00:00:00 – AGI is still a decade away

Dwarkesh Patel *00:00:00*

Today I'm speaking with Andrej Karpathy. Andrej, why do you say that this will be the decade of agents and not the year of agents?
오늘은 Andrej Karpathy와 대화를 나누고 있습니다. Andrej, 왜 올해가 에이전트의 해가 아니라 이것이 에이전트의 10년이 될 거라고 생각하세요?

Andrej Karpathy *00:00:07*

First of all, thank you for having me here. I'm excited to be here.
먼저 저를 초대해 주셔서 감사합니다. 여기 올 수 있어서 정말 기쁩니다.

The quote you've just mentioned, "It's the decade of agents," is actually a reaction to a pre-existing quote. I'm not actually sure who said this but they were alluding to this being the year of agents with respect to LLMs and how they were going to evolve. I was triggered by that because there's some over-prediction going on in the industry. In my mind, this is more accurately described as the decade of agents.
제가 언급한 "에이전트의 10년"이라는 표현은 실은 기존의 다른 주장에 대한 반응입니다. 정확히 누가 처음 말했는지는 모르겠지만, 누군가는 LLM과 그 발전과 관련해서 올해가 '에이전트의 해'가 될 거라고 말했습니다. 이것 때문에 저는 반박하고 싶었어요. 왜냐하면 업계에서 과도한 예측이 많이 있기 때문입니다. 제 생각엔 '에이전트의 10년'이 훨씬 더 정확한 표현입니다.

We have some very early agents that are extremely impressive and that I use daily—Claude and Codex and so on—but I still feel there's so much work to be done. My reaction is we'll be working with these things for a decade. They're going to get better, and it's going to be wonderful. I was just reacting to the timelines of the implication.
우리는 이미 정말 인상적인 초기 에이전트들을 가지고 있어요. Claude와 Codex 같은 것들을 매일 사용하고 있습니다. 하지만 여전히 해야 할 일이 정말 많다고 느낍니다. 제 생각은 우리가 이런 도구들과 10년 동안 함께 일하게 될 거라는 거예요. 계속해서 더 나아질 것이고, 정말 멋질 거고요. 저는 단지 이런 의미에서 타임라인에 반응한 것입니다.

Dwarkesh Patel *00:00:58*

What do you think will take a decade to accomplish? What are the bottlenecks?
그럼 무엇을 이루는 데 10년이 걸릴 것 같습니까? 어떤 병목이 있을까요?

Andrej Karpathy *00:01:02*

Actually making it work. When you're talking about an agent, or what the labs have in mind and maybe what I have in mind as well, you should think of it almost like an employee or an intern that you would hire to work with you. For example, you work with some employees here. When would you prefer to have an agent like Claude or Codex do that work?
실제로 그것이 작동하게 만드는 일입니다. 에이전트에 대해 말할 때, 또는 연구실들이 생각하는 것, 그리고 제가 생각하는 것을 말할 때, 당신이 함께 일하기 위해 고용할 직원이나 인턴처럼 생각하면 됩니다. 예를 들어, 여기서 일하는 직원들이 있으시잖아요. 언제 Claude나 Codex 같은 에이전트가 그 일을 하길 원할까요?

Currently, of course they can't. What would it take for them to be able to do that? Why don't you do it today? The reason you don't do it today is because they just don't work. They don't have enough intelligence, they're not multimodal enough, they can't do computer use and all this stuff.
현재로서는 물론 불가능합니다. 그들이 그것을 할 수 있게 하려면 무엇이 필요할까요? 왜 오늘 그것을 하지 않을까요? 오늘 하지 않는 이유는 그들이 그냥 작동하지 않기 때문입니다. 지능이 충분하지 않고, 멀티모달이 충분하지 않으며, 컴퓨터 사용을 할 수 없고, 모든 이런 것들을 할 수 없습니다.

They don't do a lot of the things you've alluded to earlier. They don't have continual learning. You can't just tell them something and they'll remember it. They're cognitively lacking and it's just not working. It will take about a decade to work through all of those issues.
그들은 당신이 앞에서 언급한 많은 것들을 하지 못합니다. 지속적인 학습이 없어요. 당신이 그들에게 뭔가를 말하면 그들이 그것을 기억하지 못합니다. 그들은 인지적으로 부족하고 지금 작동하지 않습니다. 이 모든 문제들을 해결하는 데 약 10년이 걸릴 거예요.

Dwarkesh Patel *00:01:44*

Interesting. As a professional podcaster and a viewer of AI from afar, it's easy for me to identify what's lacking: continual learning is lacking, or multimodality is lacking. But I don't really have a good way of trying to put a timeline on it. If somebody asks how long continual learning will take, I have no prior about whether this is a project that should take 5 years, 10 years, or 50 years. Why a decade? Why not one year? Why not 50 years?
흥미롭네요. 전문 팟캐스터이면서 먼 곳에서 AI를 관찰하는 입장이라, 무엇이 부족한지 파악하기는 쉬워요. 지속적인 학습이 부족하거나 멀티모달 기능이 부족하다는 것 같은 거죠. 하지만 저는 이것을 타임라인에 놓을 좋은 방법을 가지고 있지 않습니다. 누군가 지속적인 학습에 얼마나 걸릴지 묻는다면, 저는 그것이 5년, 10년, 또는 50년이 걸려야 하는 프로젝트인지 알 수 없어요. 왜 10년일까요? 왜 1년이 아니라? 왜 50년이 아니라?

Andrej Karpathy *00:02:16*

This is where you get into a bit of my own intuition, and doing a bit of an extrapolation with respect to my own experience in the field. I've been in AI for almost two decades. It's going to be 15 years or so, not that long. You had Richard Sutton here, who was around for much longer. I do have about 15 years of experience of people making predictions, of seeing how they turned out. Also I was in the industry for a while, I was in research, and I've worked in the industry for a while. I have a general intuition that I have left from that.
여기서는 제 개인적인 직관이 좀 들어가고, 제 경험을 바탕으로 외삽법을 사용하는 거라고 할 수 있습니다. 저는 거의 20년을 AI 분야에서 보냈어요. 15년 정도가 될 거 같긴 하지만 그렇게 오래되지는 않았습니다. 여기 Richard Sutton이 있었는데, 그는 훨씬 더 오래 있었어요. 저는 약 15년의 경험을 가지고 있습니다. 사람들이 예측을 하고 그것이 어떻게 나왔는지를 보는 경험 말이에요. 또한 저는 한동안 업계에 있었고, 연구도 했고, 한동안 산업계에서도 일했습니다. 그 모든 것에서 나온 일반적인 직관을 가지고 있어요.

I feel like the problems are tractable, they're surmountable, but they're still difficult. If I just average it out, it just feels like a decade to me.
제 생각엔 이 문제들은 다루기 쉽고, 극복할 수 있지만, 여전히 어렵다는 거예요. 만약 제가 평균을 내본다면, 제게는 10년이 적절해 보입니다.

Dwarkesh Patel *00:02:57*

This is quite interesting. I want to hear not only the history, but what people in the room felt was about to happen at various different breakthrough moments. What were the ways in which their feelings were either overly pessimistic or overly optimistic? Should we just go through each of them one by one?
이것은 정말 흥미롭습니다. 저는 역사뿐만 아니라 사람들이 다양한 돌파구 순간들에서 무엇을 느꼈는지도 알고 싶어요. 그들의 감정이 지나치게 비관적이었거나 지나치게 낙관적이었던 방식들은 어떤 것들이 있을까요? 우리가 하나씩 하나씩 살펴봐야 할까요?

Andrej Karpathy *00:03:16*

That's a giant question because you're talking about 15 years of stuff that happened. AI is so wonderful because there have been a number of seismic shifts where the entire field has suddenly looked a different way. I've maybe lived through two or three of those. I still think there will continue to be some because they come with almost surprising regularity.
15년 동안 일어난 일을 이야기하는 거대한 질문입니다. AI는 정말 멋진데, 전체 분야가 갑자기 다르게 보이는 여러 지각변동이 있었기 때문입니다. 저는 아마 두세 개를 경험했을 겁니다. 거의 놀라운 규칙성으로 오기 때문에 앞으로도 계속 있을 거라고 생각합니다.

When my career began, when I started to work on deep learning, when I became interested in deep learning, this was by chance of being right next to Geoff Hinton at the University of Toronto. Geoff Hinton, of course, is the godfather figure of AI. He was training all these neural networks. I thought it was incredible and interesting.

This was not the main thing that everyone in AI was doing by far. This was a niche little subject on the side. That's maybe the first dramatic seismic shift that came with the AlexNet and so on.
제 경력이 시작되었을 때, 딥러닝 작업을 시작했을 때, 딥러닝에 관심을 갖게 되었을 때, 토론토 대학교에서 제프 힌튼 바로 옆에 있었던 우연 덕분이었습니다. 제프 힌튼은 물론 AI의 대부입니다. 그는 이 모든 신경망을 훈련시키고 있었습니다. 저는 그것이 놀랍고 흥미롭다고 생각했습니다. 이것은 AI의 모든 사람이 하는 주요한 일이 전혀 아니었습니다. 옆에 있는 작은 틈새 주제였습니다. 그것이 AlexNet 등과 함께 온 첫 번째 극적인 지각변동이었을 겁니다.

AlexNet reoriented everyone, and everyone started to train neural networks, but it was still very per-task, per specific task. Maybe I have an image classifier or I have a neural machine translator or something like that. People became very slowly interested in agents. People started to think, "Okay, maybe we have a check mark next to the visual cortex or something like that, but what about the other parts of the brain, and how can we get a full agent or a full entity that can interact in the world?"
AlexNet은 모든 사람을 재정향시켰고, 모든 사람이 신경망을 훈련시키기 시작했지만, 여전히 작업별, 특정 작업별이었습니다. 이미지 분류기나 신경 기계 번역기 같은 것들이요. 사람들은 매우 천천히 에이전트에 관심을 갖게 되었습니다. "좋아, 우리가 시각 피질 같은 것 옆에 체크 표시를 했을지 모르지만, 뇌의 다른 부분은 어떻고, 세계와 상호작용할 수 있는 완전한 에이전트나 완전한 개체를 어떻게 얻을 수 있을까?"라고 생각하기 시작했습니다.

The Atari deep reinforcement learning shift in 2013 or so was part of that early effort of agents, in my mind, because it was an attempt to try to get agents that not just perceive the world, but also take actions and interact and get rewards from environments. At the time, this was Atari games.
2013년경 아타리 심층 강화 학습 전환은 제 마음속에서 에이전트의 초기 노력의 일부였습니다. 왜냐하면 세계를 인식할 뿐만 아니라 행동을 취하고 상호작용하며 환경으로부터 보상을 받는 에이전트를 얻으려는 시도였기 때문입니다. 당시 이것은 아타리 게임이었습니다.

I feel that was a misstep. It was a misstep that even the early OpenAI that I was a part of adopted because at that time, the zeitgeist was reinforcement learning environments, games, game playing, beat games, get lots of different types of games, and OpenAI was doing a lot of that. That was another prominent part of AI where maybe for two or three or four years, everyone was doing reinforcement learning on games. That was all a bit of a misstep.
그것이 실수였다고 느낍니다. 제가 일부였던 초기 OpenAI도 채택한 실수였습니다. 그 당시 시대정신은 강화 학습 환경, 게임, 게임 플레이, 게임을 이기는 것, 다양한 유형의 게임을 얻는 것이었고, OpenAI는 그것을 많이 하고 있었습니다. 그것은 AI의 또 다른 두드러진 부분이었는데, 2-4년 동안 모든 사람이 게임에서 강화 학습을 하고 있었습니다. 그것은 모두 약간의 실수였습니다.

What I was trying to do at OpenAI is I was always a bit suspicious of games as being this thing that would lead to AGI. Because in my mind, you want something like an accountant or something that's interacting with the real world. I just didn't see how games add up to it. My project at OpenAI, for example, was within the scope of

the Universe project, on an agent that was using keyboard and mouse to operate web pages. I really wanted to have something that interacts with the actual digital world that can do knowledge work.
OpenAI에서 제가 하려고 했던 것은, 저는 항상 게임이 AGI로 이어질 것이라는 점에 약간 의심스러웠습니다. 제 마음속에서는 회계사나 실제 세계와 상호작용하는 무언가를 원합니다. 게임이 어떻게 그것에 더해지는지 보지 못했습니다. 예를 들어 OpenAI에서 제 프로젝트는 유니버스 프로젝트의 범위 내에서 키보드와 마우스를 사용하여 웹 페이지를 조작하는 에이전트였습니다. 실제 디지털 세계와 상호작용하고 지식 작업을 할 수 있는 무언가를 정말 원했습니다.

It just so turns out that this was extremely early, way too early, so early that we shouldn't have been working on that. Because if you're just stumbling your way around and keyboard mashing and mouse clicking and trying to get rewards in these environments, your reward is too sparse and you just won't learn. You're going to burn a forest computing, and you're never going to get something off the ground. What you're missing is this power of representation in the neural network.
이것이 극도로 이른, 너무 이른, 우리가 작업하지 말았어야 할 정도로 이른 시기였다는 것이 밝혀졌습니다. 왜냐하면 그냥 더듬거리며 키보드를 두드리고 마우스를 클릭하며 이러한 환경에서 보상을 얻으려고 한다면, 보상이 너무 희박하고 배우지 못할 것이기 때문입니다. 숲을 태울 정도로 계산하고도 아무것도 얻지 못할 겁니다. 놓치고 있는 것은 신경망에서의 표현의 힘입니다.

For example, today people are training those computer-using agents, but they're doing it on top of a large language model. You have to get the language model first, you have to get the representations first, and you have to do that by all the pre-training and all the LLM stuff.
예를 들어, 오늘날 사람들은 컴퓨터 사용 에이전트를 훈련시키고 있지만, 대규모 언어 모델 위에서 하고 있습니다. 먼저 언어 모델을 얻어야 하고, 먼저 표현을 얻어야 하며, 모든 사전 훈련과 모든 LLM 작업으로 그것을 해야 합니다.

I feel maybe loosely speaking, people kept trying to get the full thing too early a few times, where people really try to go after agents too early, I would say. That was Atari and Universe and even my own experience. You actually have to do some things first before you get to those agents. Now the agents are a lot more competent, but maybe we're still missing some parts of that stack.
느슨하게 말해서, 사람들은 몇 번 너무 일찍 전체를 얻으려고 계속 시도했다고 느낍니다. 사람들이 너무 일찍 에이전트를 추구하려고 정말 시도했다고 말하고 싶습니다. 그것이 아타리와 유니버스, 그리고 제 자신의 경험이었습니다. 실제로 그러한 에이전트에 도달하기 전에 먼저 몇 가지를 해야 합니다. 이제 에이전트는 훨씬 더 유능하지만, 우리는 여전히 그 스택의 일부를 놓치고 있을 수 있습니다.

I would say those are the three major buckets of what people were doing: training neural nets per-tasks, trying the first round of agents, and then maybe the LLMs and seeking the representation power of the neural networks before you tack on everything else on top.
사람들이 하고 있던 세 가지 주요 버킷이라고 말하고 싶습니다: 작업별 신경망 훈련, 에이전트의 첫 번째 라운드 시도, 그리고 위에 다른 모든 것을 붙이기 전에 신경망의 표현력을 추구하는 LLM입니다.

Dwarkesh Patel *00:07:02*

Interesting. If I were to steelman the Sutton perspective, it would be that humans can just take on everything at once, or even animals can take on everything at once. Animals are maybe a better example because they don't even have the scaffold of language. They just get thrown out into the world, and they just have to make sense of everything without any labels.
흥미롭네요. 서튼의 관점을 강화하자면, 인간은 한 번에 모든 것을 받아들일 수 있고, 동물도 한 번에 모든 것을 받아들일 수 있습니다. 동물이 더 나은 예일 수 있는데, 언어의 발판조차 없기 때문입니다. 그들은 그냥 세상에 던져지고, 어떤 라벨도 없이 모든 것을 이해해야 합니다.

The vision for AGI then should just be something which looks at sensory data, looks at the computer screen, and it just figures out what's going on from scratch. If a human were put in a similar situation and had to be trained from scratch… This is like a human growing up or an animal growing up. Why shouldn't that be the vision for AI, rather than this thing where we're doing millions of years of training?
그렇다면 AGI의 비전은 감각 데이터를 보고, 컴퓨터 화면을 보고, 처음부터 무슨 일이 일어나는지 알아내는 것이어야 합니다. 인간이 비슷한 상황에 놓여 처음부터 훈련받아야 한다면... 이것은 인간이나 동물이 자라는 것과 같습니다. 왜 이것이 수백만 년의 훈련을 하는 것보다 AI의 비전이 되어서는 안 될까요?

Andrej Karpathy *00:07:41*

That's a really good question. Sutton was on your podcast and I saw the podcast and I had a write-up about that podcast that gets into a bit of how I see things. I'm very careful to make analogies to animals because they came about by a very different optimization process. Animals are evolved, and they come with a huge amount of hardware that's built in.
정말 좋은 질문입니다. 서튼이 당신의 팟캐스트에 나왔고 저는 그 팟캐스트를 봤고 제가 어떻게 보는지에 대한 글을 썼습니다. 저는 동물과 유추하는 데 매우 조심스럽습니다. 왜냐하면 그들은 매우 다른 최적화 과정으로 생겨났기 때문입니다. 동물은 진화했고, 엄청난 양의 내장된 하드웨어를 가지고 옵니다.

For example, my example in the post was the zebra. A zebra gets born, and a few minutes later it's running around and following its mother. That's an extremely complicated thing to do. That's not reinforcement learning. That's something that's baked in. Evolution obviously has some way of encoding the weights of our neural nets in ATCGs, and I have no idea how that works, but it apparently works.
예를 들어, 제 포스트의 예는 얼룩말이었습니다. 얼룩말이 태어나면 몇 분 후에 뛰어다니며 어미를 따라갑니다. 그것은 매우 복잡한 일입니다. 그것은 강화 학습이 아닙니다. 그것은 내장된 무언가입니다.

진화는 분명히 우리 신경망의 가중치를 ATCG로 인코딩하는 방법을 가지고 있고, 저는 그것이 어떻게 작동하는지 전혀 모르지만, 분명히 작동합니다.

Brains just came from a very different process, and I'm very hesitant to take inspiration from it because we're not actually running that process. In my post, I said we're not building animals. We're building ghosts or spirits or whatever people want to call it, because we're not doing training by evolution. We're doing training by imitation of humans and the data that they've put on the Internet.
뇌는 매우 다른 과정에서 왔고, 우리가 실제로 그 과정을 실행하지 않기 때문에 그것에서 영감을 얻는 것을 매우 주저합니다. 제 포스트에서 말했듯이, 우리는 동물을 만들지 않습니다. 우리는 유령이나 영혼, 사람들이 부르고 싶은 무엇이든 만들고 있습니다. 왜냐하면 우리는 진화로 훈련하지 않기 때문입니다. 우리는 인간과 그들이 인터넷에 올린 데이터를 모방함으로써 훈련하고 있습니다.

You end up with these ethereal spirit entities because they're fully digital and they're mimicking humans. It's a different kind of intelligence. If you imagine a space of intelligences, we're starting off at a different point almost. We're not really building animals. But it's also possible to make them a bit more animal-like over time, and I think we should be doing that.
이러한 천상의 영혼 개체로 끝나게 됩니다. 왜냐하면 그들은 완전히 디지털이고 인간을 모방하기 때문입니다. 다른 종류의 지능입니다. 지능의 공간을 상상한다면, 우리는 거의 다른 지점에서 시작하고 있습니다. 우리는 정말로 동물을 만들지 않습니다. 하지만 시간이 지남에 따라 그들을 좀 더 동물처럼 만드는 것도 가능하고, 그렇게 해야 한다고 생각합니다.

