING ADVICE

The talk ends on a vision rather than advice, so this is the closest thing he said.

AI for science should be an amplifier for experimentalists. Scattered experimental observations are science’s equivalent of all the words on the internet; train a general model that learns the rules underneath and it can fill in the rest of the picture. Start where you have data, then find the problems it can be applied to, as happened with protein interactions.

The most exciting open question, he says, is how general this will be: a few narrow places with transformative impact, or very broad systems, including LLMs, in which we find more and more scientific knowledge. He expects it will ultimately be the latter.

And for builders, the lesson running through the talk: do not assume data and compute are the whole story. Ideas are worth a hundredfold, and measure honestly against problems whose answers you cannot see.

Dylan Field: Craft Is the Differentiator



“Dylan Field: Scaling Figma and the Future of Design”

Dylan Field Co-founder and CEO, Figma. Interviewed by a YC partner, followed by audience Q&A. The interviewer is not named in the captions.

Video: <https://www.youtube.com/watch?v=-7Qz7tStfUU>

Figma today has about 1,700 people, hybrid, and eight products, having just doubled its lineup at its Config conference. The conversation covers the long road to Figma, why design matters more now, and advice.

17.1 Getting started

Field and Evan Wallace met at Brown, where Wallace was his teaching assistant. They asked “why now?”: what was changing in the world that they believed in deeply? Two answers: drones and quadcopters, which Field favoured, and WebGL, a way to use a computer’s GPU in the browser (WebGPU is its successor). After a month Wallace was not into drones, so it was WebGL. That split quickly into games or tools; they chose tools, then explored many kinds of tool.

They started talking in December 2011, started in earnest in August 2012, and did not go all in on Figma as it exists today until around June or July 2013, and even then the product kept narrowing.

TRY THIS

Give yourself a downside case you can live with, and give yourself time. It was always meant to be a startup that could scale. His downside case: a few years working with Evan, whom he calls the smartest person he knows (“look up his GitHub”), learning a lot, then going back to school. He saw no real risk. The Thiel Fellowship helped: \$100,000 over two years meant no savings or debt, but more importantly it bought time. Had they judged the idea at six months, Figma would not exist.

17.1.1 Surviving the pivots

What kept them going was that working with Evan was fun and they thought through ideas by building them, which felt like inventing the future every week. One low point: Field was convinced memes would go to the moon and persuaded Wallace to build a meme generator in 2012. He thinks it would have been the best on the market, and the thesis was right. After a week, both were ready to quit, and Field asked himself why he had dropped out of Brown for this. With a co-founder, your lows and their highs can cancel out and you keep each other going.

17.2 First users, and launching too late

Early users came from cold emails and his network from internships at Flipboard, LinkedIn and O'Reilly Media. He looked up designers whose work he respected and asked to buy them coffee. Many replied, which he still finds wild.

THE ONE THING

Designers give great feedback. Not just “your product sucks” but exactly why, what would be better, and what it would take for them to use it. Engaging with that made the product better, and eventually many converted.

After raising venture money, he had investors introduce him to their portfolio companies and spent a whole summer demoing to five to seven companies a week, asking whether they would use it and if not why not. The conversion rate was very low: perhaps two all summer, Notion and the company that became Coda. Both were cloud document tools with philosophies like Figma's. After launch, many more people resonated with the message.

WATCH OUT

“If you take anything away from this, don't do what I did.” Launch as soon as you can, and charge money sooner to see whether you can make money, unless you have some galaxy-brain consumer plan. The feedback clearly said the product was not ready, so they held back. In hindsight, they had the capital and should have grown the team faster to ship sooner.

Today when a Figma team brings him an epic, “perfect” roadmap, his first question is how to slim it down into something bite-sized to test with users earlier. He pushes for one- to three-month cadences at most; a nine-month, twelve-month or two-year plan gets a much blunter response. Constraints breed creativity. The leader's loop is to find what they are doing most of, get someone (or perhaps, in future, AI) to help with it, find the resources to do that, and repeat.

17.2.1 Product–market pull

He believed Figma might be real much later than his users did. People sent twelve-page documents of requests while the product was still bad. What finally convinced him, about five years in, was Microsoft

saying Figma was spreading like wildfire internally, and asking whether to shut it down or continue, partly because Figma was not charging them.

THE ONE THING

Everyone talks about product–market fit; watch for *product–market pull*: people highly engaged, obsessive about what you are doing, seeing the future you are pointing at. Many founders misread that as “if only we had all these requested features we might have fit; we must grind for years.” The right reading is: they care enough to give us this feedback, and that is huge. Double down.

Why was he comfortable seeking harsh feedback? He was a child actor (commercials and some TV, nothing you would know), which meant auditioning constantly and being rejected constantly. He got used to it and enjoyed the process. If you are not there yet, seek rejection: it contains interesting data.

17.3 Why design matters more now

The interviewer lists a big month for design: redesigns at Airbnb and Netflix, Apple’s controversial Liquid Glass interface, Figma’s launches, YC’s call for design founders, and OpenAI acquiring Jony Ive’s company for more than \$6 billion.

THE ONE THING

Design has grown in importance for a decade, moving from lipstick on a pig at the end of the process to deep thinking at every step. With AI, if building software becomes easier and faster, what is your differentiator? Design, craft, attention to detail, point of view. Airbnb has said outright that design is its differentiator.

TRY THIS

On the OpenAI deal, which some call stupid and others genius: some people repeatedly do things you do not understand, and years later you realize your reaction was wrong. After enough of that, flip your stance: assume there is something to learn and that you are missing something. He understands part of the deal and not the size. But Sam Altman is right about a lot, so if you dismissed it outright, ask what you might be missing.