One more point. I do feel Sutton has a very... His framework is, "We want to build animals." I think that would be wonderful if we can get that to work. That would be amazing. If there were a single algorithm that you can just run on the Internet and it learns everything, that would be incredible. I'm not sure that it exists and that's certainly not what animals do, because animals have this outer loop of evolution.
한 가지 더. 서튼은 매우... 그의 프레임워크는 "우리는 동물을 만들고 싶다"입니다. 그것을 작동시킬 수 있다면 멋질 것입니다. 놀라울 거예요. 인터넷에서 실행할 수 있고 모든 것을 배우는 단일 알고리즘이 있다면 놀라울 것입니다. 그것이 존재하는지 확실하지 않고, 동물이 하는 것이 확실히 아닙니다. 왜냐하면 동물은 진화의 외부 루프를 가지고 있기 때문입니다.

A lot of what looks like learning is more like maturation of the brain. I think there's very little reinforcement learning for animals. A lot of the reinforcement learning is more like motor tasks; it's not intelligence tasks. So I actually kind of think humans don't really use RL, roughly speaking.
학습처럼 보이는 많은 것이 뇌의 성숙에 더 가깝습니다. 동물에게는 강화 학습이 거의 없다고 생각합니다. 많은 강화 학습은 운동 작업에 더 가깝고, 지능 작업이 아닙니다. 그래서 저는 실제로 인간이 RL을 사용하지 않는다고 생각합니다, 대략적으로 말해서.

Dwarkesh Patel *00:09:52*

Can you repeat the last sentence? A lot of that intelligence is not motor task…it's what, sorry?
마지막 문장을 반복해 주실 수 있나요? 많은 지능이 운동 작업이 아니라... 뭐라고 하셨죠?

Andrej Karpathy *00:09:54*

A lot of the reinforcement learning, in my perspective, would be things that are a lot more motor-like, simple tasks like throwing a hoop. But I don't think that humans use reinforcement learning for a lot of intelligence tasks like problem-solving and so on. That doesn't mean we shouldn't do that for research, but I just feel like that's what animals do or don't.
제 관점에서 강화 학습의 많은 부분은 농구 골대에 공 던지기 같은 운동적인 간단한 작업일 것입니다. 하지만 인간이 문제 해결 같은 많은 지능 작업에 강화 학습을 사용한다고 생각하지 않습니다. 그것이 우리가 연구에서 해서는 안 된다는 의미는 아니지만, 동물이 하거나 하지 않는 것이라고 느낍니다.

Dwarkesh Patel *00:10:17*

I'm going to take a second to digest that because there are a lot of different ideas. Here's one clarifying question I can ask to understand the perspective. You suggest that evolution is doing the kind of thing that pre-training does in the sense of building something which can then understand the world.
소화하는 데 시간이 필요한데, 많은 다른 아이디어들이 있기 때문입니다. 여기 관점을 이해하기 위해 물을 수 있는 명확한 질문이 있습니다. 진화가 세계를 이해할 수 있는 무언가를 구축한다는 점에서 사전 훈련이 하는 것과 같은 종류의 일을 한다고 제안하시는 건가요?

The difference is that evolution has to be titrated in the case of humans through three gigabytes of DNA. That's very unlike the weights of a model. Literally, the weights of the model are a brain, which obviously does not exist in the sperm and the egg. So it has to be grown. Also, the information for every single synapse in the brain simply cannot exist in the three gigabytes that exist in the DNA.
차이점은 인간의 경우 진화가 DNA의 3기가바이트를 통해 적정되어야 한다는 것입니다. 그것은 모델의 가중치와 매우 다릅니다. 문자 그대로 모델의 가중치는 뇌인데, 정자와 난자에는 존재하지 않습니다. 그래서 성장해야 합니다. 또한 뇌의 모든 단일 시냅스에 대한 정보는 DNA의 3기가바이트에 존재할 수 없습니다.

Evolution seems closer to finding the algorithm which then does the lifetime learning. Now, maybe the lifetime learning is not analogous to RL, to your point. Is that compatible with the thing you were saying, or would you disagree with that?
진화는 평생 학습을 하는 알고리즘을 찾는 것에 더 가까운 것 같습니다. 이제 평생 학습이 RL과 유사하지 않을 수 있습니다. 당신이 말한 것과 호환되나요, 아니면 동의하지 않으시나요?

Andrej Karpathy *00:11:17*

I think so. I would agree with you that there's some miraculous compression going on because obviously, the weights of the neural net are not stored in ATCGs. There's some dramatic compression. There are some learning algorithms encoded that take over and do some of the learning online. I definitely agree with you on that. I would say I'm a lot more practically minded. I don't come at it from the perspective of, let's build animals. I come from it from the perspective of, let's build useful things. I have a hard hat on, and I'm just observing that we're not going to do evolution, because I don't know how to do that.
그렇게 생각합니다. 분명히 가중치가 ATCG에 저장되지 않기 때문에 기적적인 압축이 일어나고 있다는 데 동의합니다. 극적인 압축이 있습니다. 인코딩된 학습 알고리즘이 있어서 온라인에서 학습의 일부를 수행합니다. 그 점에 확실히 동의합니다. 저는 훨씬 더 실용적인 마음을 가지고 있다고 말하고 싶습니다. 동물을 만들자는 관점에서 오지 않습니다. 유용한 것을 만들자는 관점에서 옵니다. 저는 안전모를 쓰고 있고, 우리가 진화를 하지 않을 것이라는 것을 관찰하고 있습니다. 왜냐하면 어떻게 해야 할지 모르기 때문입니다.

But it does turn out we can build these ghosts, spirit-like entities, by imitating internet documents. This works. It's a way to bring you up to something that has a lot of built-in knowledge and intelligence in some way, similar to maybe what evolution has done. That's why I call pre-training this crappy evolution. It's the practically possible version with our technology and what we have available to us to get to a starting point where we can do things like reinforcement learning and so on.
하지만 인터넷 문서를 모방함으로써 이러한 유령, 영혼 같은 개체를 만들 수 있다는 것이 밝혀졌습니다. 이것은 작동합니다. 진화가 한 것과 유사하게 어떤 식으로든 많은 내장된 지식과 지능을 가진 무언가로 이끄는 방법입니다. 그래서 저는 사전 훈련을 이 형편없는 진화라고 부릅니다. 우리의 기술과 우리가 사용할 수 있는 것으로 실제로 가능한 버전이며, 강화 학습 등을 할 수 있는 시작점에 도달하는 것입니다.

Dwarkesh Patel *00:12:15*

Just to steelman the other perspective, after doing this Sutton interview and thinking about it a bit, he has an important point here. Evolution does not give us the knowledge, really. It gives us the algorithm to find the knowledge, and that seems different from pre-training.
다른 관점을 강화하기 위해, 이 서튼 인터뷰를 하고 그것에 대해 생각한 후, 그는 여기서 중요한 포인트를 가지고 있습니다. 진화는 우리에게 지식을 주지 않습니다. 지식을 찾는 알고리즘을 줍니다. 그것은 사전 훈련과 다른 것 같습니다.

Perhaps the perspective is that pre-training helps build the kind of entity which can learn better. It teaches meta-learning, and therefore it is similar to finding an algorithm. But if it's "Evolution gives us knowledge, pre-training gives us knowledge," that analogy seems to break down.
아마도 관점은 사전 훈련이 더 잘 배울 수 있는 종류의 개체를 구축하는 데 도움이 된다는 것입니다. 메타 학습을 가르치므로 알고리즘을 찾는 것과 유사합니다. 하지만 "진화는 우리에게 지식을 주고, 사전 훈련은 우리에게 지식을 준다"면 그 유추는 무너지는 것 같습니다.

Andrej Karpathy *00:12:42*

It's subtle and I think you're right to push back on it, but basically the thing that pre-training is doing, you're getting the next-token predictor over the internet, and you're training that into a neural net. It's doing two things that are unrelated. Number one, it's picking up all this knowledge, as I call it. Number two, it's actually becoming intelligent.
미묘하고 반박하는 것이 옳다고 생각하지만, 기본적으로 사전 훈련이 하는 것은 인터넷을 통해 다음 토큰 예측기를 얻고 신경망에 훈련시키는 것입니다. 관련 없는 두 가지를 하고 있습니다. 첫째, 제가 부르는 모든 지식을 습득하고 있습니다. 둘째, 실제로 지능적이 되고 있습니다.

By observing the algorithmic patterns in the internet, it boots up all these little circuits and algorithms inside the neural net to do things like in-context learning and all this stuff. You don't need or want the knowledge. I think that's probably holding back the neural networks overall because it's getting them to rely on the knowledge a little too much sometimes.
인터넷의 알고리즘 패턴을 관찰함으로써 신경망 내부에 문맥 내 학습과 같은 일을 하는 모든 작은 회로와 알고리즘을 부팅합니다. 지식이 필요하거나 원하지 않습니다. 그것이 아마도 전체적으로 신경망을 방해하고 있다고 생각합니다. 왜냐하면 때때로 지식에 너무 많이 의존하게 만들기 때문입니다.

For example, I feel agents, one thing they're not very good at, is going off the data manifold of what exists on the internet. If they had less knowledge or less memory, maybe they would be better. What I think we have to do going forward—and this would be part of the research paradigms—is figure out ways to remove some of the knowledge and to keep what I call this cognitive core. It's this intelligent entity that is stripped from knowledge but contains the algorithms and contains the magic of intelligence and problem-solving and the strategies of it and all this stuff.
예를 들어, 에이전트가 잘하지 못하는 한 가지는 인터넷에 존재하는 데이터 매니폴드를 벗어나는 것입니다. 지식이나 메모리가 적다면 더 나을 수도 있습니다. 앞으로 우리가 해야 할 일 - 그리고 이것은 연구 패러다임의 일부일 것입니다 - 은 일부 지식을 제거하고 제가 인지적 핵심이라고 부르는 것을 유지하는 방법을 찾는 것입니다. 지식에서 벗어났지만 알고리즘을 포함하고 지능과 문제 해결의 마법과 전략 등을 포함하는 이 지능적인 개체입니다.

Dwarkesh Patel *00:13:50*

There's so much interesting stuff there. Let's start with in-context learning. This is an obvious point, but I think it's worth just saying it explicitly and meditating on it. The situation in which these models seem the most intelligent—in which I talk to them and I'm like, "Wow, there's really something on the other end that's responding to me thinking about things—is if it makes a mistake it's like, "Oh wait, that's the wrong way to think about it. I'm backing up." All that is happening in context. That's where I feel like the real intelligence is that you can visibly see.

정말 흥미로운 내용이 많네요. 문맥 내 학습부터 시작해 봅시다. 이것은 명백한 점이지만 명시적으로 말하고 명상할 가치가 있다고 생각합니다. 이 모델들이 가장 지능적으로 보이는 상황 - 제가 그들과 대화하면서 "와, 정말로 뭔가가 반대편에서 나에게 반응하고 생각하고 있구나"라고 느끼는 상황 - 은 실수를 하면 "아, 잠깐, 그건 잘못된 생각이야. 뒤로 물러나고 있어"라고 하는 것입니다. 그 모든 것이 문맥 내에서 일어나고 있습니다. 거기서 볼 수 있는 진짜 지능이 있다고 느낍니다.

That in-context learning process is developed by gradient descent on pre-training. It spontaneously meta-learns in-context learning, but the in-context learning itself is not gradient descent, in the same way that our lifetime intelligence as humans to be able to do things is conditioned by evolution but our learning during our lifetime is happening through some other process.
그 문맥 내 학습 과정은 사전 훈련에서 경사 하강법으로 개발됩니다. 자발적으로 문맥 내 학습을 메타 학습하지만, 문맥 내 학습 자체는 경사 하강법이 아닙니다. 우리의 평생 지능이 진화에 의해 조건지어지지만 우리의 평생 학습이 다른 과정을 통해 일어나는 것과 같은 방식입니다.

Andrej Karpathy *00:14:42*

I don’t fully agree with that, but you should continue your thought.
완전히 동의하지는 않지만, 계속 생각을 말씀해 주세요.

Dwarkesh Patel *00:14:44*

Well, I’m very curious to understand how that analogy breaks down.
그 유추가 어떻게 무너지는지 이해하고 싶습니다.

Andrej Karpathy *00:14:48*

I’m hesitant to say that in-context learning is not doing gradient descent. It’s not doing explicit gradient descent. In-context learning is pattern completion within a token window. It just turns out that there’s a huge amount of patterns on the internet. You’re right, the model learns to complete the pattern, and that’s inside the weights. The weights of the neural network are trying to discover patterns and complete the pattern. There’s some adaptation that happens inside the neural network, which is magical and just falls out from the internet just because there’s a lot of patterns.
문맥 내 학습이 경사 하강법을 하지 않는다고 말하는 것을 주저합니다. 명시적인 경사 하강법을 하지 않습니다. 문맥 내 학습은 토큰 창 내에서 패턴 완성입니다. 인터넷에는 엄청난 양의 패턴이 있습니다. 맞습니다, 모델은 패턴을 완성하는 법을 배우고, 그것은 가중치 내부에 있습니다. 신경망의 가중치는 패턴을 발견하고 패턴을 완성하려고 합니다. 신경망 내부에서 일어나는 적응이 있는데, 이것은 마법적이고 인터넷에서 단지 많은 패턴이 있기 때문에 떨어져 나옵니다.

I will say that there have been some papers that I thought were interesting that look at the mechanisms behind in-context learning. I do think it's possible that in-context learning runs a small gradient descent loop internally in the layers of the neural network. I recall one paper in particular where they were doing linear regression using in-context learning. Your inputs into the neural network are XY pairs, XY, XY, XY that happen to be on the line. Then you do X and you expect Y. The neural network, when you train it in this way, does linear regression.
문맥 내 학습의 메커니즘을 살펴본 흥미로운 논문들이 있다고 생각합니다. 문맥 내 학습이 신경망 레이어에서 내부적으로 작은 경사 하강 루프를 실행할 수 있다고 생각합니다. 특히 기억나는 논문 중 하나는 문맥 내 학습을 사용하여 선형 회귀를 수행하는 것이었습니다. 신경망에 대한 입력은 XY 쌍, XY, XY, XY로 선 위에 있는 것들입니다. 그런 다음 X를 하면 Y를 기대합니다. 이런 방식으로 훈련하면 신경망은 선형 회귀를 수행합니다.

Normally when you would run linear regression, you have a small gradient descent optimizer that looks at XY, looks at an error, calculates the gradient of the weights and does the update a few times. It just turns out that when they looked at the weights of that in-context learning algorithm, they found some analogies to gradient descent mechanics. In fact, I think the paper was even stronger because they hardcoded the weights of a neural network to do gradient descent through attention and all the internals of the neural network.
보통 선형 회귀를 실행할 때는 XY를 보고 오류를 보고 가중치의 경사를 계산하고 몇 번 업데이트를 수행하는 작은 경사 하강 최적화 프로그램이 있습니다. 그들이 그 문맥 내 학습 알고리즘의 가중치를 살펴봤을 때 경사 하강 메커니즘과 유사점을 발견했습니다. 사실 그 논문은 더 강력했다고 생각합니다. 왜냐하면 그들은 주의와 신경망의 모든 내부를 통해 경사 하강을 수행하도록 신경망의 가중치를 하드코딩했기 때문입니다.

That's just my only pushback. Who knows how in-context learning works, but I think that it's probably doing a bit of some funky gradient descent internally. I think that that's possible. I was only pushing back on your saying that it's not doing in-context learning. Who knows what it's doing, but it's probably maybe doing something similar to it, but we don't know.
그것이 제 유일한 반박입니다. 문맥 내 학습이 어떻게 작동하는지 누가 알겠습니까만, 아마도 내부적으로 약간 이상한 경사 하강을 하고 있을 것입니다. 그것이 가능하다고 생각합니다. 문맥 내 학습을 하지 않는다고 말하는 것에 반박했을 뿐입니다. 무엇을 하는지 누가 알겠습니까만, 아마도 비슷한 것을 하고 있을 것입니다. 하지만 우리는 모릅니다.

Dwarkesh Patel *00:16:39*

So then it's worth thinking okay, if in-context learning and pre-training are both implementing something like gradient descent, why does it feel like with in-context learning we're getting to this continual learning, real intelligence-like thing? Whereas you don't get the analogous feeling just from pre-training. You could argue that.
그렇다면 문맥 내 학습과 사전 훈련이 모두 경사 하강법과 같은 것을 구현한다면, 왜 문맥 내 학습으로

우리가 이 지속적 학습, 진짜 지능 같은 것에 도달하는 것처럼 느껴질까요? 반면 사전 훈련에서는 유사한 느낌을 얻지 못합니다. 논쟁할 수 있습니다.

If it's the same algorithm, what could be different? One way you could think about it is, how much information does the model store per information it receives from training? If you look at pre-training, if you look at Llama 3 for example, I think it's trained on 15 trillion tokens. If you look at the 70B model, that would be the equivalent of 0.07 bits per token that it sees in pre-training, in terms of the information in the weights of the model compared to the tokens it reads. Whereas if you look at the KV cache and how it grows per additional token in in-context learning, it's like 320 kilobytes. So that's a 35 million-fold difference in how much information per token is assimilated by the model. I wonder if that's relevant at all.
같은 알고리즘이라면 무엇이 다를 수 있을까요? 생각해볼 수 있는 한 가지 방법은 모델이 훈련에서 받는 정보당 얼마나 많은 정보를 저장하느냐입니다. 사전 훈련을 보면, 예를 들어 Llama 3를 보면 15조 토큰으로 훈련됩니다. 70B 모델을 보면 사전 훈련에서 보는 토큰에 비해 모델 가중치의 정보 측면에서 토큰당 0.07비트에 해당합니다. 반면 KV 캐시와 문맥 내 학습에서 추가 토큰당 성장하는 방식을 보면 320킬로바이트 정도입니다. 모델이 동화하는 토큰당 정보에서 3500만 배 차이입니다. 그것이 관련이 있는지 궁금합니다.

Andrej Karpathy *00:17:46*

I kind of agree. The way I usually put this is that anything that happens during the training of the neural network, the knowledge is only a hazy recollection of what happened in training time. That's because the compression is dramatic. You're taking 15 trillion tokens and you're compressing it to just your final neural network of a few billion parameters. Obviously it's a massive amount of compression going on. So I refer to it as a hazy recollection of the internet documents.
어느 정도 동의합니다. 제가 보통 이것을 표현하는 방식은 신경망 훈련 중에 일어나는 모든 것, 그 지식은 훈련 시간에 일어난 일의 흐릿한 기억일 뿐이라는 것입니다. 왜냐하면 압축이 극적이기 때문입니다. 15조 토큰을 가져다가 몇 십억 매개변수의 최종 신경망으로 압축하는 것입니다. 분명히 엄청난 양의 압축이 진행되고 있습니다. 그래서 저는 그것을 인터넷 문서의 흐릿한 기억이라고 부릅니다.

Whereas anything that happens in the context window of the neural network—you're plugging in all the tokens and building up all those KV cache representations—is very directly accessible to the neural net. So I compare the KV cache and the stuff that happens at test time to more like a working memory. All the stuff that's in the context window is very directly accessible to the neural net.
반면 신경망의 문맥 창에서 일어나는 모든 것 - 모든 토큰을 연결하고 모든 KV 캐시 표현을 구축하는 것 - 은 신경망에 매우 직접적으로 접근 가능합니다. 그래서 저는 KV 캐시와 테스트 시간에 일어나는 것들을 작업 메모리에 더 가깝다고 비교합니다. 문맥 창의 모든 것은 신경망에 매우 직접적으로 접근 가능합니다.

There's always these almost surprising analogies between LLMs and humans. I find them surprising because we're not trying to build a human brain directly. We're just finding that this works and we're doing it. But I do think that anything that's in the weights, it's a hazy recollection of what you read a year ago. Anything that you give it as a context at test time is directly in the working memory. That's a very powerful analogy to think through things.
우리가 LLM을 직접 구축하려고 하지 않기 때문에 놀라운 LLM과 인간 사이의 유추가 항상 있습니다. 우리는 그저 이것이 작동한다는 것을 발견하고 하고 있을 뿐입니다. 하지만 가중치의 모든 것은 1년 전에 읽은 것의 흐릿한 기억이라고 생각합니다. 테스트 시간에 문맥으로 제공하는 모든 것은 작업 메모리에 직접 있습니다. 그것은 일을 생각하는 매우 강력한 유추입니다.

When you, for example, go to an LLM and you ask it about some book and what happened in it, like Nick Lane's book or something like that, the LLM will often give you some stuff which is roughly correct. But if you give it the full chapter and ask it questions, you're going to get much better results because it's now loaded in the working memory of the model. So a very long way of saying I agree and that's why.
예를 들어 LLM에 가서 닉 레인의 책이나 그 안에서 일어난 일에 대해 묻는다면, LLM은 종종 대략적으로 맞는 것을 줄 것입니다. 하지만 전체 챕터를 주고 질문을 하면 모델의 작업 메모리에 로드되었기 때문에 훨씬 더 나은 결과를 얻을 것입니다. 동의한다고 말하는 매우 긴 방법이고 그것이 이유입니다.

Dwarkesh Patel *00:19:11*

Stepping back, what is the part about human intelligence that we have most failed to replicate with these models?
뒤로 물러나서, 이 모델들로 복제하는 데 가장 실패한 인간 지능의 부분은 무엇입니까?

Andrej Karpathy *00:19:20*

Just a lot of it. So maybe one way to think about it, I don't know if this is the best way, but I almost feel like — again, making these analogies imperfect as they are — we've stumbled by with the transformer neural network, which is extremely powerful, very general. You can train transformers on audio, or video, or text, or whatever you want, and it just learns patterns and they're very powerful, and it works really well. That to me almost indicates that this is some piece of cortical tissue. It's something like that, because the cortex is famously very plastic as well. You can rewire parts of brains. There were the slightly gruesome experiments with rewiring the visual cortex to the auditory cortex, and this animal learned fine, et cetera.
그냥 많은 부분입니다. 생각하는 한 가지 방법은, 이것이 최선의 방법인지 모르겠지만, 제가 거의 느끼는 것은 - 다시 말하지만, 이러한 유추는 불완전하지만 - 우리는 매우 강력하고 매우 일반적인 트랜스포머 신경망을 우연히 발견했습니다. 오디오, 비디오, 텍스트 또는 원하는 모든 것에 트랜스포머를 훈련시킬 수 있고, 패턴을 배우고 매우 강력하며 정말 잘 작동합니다. 그것은 거의 어떤 피질 조직 조각임을 나타냅니다. 피질도 유명하게 매우 가소성이 있기 때문에 그런 것 같습니다. 뇌의 일부를 다시 연결할 수

있습니다. 시각 피질을 청각 피질에 다시 연결하는 약간 끔찍한 실험이 있었고, 이 동물은 잘 배웠습니다, 등등.

So I think that this is cortical tissue. I think when we're doing reasoning and planning inside the neural networks, doing reasoning traces for thinking models, that's kind of like the prefrontal cortex. Maybe those are like little checkmarks, but I still think there are many brain parts and nuclei that are not explored. For example, there's a basal ganglia doing a bit of reinforcement learning when we fine-tune the models on reinforcement learning. But where's the hippocampus? Not obvious what that would be. Some parts are probably not important. Maybe the cerebellum is not important to cognition, its thoughts, so maybe we can skip some of it. But I still think there's, for example, the amygdala, all the emotions and instincts. There's probably a bunch of other nuclei in the brain that are very ancient that I don't think we've really replicated.
그래서 이것이 피질 조직이라고 생각합니다. 신경망 내부에서 추론과 계획을 수행하고, 사고 모델을 위한 추론 추적을 수행할 때, 그것은 전두엽 피질과 같습니다. 아마 작은 체크마크 같은 것이지만, 아직 탐구되지 않은 많은 뇌 부분과 핵이 있다고 생각합니다. 예를 들어, 강화 학습으로 모델을 미세 조정할 때 약간의 강화 학습을 수행하는 기저핵이 있습니다. 하지만 해마는 어디에 있습니까? 그것이 무엇인지 명확하지 않습니다. 일부 부분은 아마 중요하지 않을 것입니다. 아마 소뇌는 인지나 생각에 중요하지 않을 수 있으므로 일부를 건너뛸 수 있을 것입니다. 하지만 예를 들어 편도체, 모든 감정과 본능이 있습니다. 제가 생각하기에 우리가 정말로 복제하지 못한 뇌의 매우 고대의 다른 핵들이 아마 많이 있을 것입니다.

I don't know that we should be pursuing the building of an analog of a human brain. I'm an engineer mostly at heart. Maybe another way to answer the question is that you're not going to hire this thing as an intern. It's missing a lot of it because it comes with a lot of these cognitive deficits that we all intuitively feel when we talk to the models. So it's not fully there yet. You can look at it as not all the brain parts are checked off yet.
우리가 인간 뇌의 유사체를 구축하는 것을 추구해야 하는지 모르겠습니다. 저는 대부분 마음속으로 엔지니어입니다. 질문에 답하는 또 다른 방법은 이것을 인턴으로 고용하지 않을 것이라는 것입니다. 우리 모두가 모델과 대화할 때 직관적으로 느끼는 이러한 모든 인지 결함이 함께 제공되기 때문에 많은 것이 빠져 있습니다. 그래서 아직 완전히 거기에 있지 않습니다. 아직 모든 뇌 부분이 체크되지 않은 것으로 볼 수 있습니다.

Dwarkesh Patel *00:21:16*

This is maybe relevant to the question of thinking about how fast these issues will be solved. Sometimes people will say about continual learning, "Look, you could easily replicate this capability. Just as in-context learning emerged spontaneously as a result of pre-training, continual learning over longer horizons will emerge spontaneously if the model is incentivized to recollect information over longer horizons, or horizons longer than one session." So if there's some outer loop RL which has many sessions within that outer loop, then this continual learning where it fine-tunes itself, or it writes to an external memory or something, will just emerge spontaneously. Do you think things like that are plausible? I just don't have a prior over how plausible that is.

How likely is that to happen?
이것은 아마 이러한 문제가 얼마나 빨리 해결될 것인지 생각하는 데 관련이 있을 것입니다. 때때로 사람들은 지속적 학습에 대해 "보세요, 이 능력을 쉽게 복제할 수 있습니다. 사전 훈련의 결과로 문맥 내 학습이 자발적으로 나타난 것처럼, 모델이 한 세션보다 긴 지평선에 걸쳐 정보를 기억하도록 인센티브를 받으면 더 긴 지평선에 걸친 지속적 학습이 자발적으로 나타날 것입니다"라고 말할 것입니다. 그래서 그 외부 루프 내에 많은 세션이 있는 외부 루프 RL이 있다면, 스스로 미세 조정하거나 외부 메모리에 쓰는 이 지속적 학습이 자발적으로 나타날 것입니다. 그런 것들이 타당하다고 생각하시나요? 저는 그것이 얼마나 타당한지에 대한 사전 지식이 없습니다. 그것이 일어날 가능성은 얼마나 됩니까?

Andrej Karpathy *00:22:07*

I don’t know that I fully resonate with that. These models, when you boot them up and they have zero tokens in the window, they’re always restarting from scratch where they were. So I don’t know in that worldview what it looks like. Maybe making some analogies to humans—just because I think it’s roughly concrete and interesting to think through—I feel like when I’m awake, I’m building up a context window of stuff that’s happening during the day. But when I go to sleep, something magical happens where I don’t think that context window stays around. There’s some process of distillation into the weights of my brain. This happens during sleep and all this stuff.
그것과 완전히 공감하는지 모르겠습니다. 이 모델들을 부팅하고 창에 토큰이 0개일 때, 그들은 항상 있던 곳에서 처음부터 다시 시작합니다. 그래서 그 세계관에서 어떻게 보이는지 모르겠습니다. 인간과 유추하자면 - 대략적으로 구체적이고 생각하기에 흥미롭다고 생각하기 때문입니다 - 저는 깨어 있을 때 낮 동안 일어나는 일들의 문맥 창을 구축하고 있다고 느낍니다. 하지만 잠들 때 마법 같은 일이 일어나는데, 그 문맥 창이 계속 남아 있다고 생각하지 않습니다. 내 뇌의 가중치로 증류되는 과정이 있습니다. 이것은 수면 중에 일어나고 이 모든 것입니다.

We don’t have an equivalent of that in large language models. That’s to me more adjacent to when you talk about continual learning and so on as absent. These models don’t really have a distillation phase of taking what happened, analyzing it obsessively, thinking through it, doing some synthetic data generation process and distilling it back into the weights, and maybe having a specific neural net per person. Maybe it’s a LoRA. It’s not a full-weight neural network. It’s just some small sparse subset of the weights that are changed.
우리는 대규모 언어 모델에 그에 상응하는 것이 없습니다. 그것이 지속적 학습 등에 대해 이야기할 때 없다고 더 인접한 것입니다. 이 모델들은 일어난 일을 가져와서 집착적으로 분석하고, 생각하고, 어떤 합성 데이터 생성 과정을 수행하고 그것을 다시 가중치로 증류하고, 어쩌면 사람당 특정 신경망을 갖는 증류 단계가 실제로 없습니다. 아마 LoRA일 것입니다. 전체 가중치 신경망이 아닙니다. 변경되는 가중치의 작은 희소 하위 집합일 뿐입니다.

But we do want to create ways of creating these individuals that have very long context. It’s not only remaining in the context window because the context windows grow very, very long. Maybe we have some very elaborate,

sparse attention over it. But I still think that humans obviously have some process for distilling some of that knowledge into the weights. We're missing it. I do also think that humans have some very elaborate, sparse attention scheme, which I think we're starting to see some early hints of. DeepSeek v3.2 just came out and I saw that they have sparse attention as an example, and this is one way to have very, very long context windows. So I feel like we are redoing a lot of the cognitive tricks that evolution came up with through a very different process. But we're going to converge on a similar architecture cognitively.
하지만 우리는 매우 긴 문맥을 가진 이러한 개인을 만드는 방법을 만들고 싶습니다. 문맥 창이 매우 길게 성장하기 때문에 문맥 창에만 남아 있는 것은 아닙니다. 아마 그것에 대해 매우 정교하고 희소한 주의를 가지고 있을 것입니다. 하지만 인간은 분명히 그 지식 중 일부를 가중치로 증류하는 과정을 가지고 있다고 생각합니다. 우리는 그것을 놓치고 있습니다. 또한 인간은 매우 정교하고 희소한 주의 체계를 가지고 있다고 생각하는데, 우리가 초기 힌트를 보기 시작했다고 생각합니다. DeepSeek v3.2가 방금 나왔고 희소 주의를 예로 가지고 있는 것을 봤는데, 이것은 매우 긴 문맥 창을 갖는 한 가지 방법입니다. 그래서 우리는 진화가 매우 다른 과정을 통해 생각해낸 많은 인지 트릭을 다시 하고 있다고 느낍니다. 하지만 우리는 인지적으로 유사한 아키텍처로 수렴할 것입니다.

Dwarkesh Patel *00:24:02*

In 10 years, do you think it'll still be something like a transformer, but with much more modified attention and more sparse MLPs and so forth?
10년 후에도 여전히 트랜스포머 같은 것이 될 거라고 생각하시나요? 하지만 훨씬 더 수정된 주의와 더 희소한 MLP 등이 있을까요?

Andrej Karpathy *00:24:10*

The way I like to think about it is translation invariance in time. So 10 years ago, where were we? 2015. In 2015, we had convolutional neural networks primarily, residual networks just came out. So remarkably similar, I guess, but quite a bit different still. The transformer was not around. All these more modern tweaks on the transformer were not around. Maybe some of the things that we can bet on, I think in 10 years by translational equivariance, is that we're still training giant neural networks with a forward backward pass and update through gradient descent, but maybe it looks a bit different, and it's just that everything is much bigger.
제가 생각하기 좋아하는 방식은 시간에서의 변환 불변성입니다. 10년 전 어디에 있었나요? 2015년. 2015년에 우리는 주로 합성곱 신경망을 가지고 있었고, 잔차 네트워크가 막 나왔습니다. 그래서 현저하게 유사하지만 여전히 꽤 다릅니다. 트랜스포머는 없었습니다. 트랜스포머의 이러한 모든 현대적인 조정은 없었습니다. 변환 등가성으로 10년 후에 우리가 여전히 베팅할 수 있는 것들 중 일부는 여전히 순방향 역방향 패스로 거대한 신경망을 훈련시키고 경사 하강법을 통해 업데이트하는 것이지만, 아마 조금 다르게 보일 것이고, 모든 것이 훨씬 더 클 것입니다.

Recently I went back all the way to 1989 which was a fun exercise for me, a few years ago, because I was reproducing Yann LeCun's 1989 convolutional network, which was the first neural network I'm aware of trained via gradient descent, like modern neural network trained gradient descent on digit recognition. I was just interested in how I could modernize this. How much of this is algorithms? How much of this is data? How much of this progress is compute and systems? I was able to very quickly halve the learning just by time traveling by 33 years.
최근에 재미있는 연습으로 1989년까지 거슬러 올라갔는데, 몇 년 전이었습니다. Yann LeCun의 1989년 합성곱 네트워크를 재현하고 있었는데, 이것은 제가 아는 한 경사 하강법으로 훈련된 첫 번째 신경망이었습니다. 현대 신경망처럼 숫자 인식에 대한 경사 하강법으로 훈련되었습니다. 저는 이것을 어떻게 현대화할 수 있는지에 관심이 있었습니다. 이것 중 얼마나 많은 것이 알고리즘입니까? 얼마나 많은 것이 데이터입니까? 이 진보의 얼마나 많은 것이 계산과 시스템입니까? 33년 시간 여행으로 매우 빠르게 학습을 절반으로 줄일 수 있었습니다.

So if I time travel by algorithms 33 years, I could adjust what Yann LeCun did in 1989, and I could halve the error. But to get further gains, I had to add a lot more data, I had to 10x the training set, and then I had to add more computational optimizations. I had to train for much longer with dropout and other regularization techniques.
그래서 알고리즘으로 33년 시간 여행을 하면 1989년에 Yann LeCun이 한 것을 조정할 수 있었고 오류를 절반으로 줄일 수 있었습니다. 하지만 더 많은 이득을 얻으려면 훨씬 더 많은 데이터를 추가해야 했고, 훈련 세트를 10배로 늘려야 했으며, 더 많은 계산 최적화를 추가해야 했습니다. 드롭아웃 및 기타 정규화 기술로 훨씬 더 오래 훈련해야 했습니다.

So all these things have to improve simultaneously. We're probably going to have a lot more data, we're probably going to have a lot better hardware, probably going to have a lot better kernels and software, we're probably going to have better algorithms. All of those, it's almost like no one of them is winning too much. All of them are surprisingly equal. This has been the trend for a while.
그래서 이 모든 것들이 동시에 개선되어야 합니다. 아마 훨씬 더 많은 데이터를 가질 것이고, 아마 훨씬 더 나은 하드웨어를 가질 것이고, 아마 훨씬 더 나은 커널과 소프트웨어를 가질 것이고, 아마 더 나은 알고리즘을 가질 것입니다. 그 모든 것 중에서 어느 하나도 너무 많이 이기지 않는 것 같습니다. 그것들은 모두 놀라울 정도로 동등합니다. 이것은 한동안 추세였습니다.

So to answer your question, I expect differences algorithmically to what's happening today. But I do also expect that some of the things that have stuck around for a very long time will probably still be there. It's probably still a giant neural network trained with gradient descent. That would be my guess.
그래서 귀하의 질문에 답하자면, 오늘 일어나고 있는 일에 대해 알고리즘적으로 차이를 기대합니다. 하지만 매우 오랫동안 고착된 일부 것들이 아마 여전히 있을 것으로 기대합니다. 아마 여전히 경사 하강법으로 훈련된 거대한 신경망일 것입니다. 그것이 제 추측입니다.

Dwarkesh Patel *00:26:16*

It's surprising that all of those things together only halved the error, 30 years of progress…. Maybe half is a lot. Because if you halve the error, that actually means that…
그 모든 것들이 함께 오류를 절반으로만 줄였다는 것이 놀랍습니다. 30년의 진보가... 아마 절반이 많을 수도 있습니다. 왜냐하면 오류를 절반으로 줄이면 실제로...

Andrej Karpathy *00:26:30*

Half is a lot. But I guess what was shocking to me is everything needs to improve across the board: architecture, optimizer, loss function. It also has improved across the board forever. So I expect all those changes to be alive and well.
절반이 많습니다. 하지만 제게 충격적이었던 것은 모든 것이 전반적으로 개선되어야 한다는 것입니다: 아키텍처, 옵티마이저, 손실 함수. 그것도 영원히 전반적으로 개선되었습니다. 그래서 그 모든 변화가 살아있고 건재할 것으로 기대합니다.

Dwarkesh Patel *00:26:43*

Yeah. I was about to ask you a very similar question about nanochat. Since you just coded it up recently, every single step in the process of building a chatbot is fresh in your RAM. I'm curious if you had similar thoughts about, "Oh, there was no one thing that was relevant to going from GPT-2 to nanochat." What are some surprising takeaways from the experience?
네. 나노챗에 대해 매우 유사한 질문을 하려고 했습니다. 최근에 코딩하셨기 때문에 챗봇을 구축하는 과정의 모든 단계가 RAM에 신선하게 남아 있습니다. GPT-2에서 나노챗으로 가는 데 관련된 단 하나의 것이 없다는 것에 대해 비슷한 생각이 있었는지 궁금합니다. 그 경험에서 놀라운 교훈은 무엇입니까?

Andrej Karpathy *00:27:08*

Of building nanochat? So nanochat is a repository I released. Was it yesterday or the day before? I can't remember.
나노챗을 구축하는 것? 그래서 나노챗은 제가 공개한 저장소입니다. 어제였나 그제였나? 기억이 안 나네요.

Dwarkesh Patel *00:27:15*

We can see the sleep deprivation that went into the…
들어간 수면 부족을 볼 수 있습니다...

Andrej Karpathy *00:27:18*

It’s trying to be the simplest complete repository that covers the whole pipeline end-to-end of building a ChatGPT clone. So you have all of the steps, not just any individual step, which is a bunch. I worked on all the individual steps in the past and released small pieces of code that show you how that’s done in an algorithmic sense, in simple code. But this handles the entire pipeline. In terms of learning, I don’t know that I necessarily found something that I learned from it. I already had in my mind how you build it. This is just the process of mechanically building it and making it clean enough so that people can learn from it and that they find it useful.
ChatGPT 클론을 구축하는 전체 파이프라인을 엔드투엔드로 다루는 가장 간단한 완전한 저장소가 되려고 합니다. 그래서 개별 단계뿐만 아니라 모든 단계를 가지고 있는데, 그것은 많습니다. 과거에 모든 개별 단계에서 작업했고 간단한 코드로 어떻게 수행되는지 알고리즘적 의미로 보여주는 작은 코드 조각을 공개했습니다. 하지만 이것은 전체 파이프라인을 처리합니다. 학습 측면에서 반드시 그것으로부터 배운 것을 찾았는지 모르겠습니다. 이미 머릿속에 어떻게 구축하는지 있었습니다. 이것은 단지 그것을 기계적으로 구축하고 사람들이 배울 수 있고 유용하다고 생각할 정도로 깨끗하게 만드는 과정일 뿐입니다.

Dwarkesh Patel *00:28:04*

What is the best way for somebody to learn from it? Is it to just delete all the code and try to reimplement from scratch, try to add modifications to it?
누군가가 그것으로부터 배우는 가장 좋은 방법은 무엇입니까? 모든 코드를 삭제하고 처음부터 다시 구현하려고 시도하거나 수정 사항을 추가하려고 시도하는 것입니까?

Andrej Karpathy *00:28:10*

That’s a great question. Basically it’s about 8,000 lines of code that takes you through the entire pipeline. I would probably put it on the right monitor. If you have two monitors, you put it on the right. You want to build it from scratch, you build it from the start. You’re not allowed to copy-paste, you’re allowed to reference, you’re not allowed to copy-paste. Maybe that’s how I would do it.
좋은 질문입니다. 기본적으로 전체 파이프라인을 거치는 약 8,000줄의 코드입니다. 아마 오른쪽 모니터에 놓을 것입니다. 모니터가 두 개 있다면 오른쪽에 놓으세요. 처음부터 구축하고 싶고, 처음부터 구축합니다. 복사-붙여넣기는 허용되지 않고, 참조는 허용되지만 복사-붙여넣기는 허용되지 않습니다. 아마 제가 그렇게 할 것입니다.