17.3.1 How Figma decides what to build

Figma’s pattern is to notice a behaviour happening inside Figma Design, pull it out into its own product, and let Figma Design stay a product design tool. FigJam was whiteboarding; Slides came from seeing about 5% of Figma Design files were presentations. Adding everything slides need to the design tool would complicate it, and “one plus one is not three, more like 1.5.” Recent launches follow the same pattern: Draw, for deeper vector work; Buzz, for marketing teams mass-producing assets from brand templates; Sites, for publishing websites people were already designing; and Make, which goes from prompt to app and has changed how Figma prototypes, making it faster to throw ideas away.

Asked how he watches these emerging behaviours: a mix of support requests, qualitative interviews and sitting with users, data science, and social media, digested into intuition and testable hypotheses. Art plus science.

17.3.2 Blurring roles

Inside Figma, it is about speed of iteration and cheap prototyping, plus internal tools he cannot discuss yet. Design, development, product and even parts of research are blurring, a trend that predates AI but that AI accelerates, because AI empowers generalist behaviour.

WATCH OUT

Current AI coding tools are much better at the early phases than at large, established codebases. They suit zero-to-one and prototyping more than one-to-a-hundred, for now. That could change in a week.

17.3.3 Interfaces**AGAINST THE GRAIN**

It feels like the MS-DOS era of AI. Ten years from now people will say: can you believe we just had a chat box? But showing users everything these models can do is a very hard, unsolved problem.

Some experiments hint at answers. Midjourney started in Discord, where you see what everyone else is making, which teaches you what is possible. Meta's AI app drew criticism for people accidentally sharing private things publicly, but the flip side, underexplored in the press, is that you learn what you can do. He also expects AI to become contextual inside applications, and surfaces to multiply beyond phones, laptops and tablets to glasses and many kinds of displays. Keeping that consistent and navigable is a big design challenge.

17.4 Design in research**TRY THIS**

Many researchers are trained to approach problems as abstractions. For pure maths, keep going. For applied work like AI, think like a designer and work with designers. Figma embeds designers in its research teams, because researchers need intuition about how designers think. More generally, designers think about an audience with a problem to solve, and qualitative research should pair with deep AI research. If you come from research, go into the field and talk to people; it will make you faster.

THE ONE THING

Designers and product people should contribute to evals. Usually researchers build them, which he thinks is the wrong model, since engineers and researchers typically have less contact with and understanding of the end user than designers and product people do. As models are built, evals matter more.

17.4.1 The designer's role in ten years

He expects designers to have far more leverage and design's value to keep rising. He points to YC's call for design founders, and to Brian Chesky at Airbnb and Karri Saarinen at Linear, and expects the number of designer founders and designer general managers to multiply. Like a company today with one best writer while everyone has a word processor, there will be designers who are the best at problem solving, exploring the idea maze and building a system, while nearly everyone contributes to design. That means lots of curation, and designers will need to step up and lead.

17.5 Advice for people in their twenties

What people have always done to learn to think critically still matters: learn deeply about as many areas as you are curious about so you can make connections, experience the world, and keep relating to people.

WATCH OUT

He has read that people in their twenties are dating less. It would be easy to talk to an AI model all day and feel socially connected. He strongly advises against it, and advises dating. Going further, and speaking about what the future could hold rather than past products, he calls AI boyfriends and girlfriends, if developed and allowed to become a primary mode of relationship, “a societal self-own” and actively poisonous, something society needs a broad discussion about.

The most fun period of building Figma, he says, is right now: brilliant people, a great problem set, and more ideas than ever. The challenge is choosing the right things.

17.6 From the Q&A

Is Cursor a competitor? No. Figma just launched an MCP server so designs flow into Cursor, Windsurf and VS Code faster. If design is the differentiator, your first one-shot generation is probably not what wins; the question is how to make it good. Some people iterate structurally in code and a browser, others think more freely and visually, which is what Make is for. Figma announced that day it had acquired Payload, an open-source CMS, and wants to support open source more.

The meaning of life? Explore consciousness, learn as much as you can, share love, and make sure you and those around you are fulfilled and happy, whether locally or at scale. If you live true to your values, you are leading a fulfilling life.

A design principle people ignore. Not his own, and decades old: keep the simple things simple and make the complex things possible. If enabling everything makes the product unapproachable, you are messing up. Start with simple things simple.

Pitching a founder you admire as an angel. Send a Loom by email so they can watch when they have time. Mutual connections help, but cold emails work too.

Narrow or broad at the start? Narrow: product design for digital products, and more narrowly, teams that cared about design, because it was an easier sale. He started out wanting to do everything, and the team pushed back. He is glad they did.

Ethics of AI in creative tools. There are many challenges, from the energy used by inference to models regurgitating what they have seen. Much of Figma’s current work uses third-party models, where it has less control; as it does more in-house, these questions become very relevant, as they are for the art world.

PARTING ADVICE

Asked whether AI interaction should be designed as a tool or something more human, given people anthropomorphize it:

He sees a split between a materialist worldview, where consciousness arises from matter, and religious views that say AI obviously has no soul. Whatever you believe, he predicts more and more people will project consciousness onto AI, which leads into very hard territory. What that means for human–computer interaction is badly underexplored, and he told the researcher asking that he is excited to see what they do with it.

The advice running through the whole conversation: launch earlier than feels comfortable, treat demanding feedback as a sign of pull, and remember that when building becomes easy, craft and point of view are what set you apart.