But I also think the repository by itself is a pretty large beast. When you write this code, you don’t go from top to bottom, you go from chunks and you grow the chunks, and that information is absent. You wouldn’t know where to start. So it’s not just a final repository that’s needed, it’s the building of the repository, which is a complicated chunk-growing process. So that part is not there yet. I would love to add that probably later this

week. It's probably a video or something like that. Roughly speaking, that's what I would try to do. Build the stuff yourself, but don't allow yourself copy-paste.
하지만 저장소 자체는 꽤 큰 짐승이라고 생각합니다. 이 코드를 작성할 때 위에서 아래로 가지 않고 청크에서 가서 청크를 성장시키는데, 그 정보는 없습니다. 어디서 시작해야 할지 모를 것입니다. 그래서 필요한 것은 최종 저장소뿐만 아니라 저장소를 구축하는 것인데, 이것은 복잡한 청크 성장 과정입니다. 그 부분은 아직 없습니다. 아마 이번 주 후반에 추가하고 싶습니다. 아마 비디오나 그런 것일 겁니다.
대략적으로 말해서, 그것이 제가 하려고 하는 것입니다. 스스로 물건을 만들되 복사-붙여넣기는 허용하지 마세요.

I do think that there's two types of knowledge, almost. There's the high-level surface knowledge, but when you build something from scratch, you're forced to come to terms with what you don't understand and you don't know that you don't understand it.
두 가지 유형의 지식이 있다고 생각합니다. 고수준 표면 지식이 있지만, 무언가를 처음부터 구축할 때 이해하지 못하는 것과 이해하지 못한다는 것을 모르는 것과 마주하게 됩니다.

It always leads to a deeper understanding. It's the only way to build. If I can't build it, I don't understand it. That's a Feynman quote, I believe. I 100% have always believed this very strongly, because there are all these micro things that are just not properly arranged and you don't really have the knowledge. You just think you have the knowledge. So don't write blog posts, don't do slides, don't do any of that. Build the code, arrange it, get it to work. It's the only way to go. Otherwise, you're missing knowledge.
항상 더 깊은 이해로 이어집니다. 구축하는 것이 유일한 방법입니다. 구축할 수 없다면 이해하지 못합니다. 그것은 파인만 인용구라고 생각합니다. 저는 이것을 항상 매우 강하게 믿어왔습니다. 왜냐하면 제대로 배열되지 않은 이 모든 마이크로 것들이 있고 실제로 지식이 없기 때문입니다. 당신은 지식이 있다고 생각할 뿐입니다. 그래서 블로그 게시물을 작성하지 말고, 슬라이드를 만들지 말고, 그런 것들을 하지 마세요. 코드를 구축하고, 정리하고, 작동하게 하세요. 그것이 유일한 방법입니다. 그렇지 않으면 지식이 누락됩니다.

00:29:45 – LLM cognitive deficits

Dwarkesh Patel *00:29:45*

You tweeted out that coding models were of very little help to you in assembling this repository. I'm curious why that was.
이 저장소를 만드는 데 코딩 모델들이 거의 도움이 되지 않았다고 트윗하셨는데요. 왜 그랬는지 궁금합니다.

Andrej Karpathy *00:29:53*

I guess I built the repository over a period of a bit more than a month. I would say there are three major classes of how people interact with code right now. Some people completely reject all of LLMs and they are just writing by scratch. This is probably not the right thing to do anymore.
한 달 조금 넘는 기간 동안 이 저장소를 만들었는데요. 사람들이 지금 코드와 상호작용하는 방식은 크게 세 가지로 나눌 수 있다고 봅니다. 어떤 사람들은 LLM을 완전히 거부하고 처음부터 직접 작성합니다. 이건 아마 더 이상 올바른 방법이 아닐 겁니다.

The intermediate part, which is where I am, is you still write a lot of things from scratch, but you use the autocomplete that's available now from these models. So when you start writing out a little piece of it, it will autocomplete for you and you can just tap through. Most of the time it's correct, sometimes it's not, and you edit it. But you're still very much the architect of what you're writing. Then there's the vibe coding: "Hi, please implement this or that," enter, and then let the model do it. That's the agents.
제가 있는 중간 단계는 여전히 많은 것을 처음부터 작성하지만, 지금 이 모델들에서 사용할 수 있는 자동완성을 사용하는 겁니다. 작은 부분을 작성하기 시작하면 자동으로 완성해주고 탭으로 진행할 수 있죠. 대부분은 맞지만 가끔 틀릴 때도 있고, 그럼 수정합니다. 하지만 여전히 당신이 작성하는 것의 설계자입니다. 그리고 '바이브 코딩'이 있습니다: "안녕, 이거나 저거 구현해줘" 엔터, 그리고 모델이 하게 둡니다. 이게 에이전트죠.

I do feel like the agents work in very specific settings, and I would use them in specific settings. But these are all tools available to you and you have to learn what they're good at, what they're not good at, and when to use them. So the agents are pretty good, for example, if you're doing boilerplate stuff. Boilerplate code that's just copy-paste stuff, they're very good at that. They're very good at stuff that occurs very often on the Internet because there are lots of examples of it in the training sets of these models. There are features of things where the models will do very well.
에이전트는 특정 상황에서는 잘 작동하고, 저도 특정 상황에서는 사용합니다. 하지만 이것들은 모두 사용 가능한 도구이고, 무엇에 능하고 무엇에 능하지 않은지, 언제 사용해야 하는지 배워야 합니다. 예를 들어 에이전트는 보일러플레이트 작업에는 꽤 좋습니다. 그냥 복사-붙여넣기 같은 보일러플레이트 코드에는 매우 좋죠. 인터넷에 자주 나타나는 것들에도 좋은데, 이런 모델들의 학습 세트에 많은 예시가 있기 때문입니다. 모델이 잘 수행할 특징들이 있습니다.

I would say nanochat is not an example of those because it's a fairly unique repository. There's not that much code in the way that I've structured it. It's not boilerplate code. It's intellectually intense code almost, and everything has to be very precisely arranged. The models have so many cognitive deficits. One example, they kept misunderstanding the code because they have too much memory from all the typical ways of doing things on the Internet that I just wasn't adopting. The models, for example—I don't know if I want to get into the full details—but they kept thinking I'm writing normal code, and I'm not.
nanochat은 그런 예가 아닙니다. 꽤 독특한 저장소거든요. 제가 구조화한 방식의 코드가 많지 않습니다. 보일러플레이트 코드가 아닙니다. 거의 지적으로 집약적인 코드이고, 모든 것이 매우 정밀하게

배치되어야 합니다. 모델들은 인지적 결함이 너무 많습니다. 한 예로, 그들은 계속 코드를 잘못 이해했는데, 제가 채택하지 않은 인터넷의 일반적인 방식에 대한 기억이 너무 많았기 때문입니다. 예를 들어---전체 세부사항에 들어가고 싶지는 않지만---그들은 계속 제가 일반 코드를 작성한다고 생각했는데, 전 그렇지 않거든요.

Dwarkesh Patel *00:31:49*

Maybe one example?
예를 한 가지 들어주실 수 있나요?

Andrej Karpathy *00:31:51*

You have eight GPUs that are all doing forward, backwards. The way to synchronize gradients between them is to use a Distributed Data Parallel container of PyTorch, which automatically as you're doing the backward, it will start communicating and synchronizing gradients. I didn't use DDP because I didn't want to use it, because it's not necessary. I threw it out and wrote my own synchronization routine that's inside the step of the optimizer. The models were trying to get me to use the DDP container. They were very concerned. This gets way too technical, but I wasn't using that container because I don't need it and I have a custom implementation of something like it.
8개의 GPU가 모두 forward, backward를 수행하고 있습니다. 그들 사이의 그래디언트를 동기화하는 방법은 PyTorch의 Distributed Data Parallel 컨테이너를 사용하는 것인데, backward를 수행하면서 자동으로 그래디언트를 통신하고 동기화합니다. 저는 DDP를 사용하지 않았습니다. 사용하고 싶지 않았고 필요하지도 않았거든요. 버리고 optimizer의 step 안에 제 자체 동기화 루틴을 작성했습니다. 모델들은 계속 DDP 컨테이너를 사용하라고 했습니다. 매우 걱정했죠. 너무 기술적이지만, 저는 그 컨테이너를 사용하지 않았습니다. 필요 없고 비슷한 것의 사용자 정의 구현이 있기 때문입니다.

Dwarkesh Patel *00:32:26*

They just couldn't internalize that you had your own.
그들은 당신이 자체 구현이 있다는 걸 내재화할 수 없었군요.

Andrej Karpathy *00:32:28*

They couldn't get past that. They kept trying to mess up the style. They're way too over-defensive. They make all these try-catch statements. They keep trying to make a production code base, and I have a bunch of assumptions in my code, and it's okay. I don't need all this extra stuff in there. So I feel like they're bloating the code base, bloating the complexity, they keep misunderstanding, they're using deprecated APIs a bunch of

times. It's a total mess. It's just not net useful. I can go in, I can clean it up, but it's not net useful.
그걸 넘어설 수 없었습니다. 계속 스타일을 망치려고 했죠. 너무 방어적입니다. try-catch 문을 다 만들어요. 계속 프로덕션 코드베이스를 만들려고 하는데, 제 코드에는 많은 가정이 있고, 괜찮습니다. 이런 추가 요소가 필요 없어요. 코드베이스를 부풀리고, 복잡성을 부풀리고, 계속 오해하고, 많은 경우 더 이상 사용되지 않는 API를 사용합니다. 완전 엉망입니다. 전혀 유용하지 않아요. 들어가서 정리할 수 있지만 실제로 유용하지 않습니다.

I also feel like it's annoying to have to type out what I want in English because it's too much typing. If I just navigate to the part of the code that I want, and I go where I know the code has to appear and I start typing out the first few letters, autocomplete gets it and just gives you the code. This is a very high information bandwidth to specify what you want. You point to the code where you want it, you type out the first few pieces, and the model will complete it.
또한 영어로 원하는 것을 타이핑하는 게 성가시다고 느낍니다. 타이핑이 너무 많거든요. 원하는 코드 부분으로 이동해서 코드가 나타나야 할 위치를 알고 처음 몇 글자를 타이핑하기 시작하면, 자동완성이 이해하고 코드를 제공합니다. 이것이 원하는 것을 지정하는 매우 높은 정보 대역폭입니다. 원하는 코드 위치를 가리키고, 처음 몇 조각을 타이핑하면 모델이 완성합니다.

So what I mean is, these models are good in certain parts of the stack. There are two examples where I use the models that I think are illustrative. One was when I generated the report. That's more boilerplate-y, so I partially vibe-coded some of that stuff. That was fine because it's not mission-critical stuff, and it works fine.
제가 말하고자 하는 것은, 이 모델들이 스택의 특정 부분에서는 좋다는 겁니다. 제가 모델을 사용한 두 가지 예시가 있습니다. 하나는 리포트를 생성할 때였습니다. 더 보일러플레이트 같아서 부분적으로 바이브 코딩했죠. 미션 크리티컬하지 않았고 잘 작동했습니다.

The other part is when I was rewriting the tokenizer in Rust. I'm not as good at Rust because I'm fairly new to Rust. So there's a bit of vibe coding going on when I was writing some of the Rust code. But I had a Python implementation that I fully understand, and I'm just making sure I'm making a more efficient version of it, and I have tests so I feel safer doing that stuff. They increase accessibility to languages or paradigms that you might not be as familiar with. I think they're very helpful there as well. There's a ton of Rust code out there, the models are pretty good at it. I happen to not know that much about it, so the models are very useful there.
다른 부분은 tokenizer를 Rust로 다시 작성할 때였습니다. Rust에 익숙하지 않아서 Rust 코드 작성할 때 약간의 바이브 코딩이 있었습니다. 하지만 제가 완전히 이해하는 Python 구현이 있었고, 더 효율적인 버전을 만들고 있다는 것을 확인하고 있었고, 테스트가 있어서 더 안전하게 느껴졌습니다. 익숙하지 않은 언어나 패러다임에 대한 접근성을 높여줍니다. 거기서 매우 도움이 된다고 생각합니다. Rust 코드가 많이 있고, 모델들이 꽤 잘합니다. 저는 잘 모르지만 모델들이 거기서 매우 유용합니다.

Dwarkesh Patel *00:34:23*

The reason this question is so interesting is because the main story people have about AI exploding and getting to superintelligence pretty rapidly is AI automating AI engineering and AI research. They'll look at the fact that you can have Claude Code and make entire applications, CRUD applications, from scratch and think, "If you had this same capability inside of OpenAI and DeepMind and everything, just imagine a thousand of you or a million of you in parallel, finding little architectural tweaks."
이 질문이 매우 흥미로운 이유는 사람들이 AI가 폭발하고 초지능에 빠르게 도달하는 주요 이야기가 AI가 AI 엔지니어링과 AI 연구를 자동화하는 것이기 때문입니다. Claude Code로 전체 애플리케이션, CRUD 애플리케이션을 처음부터 만들 수 있다는 사실을 보고 "OpenAI와 DeepMind 등에서 이와 같은 능력이 있다면, 당신의 수천 또는 수백만 복사본이 병렬로 작은 아키텍처 개선을 찾는다고 상상해보세요"라고 생각합니다.

It's quite interesting to hear you say that this is the thing they're asymmetrically worse at. It's quite relevant to forecasting whether the AI 2027-type explosion is likely to happen anytime soon.
이것이 그들이 비대칭적으로 더 못하는 것이라고 듣는 것은 꽤 흥미롭습니다. AI 2027 같은 폭발이 곧 일어날 가능성을 예측하는 데 꽤 관련이 있습니다.

Andrej Karpathy *00:35:05*

That's a good way of putting it, and you're getting at why my timelines are a bit longer. You're right. They're not very good at code that has never been written before, maybe it's one way to put it, which is what we're trying to achieve when we're building these models.
좋은 표현입니다. 제 타임라인이 조금 더 긴 이유를 이해하셨네요. 맞습니다. 이전에 작성된 적이 없는 코드에는 매우 능숙하지 않습니다. 이것이 우리가 이 모델들을 만들 때 달성하려는 것입니다.

Dwarkesh Patel *00:35:19*

Very naive question, but the architectural tweaks that you're adding to nanochat, they're in a paper somewhere, right? They might even be in a repo somewhere. Is it surprising that they aren't able to integrate that into whenever you're like, "Add RoPE embeddings" or something, they do that in the wrong way?
매우 순진한 질문이지만, nanochat에 추가하는 아키텍처 개선사항들은 어딘가 논문에 있지 않나요? 어딘가 저장소에도 있을 수 있고요. "RoPE 임베딩 추가"같은 걸 말할 때 그들이 잘못된 방식으로 하는 게 놀랍지 않나요?

Andrej Karpathy *00:35:42*

It's tough. They know, but they don't fully know. They don't know how to fully integrate it into the repo and your style and your code and your place, and some of the custom things that you're doing and how it fits with all the assumptions of the repository. They do have some knowledge, but they haven't gotten to the place where

they can integrate it and make sense of it.
어렵습니다. 그들은 알지만 완전히 알지는 못합니다. 저장소와 스타일과 코드와 위치, 그리고 당신이 하는 사용자 정의 작업들과 어떻게 통합하는지, 저장소의 모든 가정과 어떻게 맞는지 완전히 모릅니다. 어느 정도 지식은 있지만 통합하고 이해할 수 있는 위치에 도달하지 못했습니다.

A lot of the stuff continues to improve. Currently, the state-of-the-art model that I go to is the GPT-5 Pro, and that's a very powerful model. If I have 20 minutes, I will copy-paste my entire repo and I go to GPT-5 Pro, the oracle, for some questions. Often it's not too bad and surprisingly good compared to what existed a year ago.
많은 것들이 계속 개선되고 있습니다. 현재 제가 사용하는 최첨단 모델은 GPT-5 Pro인데, 매우 강력한 모델입니다. 20분이 있으면 전체 저장소를 복사-붙여넣기하고 GPT-5 Pro, 오라클에게 질문합니다. 종종 나쁘지 않고 1년 전에 비해 놀랍게 좋습니다.

Overall, the models are not there. I feel like the industry is making too big of a jump and is trying to pretend like this is amazing, and it's not. It's slop. They're not coming to terms with it, and maybe they're trying to fundraise or something like that. I'm not sure what's going on, but we're at this intermediate stage. The models are amazing. They still need a lot of work. For now, autocomplete is my sweet spot. But sometimes, for some types of code, I will go to an LLM agent.
전반적으로 모델들이 아직 거기까지 도달하지 못했습니다. 업계가 너무 큰 도약을 하고 있고 이것이 놀랍다고 가장하려고 한다고 느낍니다. 그렇지 않습니다. 쓰레기입니다. 그들은 이를 인정하지 않고 있고, 아마 자금을 모으려고 하거나 그런 것 같습니다. 무슨 일이 일어나고 있는지 확실하지 않지만, 우리는 이 중간 단계에 있습니다. 모델들은 놀랍습니다. 여전히 많은 작업이 필요합니다. 지금은 자동완성이 제 스위트 스팟입니다. 하지만 때때로, 일부 유형의 코드에 대해서는 LLM 에이전트를 사용합니다.

Dwarkesh Patel *00:36:53*

Here's another reason this is really interesting. Through the history of programming, there have been many productivity improvements—compilers, linting, better programming languages—which have increased programmer productivity but have not led to an explosion. That sounds very much like the autocomplete tab, and this other category is just automation of the programmer. It's interesting you're seeing more in the category of the historical analogies of better compilers or something.
이것이 정말 흥미로운 또 다른 이유가 있습니다. 프로그래밍 역사를 통틀어 많은 생산성 향상이 있었습니다---컴파일러, 린팅, 더 나은 프로그래밍 언어---프로그래머 생산성을 높였지만 폭발로 이어지지는 않았습니다. 그것은 자동완성 탭 카테고리와 매우 비슷하게 들리고, 다른 카테고리는 프로그래머의 자동화입니다. 더 나은 컴파일러 같은 역사적 유추 카테고리에서 더 많이 보고 있다는 게 흥미롭네요.

Andrej Karpathy *00:37:26*

Maybe this gets to one other thought. I have a hard time differentiating where AI begins and stops because I see AI as fundamentally an extension of computing in a pretty fundamental way. I see a continuum of this recursive self-improvement or speeding up programmers all the way from the beginning: code editors, syntax highlighting, or checking even of the types, like data type checking—all these tools that we've built for each other.
이것이 또 다른 생각으로 이어집니다. 저는 AI가 어디서 시작하고 멈추는지 구분하기 어렵습니다. AI를 근본적으로 꽤 기본적인 방식으로 컴퓨팅의 확장으로 보기 때문입니다. 이 재귀적 자기 개선이나 프로그래머 속도 향상의 연속체를 처음부터 봅니다: 코드 편집기, 구문 강조, 타입 체크 같은 것들---우리가 서로를 위해 만든 모든 이런 도구들.

Even search engines. Why aren't search engines part of AI? Ranking is AI. At some point, Google, even early on, was thinking of themselves as an AI company doing Google Search engine, which is totally fair.
검색 엔진도 마찬가지입니다. 왜 검색 엔진이 AI의 일부가 아닌가요? 랭킹은 AI입니다. 어느 시점에서 구글도 초기에도 자신들을 구글 검색 엔진을 하는 AI 회사로 생각했고, 완전히 타당합니다.

I see it as a lot more of a continuum than other people do, and it's hard for me to draw the line. I feel like we're now getting a much better autocomplete, and now we're also getting some agents which are these loopy things, but they go off-rails sometimes. What's going on is that the human is progressively doing a bit less and less of the low-level stuff. We're not writing the assembly code because we have compilers. Compilers will take my high-level language in C and write the assembly code.
저는 다른 사람들보다 훨씬 더 연속체로 봅니다. 선을 긋기가 어렵습니다. 우리는 이제 훨씬 더 나은 자동완성을 얻고 있고, 이제 루프 형태의 에이전트도 얻고 있지만 때때로 궤도를 벗어납니다. 일어나고 있는 일은 인간이 점진적으로 저수준 작업을 조금씩 덜 하고 있다는 것입니다. 우리는 어셈블리 코드를 작성하지 않습니다. 컴파일러가 있기 때문입니다. 컴파일러가 C로 된 제 고수준 언어를 가져와 어셈블리 코드를 작성합니다.

We're abstracting ourselves very, very slowly. There's this what I call "autonomy slider," where more and more stuff is automated—of the stuff that can be automated at any point in time—and we're doing a bit less and less and raising ourselves in the layer of abstraction over the automation.
우리는 매우 천천히 추상화하고 있습니다. 제가 "자율성 슬라이더"라고 부르는 것이 있는데, 점점 더 많은 것이 자동화됩니다---어느 시점에서든 자동화할 수 있는 것 중에서---그리고 우리는 점점 덜 하고 자동화 위의 추상화 계층에서 우리 자신을 높이고 있습니다.

00:40:05 – RL is terrible

Dwarkesh Patel *00:40:05*

Let's talk about RL a bit. You tweeted some very interesting things about this. Conceptually, how should we think about the way that humans are able to build a rich world model just from interacting with our

environment, and in ways that seem almost irrespective of the final reward at the end of the episode?
RL에 대해 좀 이야기해봅시다. 당신이 트윗한 매우 흥미로운 내용들이 있었죠. 개념적으로, 인간이 단순히 환경과 상호작용하는 것만으로 풍부한 세계 모델을 구축할 수 있는 방식을 어떻게 이해해야 할까요? 그것도 에피소드 끝의 최종 보상과는 거의 무관하게 말이죠.

If somebody is starting a business, and at the end of 10 years, she finds out whether the business succeeded or failed, we say that she's earned a bunch of wisdom and experience. But it's not because the log probs of every single thing that happened over the last 10 years are up-weighted or down-weighted. Something much more deliberate and rich is happening. What is the ML analogy, and how does that compare to what we're doing with LLMs right now?
누군가 사업을 시작하고 10년 후에 그 사업이 성공했는지 실패했는지 알게 된다면, 우리는 그녀가 많은 지혜와 경험을 얻었다고 말합니다. 하지만 그것은 지난 10년간 일어난 모든 일의 로그 확률이 상향 또는 하향 가중치를 받았기 때문이 아닙니다. 훨씬 더 의도적이고 풍부한 무언가가 일어나고 있죠. ML 관점에서 이것의 유사체는 무엇이고, 현재 LLM으로 우리가 하고 있는 것과는 어떻게 비교되나요?

Andrej Karpathy *00:40:47*

Maybe the way I would put it is that humans don't use reinforcement learning, as I said. I think they do something different. Reinforcement learning is a lot worse than I think the average person thinks. Reinforcement learning is terrible. It just so happens that everything that we had before it is much worse because previously we were just imitating people, so it has all these issues.
제가 표현하자면, 제가 말했듯이 인간은 강화학습을 사용하지 않습니다. 그들은 다른 것을 합니다. 강화학습은 일반인들이 생각하는 것보다 훨씬 나쁩니다. 강화학습은 형편없어요. 다만 이전에 우리가 가졌던 모든 것이 훨씬 더 나빴기 때문에 그나마 나은 것뿐입니다. 이전에는 단지 사람들을 모방하고 있었기 때문에 많은 문제가 있었죠.

In reinforcement learning, say you're solving a math problem, because it's very simple. You're given a math problem and you're trying to find the solution. In reinforcement learning, you will try lots of things in parallel first. You're given a problem, you try hundreds of different attempts. These attempts can be complex. They can be like, "Oh, let me try this, let me try that, this didn't work, that didn't work," etc. Then maybe you get an answer. Now you check the back of the book and you see, "Okay, the correct answer is this." You can see that this one, this one, and that one got the correct answer, but these other 97 of them didn't. Literally what reinforcement learning does is it goes to the ones that worked really well and every single thing you did along the way, every single token gets upweighted like, "Do more of this."
강화학습에서, 예를 들어 수학 문제를 푼다고 해봅시다. 매우 간단한 예시죠. 수학 문제가 주어지고 해답을 찾으려고 합니다. 강화학습에서는 먼저 병렬로 많은 시도를 할 겁니다. 문제가 주어지면 수백 가지의 다른 시도를 합니다. 이 시도들은 복잡할 수 있어요. "이것을 시도해보자, 저것을 시도해보자, 이건 안 됐어, 저것도 안 됐어" 등등. 그러다 답을 얻을 수도 있습니다. 이제 책 뒤의 답을 확인해서 "좋아, 정답은

이거야"라고 봅니다. 이것, 이것, 저것이 정답을 맞췄지만 나머지 97개는 틀렸다는 걸 알게 되죠.
강화학습이 문자 그대로 하는 것은 잘 작동한 것들로 가서 그 과정에서 한 모든 것, 모든 토큰에 대해 "이런 걸 더 해"라고 가중치를 높이는 것입니다.

The problem with that is people will say that your estimator has high variance, but it's just noisy. It's noisy. It almost assumes that every single little piece of the solution that you made that arrived at the right answer was the correct thing to do, which is not true. You may have gone down the wrong alleys until you arrived at the right solution. Every single one of those incorrect things you did, as long as you got to the correct solution, will be upweighted as, "Do more of this." It's terrible. It's noise.
문제는 사람들이 당신의 추정기가 높은 분산을 가지고 있다고 말하겠지만, 그냥 노이즈가 많다는 거예요. 노이즈가 많습니다. 정답에 도달하기 위해 만든 해결책의 모든 작은 조각이 올바른 행동이었다고 거의 가정하는데, 이는 사실이 아닙니다. 올바른 해답에 도달할 때까지 잘못된 길로 갔을 수도 있어요. 정답을 얻기만 하면, 당신이 한 모든 잘못된 것들도 "이런 걸 더 해"라고 가중치가 올라갑니다. 끔찍해요.
노이즈입니다.

You've done all this work only to find, at the end, you get a single number of like, "Oh, you did correct." Based on that, you weigh that entire trajectory as like, upweight or downweight. The way I like to put it is you're sucking supervision through a straw. You've done all this work that could be a minute of rollout, and you're sucking the bits of supervision of the final reward signal through a straw and you're broadcasting that across the entire trajectory and using that to upweight or downweight that trajectory. It's just stupid and crazy.
이 모든 작업을 했는데 결국 "오, 맞았네"라는 단일 숫자만 얻게 됩니다. 그것을 기반으로 전체 궤적에 대해 가중치를 높이거나 낮춥니다. 제가 표현하기를 빨대로 감독 신호를 빨아들이는 것과 같습니다. 1분의 롤아웃이 될 수 있는 이 모든 작업을 했는데, 최종 보상 신호의 감독 비트를 빨대로 빨아들여서 전체 궤적에 방송하고 그것을 사용해 해당 궤적의 가중치를 높이거나 낮춥니다. 그냥 멍청하고 미친 짓이에요.

A human would never do this. Number one, a human would never do hundreds of rollouts. Number two, when a person finds a solution, they will have a pretty complicated process of review of, "Okay, I think these parts I did well, these parts I did not do that well. I should probably do this or that." They think through things. There's nothing in current LLMs that does this. There's no equivalent of it. But I do see papers popping out that are trying to do this because it's obvious to everyone in the field.
인간은 절대 이렇게 하지 않을 겁니다. 첫째, 인간은 수백 번의 롤아웃을 하지 않을 겁니다. 둘째, 사람이 해답을 찾으면, 꽤 복잡한 리뷰 과정을 거칩니다. "좋아, 이 부분들은 잘했고, 이 부분들은 그다지 잘하지 못했어. 아마 이렇게 하거나 저렇게 해야 할 거야." 그들은 생각을 합니다. 현재 LLM에는 이런 것이 전혀 없습니다. 이에 상응하는 것이 없어요. 하지만 이 분야의 모든 사람들에게 명백하기 때문에 이것을 시도하는 논문들이 나오는 것을 봅니다.

The first imitation learning, by the way, was extremely surprising and miraculous and amazing, that we can fine-tune by imitation on humans. That was incredible. Because in the beginning, all we had was base models. Base models are autocomplete. It wasn't obvious to me at the time, and I had to learn this. The paper that blew

my mind was InstructGPT, because it pointed out that you can take the pretrained model, which is autocomplete, and if you just fine-tune it on text that looks like conversations, the model will very rapidly adapt to become very conversational, and it keeps all the knowledge from pre-training. This blew my mind because I didn't understand that stylistically, it can adjust so quickly and become an assistant to a user through just a few loops of fine-tuning on that kind of data. It was very miraculous to me that that worked. So incredible. That was two to three years of work.
첫 번째 모방 학습은 정말 놀랍고 기적적이고 대단했습니다. 인간을 모방하여 파인튜닝할 수 있다는 것이 말이죠. 정말 놀라웠어요. 처음에는 베이스 모델만 있었거든요. 베이스 모델은 자동완성입니다. 당시 저에게는 명백하지 않았고 배워야 했습니다. 제 마음을 날려버린 논문은 InstructGPT였습니다. 자동완성인 사전 훈련 모델을 가져와서 대화처럼 보이는 텍스트에 파인튜닝하면, 모델이 매우 빠르게 대화형으로 적응하고 사전 훈련의 모든 지식을 유지한다는 것을 지적했기 때문입니다. 이것은 제 마음을 날려버렸습니다. 스타일적으로 그렇게 빨리 조정되고 그런 종류의 데이터에 대한 몇 번의 파인튜닝 루프를 통해 사용자의 어시스턴트가 될 수 있다는 것을 이해하지 못했거든요. 그것이 작동한다는 것이 매우 기적적이었습니다. 정말 놀라웠어요. 그것은 2-3년의 작업이었습니다.

Now came RL. And RL allows you to do a bit better than just imitation learning because you can have these reward functions and you can hill-climb on the reward functions. Some problems have just correct answers, you can hill-climb on that without getting expert trajectories to imitate. So that's amazing. The model can also discover solutions that a human might never come up with. This is incredible. Yet, it's still stupid.
이제 RL이 왔습니다. RL은 단순한 모방 학습보다 조금 더 나은 것을 할 수 있게 해줍니다. 보상 함수를 가질 수 있고 보상 함수를 타고 올라갈 수 있기 때문이죠. 일부 문제들은 정답이 있고, 모방할 전문가 궤적을 얻지 않고도 그것을 타고 올라갈 수 있습니다. 그래서 놀랍습니다. 모델은 또한 인간이 절대 생각해내지 못할 해결책을 발견할 수도 있습니다. 이것은 놀라운 일입니다. 하지만 여전히 멍청합니다.

We need more. I saw a paper from Google yesterday that tried to have this reflect & review idea in mind. Was it the memory bank paper or something? I don't know. I've seen a few papers along these lines. So I expect there to be some major update to how we do algorithms for LLMs coming in that realm. I think we need three or four or five more, something like that.
우리는 더 많은 것이 필요합니다. 어제 구글의 논문을 봤는데 이런 reflect & review 아이디어를 염두에 둔 것 같았습니다. 메모리 뱅크 논문이었나요? 모르겠습니다. 이런 라인을 따라가는 논문을 몇 개 봤습니다. 그래서 저는 그 영역에서 LLM을 위한 알고리즘을 수행하는 방법에 대한 주요 업데이트가 있을 것으로 기대합니다. 우리는 3개, 4개, 5개 정도 더 필요하다고 생각합니다.

Dwarkesh Patel *00:44:54*

You're so good at coming up with evocative phrases. "Sucking supervision through a straw." It's so good.
당신은 강렬한 문구를 만드는 데 정말 뛰어납니다. "빨대로 감독 신호를 빨아들이기." 정말 좋네요.

You're saying the problem with outcome-based reward is that you have this huge trajectory, and then at the end, you're trying to learn every single possible thing about what you should do and what you should learn about the world from that one final bit. Given the fact that this is obvious, why hasn't process-based supervision as an alternative been a successful way to make models more capable? What has been preventing us from using this alternative paradigm?
당신이 말하는 것은 결과 기반 보상의 문제가 이 거대한 궤적이 있고, 마지막에 그 하나의 최종 비트에서 세상에 대해 무엇을 해야 하고 무엇을 배워야 하는지에 대한 모든 가능한 것을 배우려고 한다는 것입니다. 이것이 명백하다는 사실을 감안할 때, 왜 대안으로서의 프로세스 기반 감독이 모델을 더 유능하게 만드는 성공적인 방법이 되지 못했을까요? 무엇이 이 대안 패러다임을 사용하는 것을 막고 있나요?

Andrej Karpathy *00:45:29*

Process-based supervision just refers to the fact that we're not going to have a reward function only at the very end. After you've done 10 minutes of work, I'm not going to tell you you did well or not well. I'm going to tell you at every single step of the way how well you're doing. The reason we don't have that is it's tricky how you do that properly. You have partial solutions and you don't know how to assign credit. So when you get the right answer, it's just an equality match to the answer. It's very simple to implement. If you're doing process supervision, how do you assign in an automatable way, a partial credit assignment? It's not obvious how you do it.
프로세스 기반 감독은 단지 맨 끝에만 보상 함수를 갖지 않는다는 것을 의미합니다. 10분 동안 작업을 한 후에 잘했는지 못했는지 말하지 않을 거예요. 매 단계마다 얼마나 잘하고 있는지 알려줄 겁니다. 우리가 그것을 갖지 못한 이유는 그것을 제대로 하는 것이 까다롭기 때문입니다. 부분적인 해결책이 있고 신용을 어떻게 할당할지 모릅니다. 정답을 얻으면 답과의 동등성 일치일 뿐입니다. 구현하기 매우 간단합니다. 프로세스 감독을 하고 있다면, 자동화 가능한 방식으로 부분 신용 할당을 어떻게 할당합니까? 어떻게 하는지 명백하지 않습니다.

Lots of labs are trying to do it with these LLM judges. You get LLMs to try to do it. You prompt an LLM, "Hey, look at a partial solution of a student. How well do you think they're doing if the answer is this?" and they try to tune the prompt.
많은 연구소들이 이런 LLM 심판으로 그것을 시도하고 있습니다. LLM이 그것을 시도하도록 합니다. LLM에게 프롬프트를 줍니다. "이봐, 학생의 부분 해결책을 봐. 답이 이것이라면 그들이 얼마나 잘하고 있다고 생각해?" 그리고 프롬프트를 조정하려고 합니다.

The reason that this is tricky is quite subtle. It's the fact that anytime you use an LLM to assign a reward, those LLMs are giant things with billions of parameters, and they're gameable. If you're reinforcement learning with respect to them, you will find adversarial examples for your LLM judges, almost guaranteed. So you can't do this for too long. You do maybe 10 steps or 20 steps, and maybe it will work, but you can't do 100 or 1,000. I understand it's not obvious, but basically the model will find little cracks. It will find all these spurious things in

the nooks and crannies of the giant model and find a way to cheat it.
이것이 까다로운 이유는 꽤 미묘합니다. LLM을 사용하여 보상을 할당할 때마다, 그 LLM들은 수십억 개의 매개변수를 가진 거대한 것들이고, 게임 가능합니다. 그들에 대해 강화학습을 하고 있다면, 거의 보장컨대 LLM 심판을 위한 적대적 예제를 찾을 것입니다. 그래서 너무 오래 할 수 없습니다. 10단계나 20단계 정도는 할 수 있고 작동할 수도 있지만, 100이나 1,000은 할 수 없습니다. 명백하지 않지만, 기본적으로 모델은 작은 균열을 찾을 것입니다. 거대한 모델의 구석구석에서 이런 모든 가짜 것들을 찾아 속이는 방법을 찾을 겁니다.

One example that's prominently in my mind, this was probably public, if you're using an LLM judge for a reward, you just give it a solution from a student and ask it if the student did well or not. We were training with reinforcement learning against that reward function, and it worked really well. Then, suddenly, the reward became extremely large. It was a massive jump, and it did perfect. You're looking at it like, "Wow, this means the student is perfect in all these problems. It's fully solved math."
제 마음에 떠오르는 한 가지 예는, 아마 공개된 것일 텐데, 보상을 위해 LLM 심판을 사용하는 경우, 학생의 해결책을 주고 학생이 잘했는지 아닌지 묻습니다. 우리는 그 보상 함수에 대해 강화학습으로 훈련하고 있었고, 정말 잘 작동했습니다. 그런데 갑자기 보상이 극도로 커졌습니다. 대규모 점프였고 완벽했습니다. "와, 이것은 학생이 이 모든 문제에서 완벽하다는 뜻이야. 수학을 완전히 풀었어"라고 보고 있죠.

But when you look at the completions that you're getting from the model, they are complete nonsense. They start out okay, and then they change to "dhdhdhdh." It's just like, "Oh, okay, let's take two plus three and we do this and this, and then dhdhdhdh." You're looking at it, and it's like, this is crazy. How is it getting a reward of one or 100%? You look at the LLM judge, and it turns out that "dhdhdhdh" is an adversarial example for the model, and it assigns 100% probability to it.
하지만 모델에서 얻는 완성을 보면 완전히 말도 안 되는 것들입니다. 괜찮게 시작하다가 "dhdhdhdh"로 바뀝니다. "자, 2 더하기 3을 해보자, 이렇게 저렇게 하고, 그다음 dhdhdhdh." 이걸 보면서 이건 미친 짓이야. 어떻게 1의 보상 또는 100%를 받고 있지? LLM 심판을 보면 "dhdhdhdh"가 모델에 대한 적대적 예제이고 100% 확률을 할당한다는 것이 밝혀졌습니다.

It's just because this is an out-of-sample example to the LLM. It's never seen it during training, and you're in pure generalization land. It's never seen it during training, and in the pure generalization land, you can find these examples that break it.
이것은 LLM에게 샘플 외 예제이기 때문입니다. 훈련 중에 본 적이 없고, 순수한 일반화 영역에 있습니다. 훈련 중에 본 적이 없고, 순수한 일반화 영역에서 이것을 깨뜨리는 예제를 찾을 수 있습니다.

Dwarkesh Patel *00:47:52*

You're basically training the LLM to be a prompt injection model.
기본적으로 LLM을 프롬프트 인젝션 모델로 훈련시키고 있는 거네요.

Andrej Karpathy *00:47:56*

Not even that. Prompt injection is way too fancy. You're finding adversarial examples, as they're called. These are nonsensical solutions that are obviously wrong, but the model thinks they are amazing.
그것도 아니에요. 프롬프트 인젝션은 너무 화려합니다. 그들이 적대적 예제라고 부르는 것을 찾고 있는 겁니다. 명백히 잘못된 무의미한 해결책이지만 모델은 놀랍다고 생각합니다.

Dwarkesh Patel *00:48:07*

To the extent you think this is the bottleneck to making RL more functional, then that will require making LLMs better judges, if you want to do this in an automated way. Is it just going to be some sort of GAN-like approach where you have to train models to be more robust?
이것이 RL을 더 기능적으로 만드는 병목이라고 생각한다면, 자동화된 방식으로 이것을 하려면 LLM을 더 나은 심판으로 만들어야 합니다. 모델을 더 견고하게 훈련시켜야 하는 일종의 GAN 같은 접근법이 될까요?

Andrej Karpathy *00:48:22*

The labs are probably doing all that. The obvious thing is, "dhdhdhdh" should not get 100% reward. Okay, well, take "dhdhdhdh," put it in the training set of the LLM judge, and say this is not 100%, this is 0%. You can do this, but every time you do this, you get a new LLM, and it still has adversarial examples. There's an infinity of adversarial examples.
연구소들은 아마 그 모든 것을 하고 있을 겁니다. 명백한 것은 "dhdhdhdh"가 100% 보상을 받아서는 안 된다는 것입니다. 좋아, "dhdhdhdh"를 가져와서 LLM 심판의 훈련 세트에 넣고 이것은 100%가 아니라 0%라고 말합니다. 이렇게 할 수 있지만, 이렇게 할 때마다 새로운 LLM을 얻고 여전히 적대적 예제가 있습니다. 적대적 예제는 무한합니다.

Probably if you iterate this a few times, it'll probably be harder and harder to find adversarial examples, but I'm not 100% sure because this thing has a trillion parameters or whatnot. I bet you the labs are trying. I still think we need other ideas.
아마 이것을 몇 번 반복하면 적대적 예제를 찾기가 점점 더 어려워질 것이지만, 이것이 1조 개의 매개변수나 뭐든 가지고 있기 때문에 100% 확실하지는 않습니다. 연구소들이 시도하고 있을 거라고 확신합니다. 여전히 다른 아이디어가 필요하다고 생각합니다.

Dwarkesh Patel *00:48:57*

Interesting. Do you have some shape of what the other idea could be?
흥미롭네요. 다른 아이디어가 무엇일 수 있는지 어떤 형태라도 있나요?

Andrej Karpathy *00:49:02*

This idea of a review solution encompassing synthetic examples such that when you train on them, you get better, and meta-learn it in some way. I think there are some papers that I'm starting to see pop out. I am only at a stage of reading abstracts because a lot of these papers are just ideas. Someone has to make it work on a frontier LLM lab scale in full generality because when you see these papers, they pop up, and it's just a bit noisy. They're cool ideas, but I haven't seen anyone convincingly show that this is possible. That said, the LLM labs are fairly closed, so who knows what they're doing now.
해결책을 검토하는 이 아이디어는 합성 예제를 포괄하여 그것들을 훈련할 때 더 나아지고 어떤 방식으로든 메타 학습하는 것입니다. 제가 나오기 시작하는 것을 보는 일부 논문들이 있다고 생각합니다. 저는 초록을 읽는 단계에 있을 뿐입니다. 이런 논문들 중 많은 것이 그냥 아이디어이기 때문이죠. 누군가는 프런티어 LLM 연구소 규모에서 완전한 일반성으로 작동하게 만들어야 합니다. 이런 논문들이 나올 때, 그들이 나타나고 약간 노이즈가 있습니다. 멋진 아이디어들이지만 이것이 가능하다는 것을 설득력 있게 보여준 사람을 본 적이 없습니다. 그렇긴 하지만, LLM 연구소들은 꽤 폐쇄적이므로 그들이 지금 무엇을 하고 있는지 누가 알겠어요.

00:49:38 – How do humans learn?

Dwarkesh Patel *00:49:38*

I can conceptualize how you would be able to train on synthetic examples or synthetic problems that you have made for yourself. But there seems to be another thing humans do—maybe sleep is this, maybe daydreaming is this—which is not necessarily to come up with fake problems, but just to reflect.
합성 예제나 스스로 만든 합성 문제로 훈련하는 것은 개념적으로 이해가 됩니다. 하지만 인간이 하는 또 다른 일이 있는 것 같아요. 수면이나 백일몽이 그런 것일 수도 있는데, 꼭 가짜 문제를 만드는 게 아니라 그냥 성찰하는 거죠.

I'm not sure what the ML analogy is for daydreaming or sleeping, or just reflecting. I haven't come up with a new problem. Obviously, the very basic analogy would just be fine-tuning on reflection bits, but I feel like in practice that probably wouldn't work that well. Do you have some take on what the analogy of this thing is?
저는 백일몽이나 수면, 또는 단순한 성찰에 대한 ML 유사체가 무엇인지 확실하지 않습니다. 새로운 문제를 만들지 않았잖아요. 물론 가장 기본적인 유사체는 성찰 비트에 대한 파인튜닝이겠지만, 실제로는 그게 잘 작동하지 않을 것 같아요. 이런 것의 유사체가 무엇이라고 생각하시나요?

Andrej Karpathy *00:50:17*

I do think that we're missing some aspects there. As an example, let's take reading a book. Currently when LLMs are reading a book, what that means is we stretch out the sequence of text, and the model is predicting the

next token, and it's getting some knowledge from that. That's not really what humans do. When you're reading a book, I don't even feel like the book is exposition I'm supposed to be attending to and training on. The book is a set of prompts for me to do synthetic data generation, or for you to get to a book club and talk about it with your friends. It's by manipulating that information that you actually gain that knowledge. We have no equivalent of that with LLMs. They don't really do that. I'd love to see during pre-training some stage that thinks through the material and tries to reconcile it with what it already knows, and thinks through it for some amount of time and gets that to work. There's no equivalence of any of this. This is all research.
우리가 거기서 놓치고 있는 측면들이 분명히 있다고 생각해요. 예를 들어 책 읽기를 생각해보죠. 현재 LLM이 책을 읽는다는 것은 텍스트 시퀀스를 펼쳐놓고 모델이 다음 토큰을 예측하면서 지식을 얻는 걸 의미합니다. 하지만 인간은 실제로 그렇게 하지 않죠. 책을 읽을 때, 저는 책이 제가 주의를 기울이고 훈련해야 할 설명이라고 생각하지 않아요. 책은 제가 합성 데이터를 생성하도록 하는 프롬프트 세트이거나, 북클럽에서 친구들과 이야기할 거리입니다. 실제로 그 정보를 조작함으로써 지식을 얻는 거죠. LLM에는 이에 해당하는 것이 없어요. 그들은 실제로 그렇게 하지 않죠. 저는 사전 훈련 중에 자료를 생각하고 이미 알고 있는 것과 조화시키려고 노력하며, 어느 정도 시간을 들여 생각하고 그것이 작동하도록 하는 단계를 보고 싶어요. 이런 것의 동등한 것은 없습니다. 이것은 모두 연구 영역이죠.

There are some subtle—very subtle that I think are very hard to understand—reasons why it's not trivial. If I can just describe one: why can't we just synthetically generate and train on it? Because every synthetic example, if I just give synthetic generation of the model thinking about a book, you look at it and you're like, "This looks great. Why can't I train on it?" You could try, but the model will get much worse if you continue trying. That's because all of the samples you get from models are silently collapsed. Silently—it is not obvious if you look at any individual example of it—they occupy a very tiny manifold of the possible space of thoughts about content. The LLMs, when they come off, they're what we call "collapsed." They have a collapsed data distribution. One easy way to see it is to go to ChatGPT and ask it, "Tell me a joke." It only has like three jokes. It's not giving you the whole breadth of possible jokes. It knows like three jokes. They're silently collapsed.
매우 미묘한 - 이해하기 매우 어려운 미묘한 이유들이 있는데, 왜 이것이 사소하지 않은지 말이죠. 하나를 설명해보자면: 왜 우리는 단순히 합성적으로 생성하고 그것으로 훈련할 수 없을까요? 만약 제가 책에 대해 생각하는 모델의 합성 생성을 준다면, 당신이 보기에는 "이거 좋아 보이는데, 왜 이걸로 훈련할 수 없지?"라고 할 겁니다. 시도할 수는 있지만, 계속 시도하면 모델이 훨씬 나빠질 겁니다. 왜냐하면 모델에서 얻는 모든 샘플이 조용히 붕괴되어 있기 때문이죠. 조용히 - 개별 예제를 보면 명확하지 않지만 - 콘텐츠에 대한 가능한 생각 공간의 아주 작은 다양체만을 차지합니다. LLM은 나올 때 우리가 "붕괴됐다"고 부르는 상태입니다. 붕괴된 데이터 분포를 가지고 있죠. 쉽게 알아보는 방법은 ChatGPT에 가서 "농담 하나 해줘"라고 하면 세 개 정도의 농담만 합니다. 가능한 모든 농담의 폭을 제공하지 않아요. 세 개의 농담만 알고 있죠. 조용히 붕괴되어 있습니다.

You're not getting the richness and the diversity and the entropy from these models as you would get from humans. Humans are a lot noisier, but at least they're not biased, in a statistical sense. They're not silently collapsed. They maintain a huge amount of entropy. So how do you get synthetic data generation to work

despite the collapse and while maintaining the entropy? That's a research problem.
이 모델들로부터 인간에게서 얻을 수 있는 것과 같은 풍부함과 다양성, 엔트로피를 얻지 못합니다. 인간은 훨씬 더 시끄럽지만, 적어도 통계적 의미에서 편향되지 않았어요. 조용히 붕괴되지 않았죠. 엄청난 양의 엔트로피를 유지합니다. 그래서 붕괴에도 불구하고 엔트로피를 유지하면서 합성 데이터 생성을 어떻게 작동시킬 것인가? 그것이 연구 문제입니다.

Dwarkesh Patel *00:52:20*

Just to make sure I understood, the reason that the collapse is relevant to synthetic data generation is because you want to be able to come up with synthetic problems or reflections which are not already in your data distribution?
이해를 확실히 하자면, 붕괴가 합성 데이터 생성과 관련이 있는 이유는 당신이 이미 데이터 분포에 없는 합성 문제나 성찰을 만들 수 있기를 원하기 때문인가요?

Andrej Karpathy *00:52:32*

I guess what I'm saying is, say we have a chapter of a book and I ask an LLM to think about it, it will give you something that looks very reasonable. But if I ask it 10 times, you'll notice that all of them are the same.
제가 말하고자 하는 것은, 책의 한 장을 가지고 LLM에게 그것에 대해 생각해보라고 하면 매우 합리적으로 보이는 것을 줄 겁니다. 하지만 10번 요청하면 모두 같다는 걸 알게 될 거예요.

Dwarkesh Patel *00:52:44*

You can't just keep scaling "reflection" on the same amount of prompt information and then get returns from that.
같은 양의 프롬프트 정보에 대해 "성찰"을 계속 확장해서는 수익을 얻을 수 없다는 거군요.

Andrej Karpathy *00:52:54*

Any individual sample will look okay, but the distribution of it is quite terrible. It's quite terrible in such a way that if you continue training on too much of your own stuff, you actually collapse.
개별 샘플은 괜찮아 보이지만, 그것의 분포는 꽤 끔찍합니다. 자신의 것을 너무 많이 계속 훈련하면 실제로 붕괴할 정도로 끔찍해요.

I think that there's possibly no fundamental solution to this. I also think humans collapse over time. These analogies are surprisingly good. Humans collapse during the course of their lives. This is why children, they

haven't overfit yet. They will say stuff that will shock you because you can see where they're coming from, but it's just not the thing people say, because they're not yet collapsed. But we're collapsed. We end up revisiting the same thoughts. We end up saying more and more of the same stuff, and the learning rates go down, and the collapse continues to get worse, and then everything deteriorates.
이것에 대한 근본적인 해결책이 없을 수도 있다고 생각해요. 인간도 시간이 지나면서 붕괴한다고 생각합니다. 이런 유사성은 놀랍도록 좋아요. 인간은 삶의 과정에서 붕괴합니다. 그래서 아이들은 아직 오버피팅되지 않았죠. 그들은 당신을 충격에 빠뜨릴 말을 할 겁니다. 왜냐하면 그들이 어디서 오는지 알 수 있지만, 그것은 사람들이 하는 말이 아니거든요. 아직 붕괴되지 않았기 때문이죠. 하지만 우리는 붕괴되어 있습니다. 우리는 같은 생각을 되짚게 되고, 점점 더 같은 말을 하게 되며, 학습률이 떨어지고, 붕괴는 계속 악화되다가 모든 것이 악화됩니다.

Dwarkesh Patel *00:53:39*

Have you seen this super interesting paper that dreaming is a way of preventing this kind of overfitting and collapse? The reason dreaming is evolutionary adaptive is to put you in weird situations that are very unlike your day-to-day reality, so as to prevent this kind of overfitting.
꿈이 이런 종류의 오버피팅과 붕괴를 방지하는 방법이라는 정말 흥미로운 논문을 보신 적 있나요? 꿈이 진화적으로 적응적인 이유는 일상 현실과 매우 다른 이상한 상황에 놓이게 해서 이런 종류의 오버피팅을 방지하기 때문이라고 하더군요.

Andrej Karpathy *00:53:55*

It's an interesting idea. I do think that when you're generating things in your head and then you're attending to it, you're training on your own samples, you're training on your synthetic data. If you do it for too long, you go off-rails and you collapse way too much. You always have to seek entropy in your life. Talking to other people is a great source of entropy, and things like that. So maybe the brain has also built some internal mechanisms for increasing the amount of entropy in that process. That's an interesting idea.
흥미로운 아이디어네요. 머릿속에서 무언가를 생성하고 그것에 주의를 기울일 때, 자신의 샘플로 훈련하고 있는 거죠. 합성 데이터로 훈련하는 겁니다. 너무 오래 하면 궤도를 벗어나고 너무 많이 붕괴합니다. 삶에서 항상 엔트로피를 추구해야 합니다. 다른 사람들과 대화하는 것은 훌륭한 엔트로피 원천이죠. 그래서 아마도 뇌도 그 과정에서 엔트로피의 양을 늘리기 위한 내부 메커니즘을 구축했을 수도 있습니다. 흥미로운 아이디어예요.

Dwarkesh Patel *00:54:25*

This is a very ill-formed thought so I'll just put it out and let you react to it. The best learners that we are aware of, which are children, are extremely bad at recollecting information. In fact, at the very earliest stages of childhood, you will forget everything. You're just an amnesiac about everything that happens before a certain

year date. But you're extremely good at picking up new languages and learning from the world. Maybe there's some element of being able to see the forest for the trees.
이건 매우 미완성된 생각이지만 말씀드려보고 반응을 보고 싶네요. 우리가 아는 최고의 학습자인 아이들은 정보를 회상하는 데 극도로 서툽니다. 사실 유아기 초기 단계에서는 모든 것을 잊어버릴 거예요. 특정 연령 이전에 일어난 모든 일에 대해 완전히 기억상실증이 있죠. 하지만 새로운 언어를 습득하고 세상으로부터 배우는 데는 극도로 뛰어납니다. 아마도 숲을 보는 능력과 관련이 있을지도 모르겠네요.

Whereas if you compare it to the opposite end of the spectrum, you have LLM pre-training, where these models will literally be able to regurgitate word-for-word what is the next thing in a Wikipedia page. But their ability to learn abstract concepts really quickly, the way a child can, is much more limited. Then adults are somewhere in between, where they don't have the flexibility of childhood learning, but they can memorize facts and information in a way that is harder for kids. I don't know if there's something interesting about that spectrum.
반대편 끝을 보면 LLM 사전 훈련이 있는데, 이 모델들은 문자 그대로 위키피디아 페이지의 다음 내용을 단어 하나하나 그대로 재현할 수 있습니다. 하지만 아이가 할 수 있는 방식으로 추상적 개념을 빠르게 학습하는 능력은 훨씬 제한적이죠. 성인은 그 중간 어딘가에 있어서 아동기 학습의 유연성은 없지만 아이들보다는 사실과 정보를 암기할 수 있습니다. 이 스펙트럼에 흥미로운 점이 있는 것 같아요.

Andrej Karpathy *00:55:19*

I think there's something very interesting about that, 100%. I do think that humans have a lot more of an element, compared to LLMs, of seeing the forest for the trees. We're not actually that good at memorization, which is actually a feature. Because we're not that good at memorization, we're forced to find patterns in a more general sense.
100% 매우 흥미로운 점이 있다고 생각해요. 인간은 LLM에 비해 숲을 보는 요소가 훨씬 많다고 생각합니다. 우리는 실제로 암기에 그렇게 뛰어나지 않은데, 이것은 사실 기능이죠. 암기를 잘 못하기 때문에 더 일반적인 의미에서 패턴을 찾도록 강요받습니다.

LLMs in comparison are extremely good at memorization. They will recite passages from all these training sources. You can give them completely nonsensical data. You can hash some amount of text or something like that, you get a completely random sequence. If you train on it, even just for a single iteration or two, it can suddenly regurgitate the entire thing. It will memorize it. There's no way a person can read a single sequence of random numbers and recite it to you.
이에 비해 LLM은 암기에 극도로 뛰어납니다. 모든 훈련 소스에서 구절을 암송할 겁니다. 완전히 무의미한 데이터를 줄 수 있어요. 텍스트를 해시하거나 해서 완전히 무작위 시퀀스를 얻을 수 있죠. 단 한두 번만 반복해도 훈련하면 갑자기 전체를 재현할 수 있습니다. 암기할 거예요. 사람이 무작위 숫자의 단일 시퀀스를 읽고 그것을 암송할 수 있는 방법은 없습니다.

That's a feature, not a bug, because it forces you to only learn the generalizable components. Whereas LLMs are distracted by all the memory that they have of the pre-training documents, and it's probably very distracting to them in a certain sense. So that's why when I talk about the cognitive core, I want to remove the memory, which is what we talked about. I'd love to have them have less memory so that they have to look things up, and they only maintain the algorithms for thought, and the idea of an experiment, and all this cognitive glue of acting.
그것은 버그가 아니라 기능입니다. 왜냐하면 일반화 가능한 구성 요소만 학습하도록 강요하기 때문이죠. 반면 LLM은 사전 훈련 문서의 모든 기억에 의해 산만해지고, 어떤 의미에서는 아마도 매우 산만할 겁니다. 그래서 제가 인지적 핵심에 대해 이야기할 때, 저는 기억을 제거하고 싶다고 했죠. 그들이 물건을 찾아야 하고, 사고를 위한 알고리즘과 실험의 아이디어, 행동의 모든 인지적 접착제만 유지하도록 말이죠.

Dwarkesh Patel *00:56:36*

And this is also relevant to preventing model collapse?
이것이 모델 붕괴를 방지하는 것과도 관련이 있나요?

Andrej Karpathy *00:56:41*

Let me think. I'm not sure. It's almost like a separate axis. The models are way too good at memorization, and somehow we should remove that. People are much worse, but it's a good thing.
생각해보겠습니다. 확실하지 않네요. 거의 별개의 축 같아요. 모델들은 암기를 너무 잘하고, 어떻게든 그것을 제거해야 합니다. 사람들은 훨씬 못하지만, 그것은 좋은 일이죠.

Dwarkesh Patel *00:56:57*

What is a solution to model collapse? There are very naive things you could attempt. The distribution over logits should be wider or something. There are many naive things you could try. What ends up being the problem with the naive approaches?
모델 붕괴의 해결책은 무엇인가요? 시도할 수 있는 매우 단순한 것들이 있죠. 로짓의 분포가 더 넓어야 한다든지. 시도할 수 있는 단순한 것들이 많아요. 단순한 접근법의 문제점은 무엇인가요?

Andrej Karpathy *00:57:11*

That's a great question. You can imagine having a regularization for entropy and things like that. I guess they just don't work as well empirically because right now the models are collapsed. But I will say most of the tasks that we want from them don't actually demand diversity. That's probably the answer to what's going on.
좋은 질문이네요. 엔트로피에 대한 정규화 같은 것을 상상할 수 있겠죠. 경험적으로 잘 작동하지 않는 것

같은데, 지금 모델들이 붕괴되어 있기 때문이죠. 하지만 우리가 원하는 대부분의 작업이 실제로 다양성을 요구하지 않는다고 말하고 싶네요. 아마 그것이 무슨 일이 일어나고 있는지에 대한 답일 겁니다.

The frontier labs are trying to make the models useful. I feel like the diversity of the outputs is not so much... Number one, it's much harder to work with and evaluate and all this stuff, but maybe it's not what's capturing most of the value.
프론티어 연구소들은 모델을 유용하게 만들려고 노력하고 있어요. 출력의 다양성은 그다지... 첫째, 작업하고 평가하기가 훨씬 어렵고 이런 것들이 있지만, 아마도 대부분의 가치를 포착하는 것은 아닐 겁니다.

Dwarkesh Patel *00:57:42*

In fact, it's actively penalized. If you're super creative in RL, it's not good.
사실 적극적으로 처벌받죠. RL에서 너무 창의적이면 좋지 않아요.

Andrej Karpathy *00:57:48*

Yeah. Or maybe if you're doing a lot of writing, help from LLMs and stuff like that, it's probably bad because the models will silently give you all the same stuff. They won't explore lots of different ways of answering a question.
네. 또는 LLM의 글쓰기 도움을 많이 받고 있다면 아마 나쁠 겁니다. 왜냐하면 모델들이 조용히 모두 같은 것을 줄 테니까요. 질문에 답하는 다양한 방법을 탐색하지 않을 겁니다.

Maybe this diversity, not as many applications need it so the models don't have it. But then it's a problem at synthetic data generation time, et cetera. So we're shooting ourselves in the foot by not allowing this entropy to maintain in the model. Possibly the labs should try harder.
아마 이 다양성이 그렇게 많은 응용 프로그램에 필요하지 않아서 모델들이 그것을 가지지 않는 것 같아요. 하지만 합성 데이터 생성 시간 등에는 문제가 됩니다. 그래서 우리는 모델에서 이 엔트로피를 유지하도록 허용하지 않음으로써 스스로 발목을 잡고 있는 거죠. 아마도 연구소들이 더 노력해야 할 겁니다.

Dwarkesh Patel *00:58:17*

I think you hinted that it's a very fundamental problem, it won't be easy to solve. What's your intuition for that?
매우 근본적인 문제이고 해결하기 쉽지 않을 거라고 암시하신 것 같은데, 그에 대한 직관은 무엇인가요?

Andrej Karpathy *00:58:24*

I don’t know if it’s super fundamental. I don’t know if I intended to say that. I do think that I haven’t done these experiments, but I do think that you could probably regularize the entropy to be higher. So you’re encouraging the model to give you more and more solutions, but you don’t want it to start deviating too much from the training data. It’s going to start making up its own language. It’s going to start using words that are extremely rare, so it’s going to drift too much from the distribution.
초근본적인지는 모르겠어요. 그렇게 말하려던 건지 모르겠네요. 이런 실험을 해보지 않았지만, 엔트로피를 더 높게 정규화할 수 있을 것 같아요. 모델이 점점 더 많은 솔루션을 제공하도록 장려하는 거죠. 하지만 훈련 데이터에서 너무 많이 벗어나기 시작하는 것은 원하지 않아요. 자체 언어를 만들기 시작할 거예요. 극도로 드문 단어를 사용하기 시작해서 분포에서 너무 많이 표류할 겁니다.

So I think controlling the distribution is just tricky. It’s probably not trivial in that sense.
그래서 분포를 제어하는 것이 까다롭다고 생각해요. 그런 의미에서 아마 사소하지 않을 겁니다.

Dwarkesh Patel *00:58:58*

How many bits should the optimal core of intelligence end up being if you just had to make a guess? The thing we put on the von Neumann probes, how big does it have to be?
최적의 지능 핵심은 추측해본다면 몇 비트여야 할까요? 폰 노이만 프로브에 넣을 것, 얼마나 커야 할까요?

Andrej Karpathy *00:59:10*

It’s really interesting in the history of the field because at one point everything was very scaling-pilled in terms of like, “Oh, we’re gonna make much bigger models, trillions of parameter models.” What the models have done in size is they’ve gone up and now they’ve come down. State-of-the-art models are smaller. Even then, I think they memorized way too much. So I had a prediction a while back that I almost feel like we can get cognitive cores that are very good at even a billion parameters.
이 분야의 역사에서 정말 흥미로운데, 한때 모든 것이 스케일링에 매우 집착했었죠. "우리는 훨씬 더 큰 모델, 조 단위 매개변수 모델을 만들 거야"라고 했죠. 모델들이 크기 면에서 한 것은 올라갔다가 이제 내려왔습니다. 최첨단 모델들이 더 작아졌어요. 그럼에도 불구하고 여전히 너무 많이 암기한다고 생각해요. 그래서 저는 얼마 전에 예측했는데, 우리가 10억 개의 매개변수에서도 매우 좋은 인지적 핵심을 얻을 수 있을 것 같다고 생각해요.

If you talk to a billion parameter model, I think in 20 years, you can have a very productive conversation. It thinks and it’s a lot more like a human. But if you ask it some factual question, it might have to look it up, but it knows that it doesn’t know and it might have to look it up and it will just do all the reasonable things.
10억 매개변수 모델과 대화하면, 20년 후에는 매우 생산적인 대화를 할 수 있을 것 같아요. 생각하고 훨씬 더 인간 같죠. 하지만 어떤 사실적 질문을 하면 찾아봐야 할 수도 있지만, 모르는 것을 알고 찾아봐야 할 수도 있다는 것을 알고 모든 합리적인 일을 할 겁니다.

Dwarkesh Patel *00:59:54*

That's surprising that you think it'll take a billion parameters. Because already we have billion parameter models or a couple billion parameter models that are very intelligent.
10억 매개변수라고 생각하신다니 놀랍네요. 이미 우리는 10억 매개변수나 수십억 매개변수 모델이 매우 지능적이잖아요.

Andrej Karpathy *01:00:02*

Well, state-of-the-art models are like a trillion parameters. But they remember so much stuff.
최첨단 모델은 조 단위 매개변수 정도예요. 하지만 너무 많은 것을 기억하죠.

Dwarkesh Patel *01:00:06*

Yeah, but I'm surprised that in 10 years, given the pace… We have gpt-oss-20b. That's way better than GPT-4 original, which was a trillion plus parameters. Given that trend, I'm surprised you think in 10 years the cognitive core is still a billion parameters. I'm surprised you're not like, "Oh it's gonna be like tens of millions or millions."
네, 하지만 10년 후에, 속도를 고려하면... gpt-oss-20b가 있잖아요. 조 단위 이상의 매개변수였던 원래 GPT-4보다 훨씬 낫죠. 그 추세를 고려하면, 10년 후 인지적 핵심이 여전히 10억 매개변수라고 생각하신다니 놀라워요. "수천만 또는 수백만이 될 것"이라고 하지 않으셔서 놀랐어요.

Andrej Karpathy *01:00:30*

Here's the issue, the training data is the internet, which is really terrible. There's a huge amount of gains to be made because the internet is terrible. Even the internet, when you and I think of the internet, you're thinking of like *The Wall Street Journal*. That's not what this is. When you're looking at a pre-training dataset in the frontier lab and you look at a random internet document, it's total garbage. I don't even know how this works at all. It's some like stock tickers, symbols, it's a huge amount of slop and garbage from like all the corners of the internet. It's not like your *Wall Street Journal* article, that's extremely rare. So because the internet is so terrible, we have to build really big models to compress all that. Most of that compression is memory work instead of cognitive work.
여기 문제가 있어요. 훈련 데이터는 인터넷인데, 정말 끔찍해요. 인터넷이 끔찍하기 때문에 얻을 수 있는 엄청난 이득이 있죠. 당신과 제가 인터넷을 생각할 때, 월스트리트 저널 같은 것을 생각하죠. 그게 아니에요. 프론티어 연구소에서 사전 훈련 데이터셋을 보고 무작위 인터넷 문서를 보면, 완전히 쓰레기예요. 이게 어떻게 작동하는지도 모르겠어요. 주식 시세 기호나, 인터넷의 모든 구석에서 나온 엄청난 양의 쓸모없는 쓰레기들이죠. 월스트리트 저널 기사 같은 건 극히 드물어요. 인터넷이 너무

끔찍하기 때문에 그 모든 것을 압축하기 위해 정말 큰 모델을 만들어야 해요. 그 압축의 대부분은 인지 작업 대신 기억 작업이죠.

But what we really want is the cognitive part, delete the memory. I guess what I'm saying is that we need intelligent models to help us refine even the pre-training set to just narrow it down to the cognitive components. Then I think you get away with a much smaller model because it's a much better dataset and you could train it on it. But probably it's not trained directly on it, it's probably distilled from a much better model still.
하지만 우리가 정말 원하는 것은 인지적 부분이고, 기억은 삭제하는 거예요. 제가 말하는 것은 지능적인 모델이 우리가 사전 훈련 세트를 정제해서 인지적 구성 요소로만 좁히는 데 도움이 필요하다는 거예요. 그러면 훨씬 작은 모델로도 충분할 것 같아요. 왜냐하면 훨씬 더 나은 데이터셋이고 그것으로 훈련할 수 있으니까요. 하지만 아마도 직접 훈련하는 게 아니라 여전히 훨씬 더 나은 모델에서 증류될 겁니다.

Dwarkesh Patel *01:01:35*

But why is the distilled version still a billion?
하지만 왜 증류된 버전이 여전히 10억인가요?

Andrej Karpathy *01:01:39*

I just feel like distillation works extremely well. So almost every small model, if you have a small model, it's almost certainly distilled.
증류가 극도로 잘 작동한다고 생각해요. 거의 모든 작은 모델, 작은 모델이 있다면 거의 확실히 증류된 거예요.

Dwarkesh Patel *01:01:46*

Right, but why is the distillation in 10 years not getting below 1 billion?
맞아요, 하지만 왜 10년 후 증류가 10억 이하로 내려가지 않나요?

Andrej Karpathy *01:01:50*

Oh, you think it should be smaller than a billion? I mean, come on, right? I don't know. At some point it should take at least a billion knobs to do something interesting. You're thinking it should be even smaller?
오, 10억보다 작아야 한다고 생각하시나요? 제 말은, 그래요, 맞죠? 모르겠어요. 어느 시점에서는 흥미로운 일을 하려면 적어도 10억 개의 노브가 필요해야 한다고 생각해요. 더 작아야 한다고 생각하시나요?

Dwarkesh Patel *01:02:01*

Yeah. If you look at the trend over the last few years of just finding low-hanging fruit and going from trillion plus models to models that are literally two orders of magnitude smaller in a matter of two years and having better performance, it makes me think the sort of core of intelligence might be even way, way smaller. Plenty of room at the bottom, to paraphrase Feynman.
네. 지난 몇 년 동안 낮게 매달린 과일을 찾아서 조 단위 이상의 모델에서 문자 그대로 두 자릿수 더 작은 모델로 2년 만에 가면서 더 나은 성능을 보이는 추세를 보면, 지능의 핵심이 훨씬 더 작을 수도 있다고 생각하게 됩니다. 파인만의 말을 빌리자면, 바닥에는 충분한 공간이 있죠.

Andrej Karpathy *01:02:22*

I feel like I'm already contrarian by talking about a billion parameter cognitive core and you're outdoing me. Maybe we could get a little bit smaller. I do think that practically speaking, you want the model to have some knowledge. You don't want it to be looking up everything because then you can't think in your head. You're looking up way too much stuff all the time. Some basic curriculum needs to be there for knowledge, but it doesn't have esoteric knowledge.
저는 이미 10억 매개변수 인지 핵심에 대해 이야기함으로써 반대 입장인데, 당신이 저를 능가하고 있네요. 아마 조금 더 작을 수도 있겠죠. 실제로 말하면, 모델이 어느 정도 지식을 가지기를 원한다고 생각해요. 모든 것을 찾아보는 것을 원하지 않아요. 왜냐하면 그러면 머릿속에서 생각할 수 없으니까요. 너무 많은 것을 항상 찾아보고 있을 겁니다. 지식을 위한 기본 교육과정이 있어야 하지만, 난해한 지식은 없어야 해요.

Dwarkesh Patel *01:02:48*

We're discussing what plausibly could be the cognitive core. There's a separate question which is what will be the size of frontier models over time? I'm curious if you have predictions. We had increasing scale up to maybe GPT 4.5 and now we're seeing decreasing or plateauing scale. There are many reasons this could be going on. Do you have a prediction going forward? Will the biggest models be bigger, will they be smaller, will they be the same?
우리가 인지 핵심이 무엇일 수 있는지 논의하고 있어요. 시간이 지남에 따라 프론티어 모델의 크기가 어떻게 될지는 별개의 질문이죠. 예측이 있으신지 궁금해요. GPT 4.5까지 규모가 증가했다가 이제 규모가 줄어들거나 정체되고 있어요. 이런 일이 일어날 수 있는 많은 이유가 있죠. 앞으로 예측이 있으신가요? 가장 큰 모델이 더 커질까요, 더 작아질까요, 같을까요?

Andrej Karpathy *01:03:14*

I don't have a super strong prediction. The labs are just being practical. They have a flops budget and a cost budget. It just turns out that pre-training is not where you want to put most of your flops or your cost. That's why the models have gotten smaller. They are a bit smaller, the pre-training stage is smaller, but they make it up

in reinforcement learning, mid-training, and all this stuff that follows. They're just being practical in terms of all the stages and how you get the most bang for the buck.
초강력한 예측은 없어요. 연구소들은 실용적이에요. 플롭스 예산과 비용 예산이 있죠. 사전 훈련이 대부분의 플롭스나 비용을 투입하고 싶은 곳이 아니라는 것이 밝혀졌어요. 그래서 모델이 작아진 거죠. 조금 더 작고, 사전 훈련 단계가 작지만, 강화 학습, 중간 훈련, 그리고 뒤따르는 모든 것에서 보충합니다. 그들은 모든 단계와 최대한의 가성비를 얻는 방법에 대해 실용적이죠.

Forecasting that trend is quite hard. I do still expect that there's so much low-hanging fruit. That's my basic expectation. I have a very wide distribution here.
그 추세를 예측하는 것은 꽤 어려워요. 여전히 낮게 매달린 과일이 너무 많다고 기대해요. 그것이 제 기본 기대치예요. 여기서 매우 넓은 분포를 가지고 있어요.

Dwarkesh Patel *01:03:51*

Do you expect the low-hanging fruit to be similar in kind to the kinds of things that have been happening over the last two to five years? If I look at nanochat versus nanoGPT and the architectural tweaks you made, is that the flavor of things you expect to continue to keep happening? You're not expecting any giant paradigm shifts.
낮게 매달린 과일이 지난 2-5년 동안 일어난 종류의 일들과 유사할 것으로 예상하시나요? nanochat 대 nanoGPT를 보고 당신이 만든 아키텍처 조정을 보면, 그것이 계속 일어날 것으로 예상하는 일들의 맛인가요? 거대한 패러다임 전환을 기대하지 않으시나요.

Andrej Karpathy *01:04:11*

For the most part, yeah. I expect the datasets to get much, much better. When you look at the average datasets, they're extremely terrible. They're so bad that I don't even know how anything works. Look at the average example in the training set: factual mistakes, errors, nonsensical things. Somehow when you do it at scale, the noise washes away and you're left with some of the signal. Datasets will improve a ton.
대부분 그래요. 데이터셋이 훨씬, 훨씬 더 좋아질 것으로 예상해요. 평균 데이터셋을 보면 극도로 끔찍해요. 너무 나빠서 어떻게 작동하는지도 모르겠어요. 훈련 세트의 평균 예제를 보세요: 사실 오류, 실수, 무의미한 것들. 어떻게든 규모로 하면 노이즈가 씻겨 나가고 신호의 일부가 남습니다. 데이터셋이 엄청나게 개선될 거예요.

Everything gets better. Our hardware, all the kernels for running the hardware and maximizing what you get with the hardware. Nvidia is slowly tuning the hardware itself, Tensor Cores, all that needs to happen and will continue to happen. All the kernels will get better and utilize the chip to the max extent. All the algorithms will probably improve over optimization, architecture, and all the modeling components of how everything is done and what the algorithms are that we're even training with. I do expect that nothing dominates. Everything plus 20%. This is roughly what I've seen.

모든 것이 나아집니다. 우리의 하드웨어, 하드웨어를 실행하고 하드웨어로 얻는 것을 최대화하기 위한 모든 커널. Nvidia는 천천히 하드웨어 자체를 조정하고 있고, 텐서 코어, 이 모든 것이 일어나야 하고 계속 일어날 겁니다. 모든 커널이 더 좋아지고 칩을 최대한 활용할 거예요. 모든 알고리즘이 아마도 최적화, 아키텍처, 그리고 모든 것이 어떻게 수행되는지와 우리가 훈련하는 알고리즘이 무엇인지의 모든 모델링 구성 요소에서 개선될 겁니다. 아무것도 지배하지 않을 것으로 예상해요. 모든 것이 플러스 20%. 이것이 대략 제가 본 것입니다.

01:06:25 – AGI will blend into 2% GDP growth

Dwarkesh Patel *01:06:25*

People have proposed different ways of charting how much progress we've made towards full AGI. If you can come up with some line, then you can see where that line intersects with AGI and where that would happen on the x-axis. People have proposed it's the education level. We had a high schooler, and then they went to college with RL, and they're going to get a Ph.D.
사람들은 완전한 AGI를 향한 진전을 측정하는 다양한 방법을 제안해 왔습니다. 어떤 선을 그릴 수 있다면, 그 선이 AGI와 교차하는 지점을 찾아 x축에서 언제 일어날지 알 수 있겠죠. 어떤 사람들은 교육 수준을 제안합니다. 고등학생 수준이었다가 RL로 대학생이 되고, 박사 학위를 받게 될 거라고요.

Andrej Karpathy *01:06:44*

I don't like that one.
저는 그건 별로 마음에 들지 않네요.

Dwarkesh Patel *01:06:45*

Or they'll propose horizon length. Maybe they can do tasks that take a minute, they can do those autonomously. Then they can autonomously do tasks that take an hour, a human an hour, a human a week. How do you think about the relevant y-axis here? How should we think about how AI is making progress?
아니면 작업 수행 시간의 길이를 제안하기도 합니다. 1분 걸리는 작업을 자율적으로 수행할 수 있고, 그다음엔 사람이 1시간 걸리는 작업, 일주일 걸리는 작업을 자율적으로 할 수 있다고요. 관련된 y축을 어떻게 생각하시나요? AI가 어떻게 진전하고 있는지 어떻게 생각해야 할까요?

Andrej Karpathy *01:07:05*

I have two answers to that. Number one, I'm almost tempted to reject the question entirely because I see this as an extension of computing. Have we talked about how to chart progress in computing, or how do you chart progress in computing since the 1970s or whatever? What is the y-axis? The whole question is funny from that

perspective a little bit.
두 가지 답변이 있습니다. 첫째, 저는 이 질문 자체를 거부하고 싶은 유혹을 느낍니다. 왜냐하면 저는 이것을 컴퓨팅의 확장으로 보기 때문입니다. 1970년대 이후 컴퓨팅의 진전을 어떻게 측정할지 논의해본 적이 있나요? y축은 무엇인가요? 그런 관점에서 이 질문 전체가 좀 이상합니다.

When people talk about AI and the original AGI and how we spoke about it when OpenAI started, AGI was a system you could go to that can do any economically valuable task at human performance or better. That was the definition. I was pretty happy with that at the time. I've stuck to that definition forever, and then people have made up all kinds of other definitions. But I like that definition.
사람들이 AI와 원래의 AGI에 대해 이야기할 때, OpenAI가 시작할 때 우리가 말했던 AGI는 경제적으로 가치 있는 모든 작업을 인간 수준 이상으로 수행할 수 있는 시스템이었습니다. 그것이 정의였죠. 저는 당시 그 정의에 꽤 만족했고, 계속 그 정의를 고수해왔습니다. 그 후에 사람들이 온갖 다른 정의를 만들어냈지만, 저는 그 정의가 마음에 듭니다.

The first concession that people make all the time is they just take out all the physical stuff because we're just talking about digital knowledge work. That's a pretty major concession compared to the original definition, which was any task a human can do. I can lift things, etc. AI can't do that, obviously, but we'll take it. What fraction of the economy are we taking away by saying, "Oh, only knowledge work?" I don't know the numbers. I feel about 10% to 20%, if I had to guess, is only knowledge work, someone could work from home and perform tasks, something like that. It's still a really large market. What is the size of the economy, and what is 10% or 20%? We're still talking about a few trillion dollars, even in the US, of market share or work. So it's still a very massive bucket.
사람들이 항상 하는 첫 번째 양보는 모든 물리적인 것을 제외하는 것입니다. 디지털 지식 작업만 이야기하자는 거죠. 이건 원래 정의인 "인간이 할 수 있는 모든 작업"과 비교하면 꽤 큰 양보입니다. 저는 물건을 들 수 있고 등등. AI는 분명히 그럴 수 없지만, 받아들이겠습니다. "오, 지식 작업만"이라고 말함으로써 우리가 경제의 몇 퍼센트를 빼는 걸까요? 정확한 수치는 모르지만, 추측하자면 10%에서 20% 정도가 재택근무로 수행할 수 있는 지식 작업일 겁니다. 여전히 정말 큰 시장입니다. 경제 규모가 얼마이고, 그 중 10% 또는 20%가 얼마입니까? 미국만 해도 여전히 수조 달러의 시장 점유율이나 일자리에 대해 이야기하고 있습니다. 여전히 매우 거대한 영역입니다.

Going back to the definition, what I would be looking for is to what extent is that definition true? Are there jobs or lots of tasks? If we think of tasks as not jobs but tasks. It's difficult because the problem is society will refactor based on the tasks that make up jobs, based on what's automatable or not. Today, what jobs are replaceable by AI? A good example recently was Geoff Hinton's prediction that radiologists would not be a job anymore, and this turned out to be very wrong in a bunch of ways. Radiologists are alive and well and growing, even though computer vision is really, really good at recognizing all the different things that they have to recognize in images. It's just a messy, complicated job with a lot of surfaces and dealing with patients and all this stuff in the context of it.

정의로 돌아가서, 제가 찾을 것은 그 정의가 얼마나 실현되고 있는가입니다. 일자리가 있나요, 아니면 많은 작업들이 있나요? 작업을 일자리가 아닌 작업으로 생각한다면요. 어렵습니다. 왜냐하면 사회는 자동화 가능한 것과 그렇지 않은 것에 따라 일자리를 구성하는 작업들을 재편성할 것이기 때문입니다. 오늘날 AI로 대체 가능한 일자리는 무엇일까요? 최근의 좋은 예는 제프 힌튼이 방사선과 의사가 더 이상 직업이 아닐 거라고 예측했지만, 이것이 여러 면에서 매우 틀린 것으로 밝혀진 것입니다. 방사선과 의사들은 살아있고 건재하며 성장하고 있습니다. 컴퓨터 비전이 이미지에서 인식해야 하는 모든 것을 정말 잘 인식하는데도 말이죠. 그것은 환자를 다루고 그 맥락에서 모든 일을 처리하는 등 지저분하고 복잡한 직업입니다.

I don't know that by that definition AI has made a huge dent yet. Some of the jobs that I would be looking for have some features that make it very amenable to automation earlier than later. As an example, call center employees often come up, and I think rightly so. Call center employees have a number of simplifying properties with respect to what's automatable today. Their jobs are pretty simple. It's a sequence of tasks, and every task looks similar. You take a phone call with a person, it's 10 minutes of interaction or whatever it is, probably a bit longer. In my experience, a lot longer. You complete some task in some scheme, and you change some database entries around or something like that. So you keep repeating something over and over again, and that's your job.
그 정의로 볼 때 AI가 아직 큰 진전을 이루었다고 생각하지 않습니다. 제가 주목할 일자리들 중 일부는 현재 자동화 가능한 것에 매우 적합한 특징들을 가지고 있습니다. 예를 들어, 콜센터 직원들이 자주 언급되는데, 저는 그게 맞다고 생각합니다. 콜센터 직원들은 오늘날 자동화 가능한 것과 관련하여 여러 단순화된 속성들을 가지고 있습니다. 그들의 일은 꽤 단순합니다. 일련의 작업이고, 모든 작업이 비슷해 보입니다. 사람과 전화 통화를 하고, 10분 정도의 상호작용이거나 뭐든 간에, 아마 좀 더 길겠죠. 제 경험상 훨씬 더 깁니다. 어떤 체계에서 어떤 작업을 완료하고, 데이터베이스 항목을 변경하거나 그런 것들을 합니다. 계속 같은 일을 반복하는 게 직업입니다.

You do want to bring in the task horizon—how long it takes to perform a task—and then you want to also remove context. You're not dealing with different parts of services of companies or other customers. It's just the database, you, and a person you're serving. It's more closed, it's more understandable, it's purely digital. So I would be looking for those things.
작업을 수행하는 데 걸리는 시간인 작업 수평선도 고려하고 싶고, 맥락도 제거하고 싶습니다. 회사의 다른 부서나 다른 고객들을 다루는 게 아닙니다. 그저 데이터베이스, 당신, 그리고 당신이 서비스하는 사람뿐입니다. 더 폐쇄적이고, 더 이해 가능하며, 순전히 디지털입니다. 저는 그런 것들을 찾을 겁니다.

But even there, I'm not looking at full automation yet. I'm looking for an autonomy slider. I expect that we are not going to instantly replace people. We're going to be swapping in AIs that do 80% of the volume. They delegate 20% of the volume to humans, and humans are supervising teams of five AIs doing the call center work that's more rote. I would be looking for new interfaces or new companies that provide some layer that allows you to manage some of these AIs that are not yet perfect. Then I would expect that across the economy. A lot of jobs are a lot harder than a call center employee.

하지만 거기서도 아직 완전한 자동화를 보지 못하고 있습니다. 자율성 슬라이더를 찾고 있습니다. 우리가 즉시 사람을 대체하지는 않을 거라고 예상합니다. 80%의 볼륨을 처리하는 AI를 도입하고 있습니다. 20%의 볼륨은 인간에게 위임하고, 인간은 더 단순한 콜센터 업무를 수행하는 5개의 AI 팀을 감독합니다. 저는 이런 아직 완벽하지 않은 AI들을 관리할 수 있게 해주는 어떤 레이어를 제공하는 새로운 인터페이스나 새로운 회사들을 찾을 겁니다. 그리고 경제 전반에 걸쳐 그런 일이 일어날 것으로 예상합니다. 많은 일자리들이 콜센터 직원보다 훨씬 더 어렵습니다.

Dwarkesh Patel *01:11:02*

With radiologists, I'm totally speculating and I have no idea what the actual workflow of a radiologist involves. But one analogy that might be applicable is when Waymos were first being rolled out, there'd be a person sitting in the front seat, and you just had to have them there to make sure that if something went really wrong, they're there to monitor. Even today, people are still watching to make sure things are going well. Robotaxi, which was just deployed, still has a person inside it.
방사선과 의사의 경우, 완전히 추측이고 실제 방사선과 의사의 워크플로우가 어떤지 전혀 모르지만요. 적용 가능한 한 가지 비유는 웨이모가 처음 배치됐을 때 앞좌석에 사람이 앉아 있었던 것입니다. 정말 뭔가 잘못됐을 때를 대비해 그들이 거기 있어야 했죠. 오늘날에도 여전히 사람들이 지켜보며 일이 잘 진행되는지 확인합니다. 방금 배치된 로보택시도 여전히 안에 사람이 있습니다.

Now we could be in a similar situation where if you automate 99% of a job, that last 1% the human has to do is incredibly valuable because it's bottlenecking everything else. If it were the case with radiologists, where the person sitting in the front of Waymo has to be specially trained for years in order to provide the last 1%, their wages should go up tremendously because they're the one thing bottlenecking wide deployment. Radiologists, I think their wages have gone up for similar reasons, if you're the last bottleneck and you're not fungible. A Waymo driver might be fungible with others. So you might see this thing where your wages go up until you get to 99% and then fall just like that when the last 1% is gone. And I wonder if we're seeing similar things with radiology or salaries of call center workers or anything like that.
이제 우리는 비슷한 상황에 있을 수 있습니다. 직업의 99%를 자동화하면, 인간이 해야 하는 마지막 1%는 다른 모든 것을 병목으로 만들기 때문에 엄청나게 가치가 있습니다. 방사선과 의사의 경우, 웨이모 앞좌석에 앉아 있는 사람이 마지막 1%를 제공하기 위해 수년간 특별히 훈련받아야 한다면, 그들의 임금은 엄청나게 올라야 합니다. 왜냐하면 그들이 광범위한 배치를 가로막는 유일한 것이기 때문입니다. 방사선과 의사들의 임금이 비슷한 이유로 올랐다고 생각합니다. 당신이 마지막 병목이고 대체 불가능하다면요. 웨이모 운전자는 다른 사람들과 대체 가능할 수 있습니다. 그래서 99%에 도달할 때까지 임금이 오르다가 마지막 1%가 사라지면 그냥 떨어지는 것을 볼 수 있을 겁니다. 그리고 우리가 방사선학이나 콜센터 직원의 급여 등에서 비슷한 것을 보고 있는지 궁금합니다.

Andrej Karpathy *01:12:17*

That's an interesting question. I don't think we're currently seeing that with radiology. I think radiology is not a good example. I don't know why Geoff Hinton picked on radiology because I think it's an extremely messy, complicated profession.
흥미로운 질문입니다. 현재 방사선학에서 그런 일이 일어나고 있다고 생각하지 않습니다. 방사선학은 좋은 예가 아니라고 생각합니다. 제프 힌튼이 왜 방사선학을 선택했는지 모르겠는데, 극도로 지저분하고 복잡한 직업이기 때문입니다.

I would be a lot more interested in what's happening with call center employees today, for example, because I would expect a lot of the rote stuff to be automatable today. I don't have first-level access to it but I would be looking for trends of what's happening with the call center employees. Some of the things I would also expect is that maybe they are swapping in AI, but then I would still wait for a year or two because I would potentially expect them to pull back and rehire some of the people.
오늘날 콜센터 직원들에게 무슨 일이 일어나고 있는지 훨씬 더 관심이 있습니다. 예를 들어, 많은 단순 작업이 오늘날 자동화 가능할 것으로 예상하기 때문입니다. 1차 접근 권한은 없지만 콜센터 직원들에게 어떤 일이 일어나고 있는지 추세를 찾을 겁니다. 제가 예상하는 것 중 일부는 AI를 도입하고 있지만, 1~2년을 기다릴 것입니다. 왜냐하면 잠재적으로 그들이 철회하고 일부 사람들을 다시 고용할 것으로 예상하기 때문입니다.

Dwarkesh Patel *01:13:00*

There's been evidence that that's already been happening generally in companies that have been adopting AI, which I think is quite surprising.
이미 AI를 도입한 회사들에서 일반적으로 그런 일이 일어나고 있다는 증거가 있는데, 꽤 놀랍다고 생각합니다.

I also found what was really surprising. AGI, right? A thing which would do everything. We'll take out physical work, but it should be able to do all knowledge work. What you would have naively anticipated is that the way this progression would happen is that you take a little task that a consultant is doing, you take that out of the bucket. You take a little task that an accountant is doing, you take that out of the bucket. Then you're just doing this across all knowledge work.
또한 정말 놀라운 것을 발견했습니다. AGI, 맞죠? 모든 것을 할 수 있는 것. 물리적 작업은 제외하지만 모든 지식 작업을 할 수 있어야 합니다. 당신이 순진하게 예상했을 것은 이 진행이 일어나는 방식이 컨설턴트가 하는 작은 작업을 가져가서 버킷에서 빼는 것입니다. 회계사가 하는 작은 작업을 가져가서 버킷에서 뺍니다. 그런 다음 모든 지식 작업에서 이렇게 합니다.

But instead, if we do believe we're on the path of AGI with the current paradigm, the progression is very much not like that. It does not seem like consultants and accountants are getting huge productivity improvements. It's very much like programmers are getting more and more chiseled away at their work. If you look at the revenues

of these companies, discounting normal chat revenue—which is similar to Google or something—just looking at API revenues, it's dominated by coding. So this thing which is "general", which should be able to do any knowledge work, is just overwhelmingly doing only coding. It's a surprising way that you would expect the AGI to be deployed.
하지만 대신, 우리가 현재 패러다임으로 AGI의 길에 있다고 믿는다면, 진행은 전혀 그렇지 않습니다. 컨설턴트와 회계사가 엄청난 생산성 향상을 얻고 있는 것 같지 않습니다. 프로그래머들이 점점 더 많이 자신들의 작업을 깎아내리고 있습니다. 이 회사들의 수익을 보면, 일반 채팅 수익(구글 같은 것과 비슷한)을 제외하고 API 수익만 보면 코딩이 지배적입니다. 그래서 모든 지식 작업을 할 수 있어야 하는 이 "일반적인" 것이 압도적으로 코딩만 하고 있습니다. AGI가 배치될 것으로 예상하는 놀라운 방식입니다.

Andrej Karpathy *01:14:13*

There's an interesting point here. I do believe coding is the perfect first thing for these LLMs and agents. That's because coding has always fundamentally worked around text. It's computer terminals and text, and everything is based around text. LLMs, the way they're trained on the Internet, love text. They're perfect text processors, and there's all this data out there. It's a perfect fit.
여기 흥미로운 포인트가 있습니다. 저는 코딩이 이 LLM과 에이전트들에게 완벽한 첫 번째 것이라고 믿습니다. 왜냐하면 코딩은 항상 근본적으로 텍스트를 중심으로 작동했기 때문입니다. 컴퓨터 터미널과 텍스트, 모든 것이 텍스트 기반입니다. 인터넷에서 훈련받은 LLM들은 텍스트를 좋아합니다. 그들은 완벽한 텍스트 프로세서이고, 거기에 이 모든 데이터가 있습니다. 완벽한 조합입니다.

We also have a lot of infrastructure pre-built for handling code and text. For example, we have Visual Studio Code or your favorite IDE showing you code, and an agent can plug into that. If an agent has a diff where it made some change, we suddenly have all this code already that shows all the differences to a code base using a diff. It's almost like we've pre-built a lot of the infrastructure for code.
우리는 또한 코드와 텍스트를 처리하기 위해 미리 구축된 많은 인프라를 가지고 있습니다. 예를 들어, Visual Studio Code나 당신이 좋아하는 IDE가 코드를 보여주고, 에이전트가 그것에 연결할 수 있습니다. 에이전트가 어떤 변경을 한 diff가 있다면, 우리는 갑자기 diff를 사용하여 코드 베이스의 모든 차이를 보여주는 이 모든 코드를 이미 가지고 있습니다. 코드를 위해 많은 인프라를 미리 구축한 것과 같습니다.

Contrast that with some of the things that don't enjoy that at all. As an example, there are people trying to build automation not for coding, but for slides. I saw a company doing slides. That's much, much harder. The reason it's much harder is because slides are not text. Slides are little graphics, they're arranged spatially, and there's a visual component to it. Slides don't have this pre-built infrastructure. For example, if an agent is to make a change to your slides, how does a thing show you the diff? How do you see the diff? There's nothing that shows diffs for slides. Someone has to build it. Some of these things are not amenable to AIs as they are, which are text processors, and code surprisingly is.
전혀 그런 것을 즐기지 못하는 것들과 대조해보세요. 예를 들어, 사람들이 코딩이 아닌 슬라이드 자동화를

구축하려고 하는 것을 봤습니다. 슬라이드를 하는 회사를 봤습니다. 훨씬, 훨씬 더 어렵습니다. 훨씬 더 어려운 이유는 슬라이드가 텍스트가 아니기 때문입니다. 슬라이드는 작은 그래픽이고, 공간적으로 배열되어 있으며, 시각적 구성 요소가 있습니다. 슬라이드에는 이런 사전 구축된 인프라가 없습니다. 예를 들어, 에이전트가 슬라이드를 변경한다면, 어떤 것이 diff를 보여줍니까? diff를 어떻게 보나요? 슬라이드의 diff를 보여주는 것은 없습니다. 누군가가 그것을 구축해야 합니다. 이런 것들 중 일부는 텍스트 프로세서인 현재의 AI에 적합하지 않고, 놀랍게도 코드는 적합합니다.

Dwarkesh Patel *01:15:48*

I'm not sure that alone explains it. I personally have tried to get LLMs to be useful in domains which are just pure language-in, language-out, like rewriting transcripts, coming up with clips based on transcripts. It's very plausible that I didn't do every single possible thing I could do. I put a bunch of good examples in context, but maybe I should have done some kind of fine-tuning.
저는 그것만으로는 설명이 안 된다고 확신합니다. 저는 개인적으로 순수한 언어 입력, 언어 출력 도메인에서 LLM이 유용하도록 노력했습니다. 대본 다시 쓰기, 대본을 기반으로 클립 만들기 같은 것들이요. 제가 할 수 있는 모든 것을 다 하지 않았을 가능성이 매우 높습니다. 문맥에 좋은 예제들을 많이 넣었지만, 어떤 종류의 파인튜닝을 했어야 했을지도 모릅니다.

Our mutual friend, Andy Matuschak, told me that he tried 50 billion things to try to get models to be good at writing spaced repetition prompts. Again, very much language-in, language-out tasks, the kind of thing that should be dead center in the repertoire of these LLMs. He tried in-context learning with a few-shot examples. He tried supervised fine-tuning and retrieval. He could not get them to make cards to his satisfaction.
우리의 공통 친구인 앤디 마투샤크는 간격 반복 프롬프트를 작성하는 데 모델을 잘 만들기 위해 500억 가지를 시도했다고 말했습니다. 다시 말하지만, 매우 언어 입력, 언어 출력 작업으로, 이런 LLM의 레퍼토리에서 정중앙에 있어야 하는 종류의 것입니다. 그는 몇 샷 예제로 문맥 내 학습을 시도했습니다. 지도 파인튜닝과 검색을 시도했습니다. 그는 자신이 만족할 만한 카드를 만들 수 없었습니다.

So I find it striking that even in language-out domains, it's very hard to get a lot of economic value out of these models separate from coding. I don't know what explains it.
그래서 코딩과 별개로 언어 출력 도메인에서도 이 모델들에서 많은 경제적 가치를 얻기가 매우 어렵다는 것이 인상적입니다. 무엇이 그것을 설명하는지 모르겠습니다.

Andrej Karpathy *01:16:57*

That makes sense. I'm not saying that anything text is trivial. I do think that code is pretty structured. Text is maybe a lot more flowery, and there's a lot more entropy in text, I would say. I don't know how else to put it. Also code is hard, and so people feel quite empowered by LLMs, even from simple knowledge. I don't know that I have a very good answer. Obviously, text makes it much, much easier, but it doesn't mean that all text is

trivial.
그게 맞습니다. 텍스트인 모든 것이 사소한 것은 아니라고 말하는 것이 아닙니다. 코드는 꽤 구조화되어 있다고 생각합니다. 텍스트는 훨씬 더 화려하고, 텍스트에 훨씬 더 많은 엔트로피가 있다고 말하겠습니다. 다르게 표현할 방법이 없네요. 또한 코드는 어렵고, 사람들은 단순한 지식에서도 LLM에 의해 꽤 힘을 얻는다고 느낍니다. 저는 매우 좋은 답변이 있는지 모르겠습니다. 분명히 텍스트는 훨씬 더 쉽게 만들지만, 모든 텍스트가 사소한 것은 아닙니다.

01:17:36 – ASI

Dwarkesh Patel *01:17:36*

How do you think about superintelligence? Do you expect it to feel qualitatively different from normal humans or human companies?
초지능에 대해 어떻게 생각하시나요? 일반적인 인간이나 인간 기업과 질적으로 다르게 느껴질 거라고 예상하시나요?

Andrej Karpathy *01:17:45*

I see it as a progression of automation in society. Extrapolating the trend of computing, there will be a gradual automation of a lot of things, and superintelligence will an extrapolation of that. We expect more and more autonomous entities over time that are doing a lot of the digital work and then eventually even the physical work some amount of time later. Basically I see it as just automation, roughly speaking.
저는 이것을 사회의 자동화가 진행되는 과정으로 봅니다. 컴퓨팅의 트렌드를 외삽하면, 많은 것들이 점진적으로 자동화될 것이고, 초지능은 그것의 외삽이 될 것입니다. 시간이 지나면서 디지털 작업의 많은 부분을 수행하고, 결국에는 나중에 물리적 작업까지도 수행하는 더 많은 자율적 개체들이 생길 것으로 예상합니다. 기본적으로 저는 이것을 대략적으로 자동화로 봅니다.

Dwarkesh Patel *01:18:10*

But automation includes the things humans can already do, and superintelligence implies things humans can't do.
하지만 자동화는 인간이 이미 할 수 있는 것들을 포함하고, 초지능은 인간이 할 수 없는 것들을 암시합니다.

Andrej Karpathy *01:18:16*

But one of the things that people do is invent new things, which I would just put into the automation if that makes sense.

하지만 사람들이 하는 일 중 하나는 새로운 것을 발명하는 것인데, 저는 그것도 자동화에 포함시킬 것입니다.

Dwarkesh Patel *01:18:20*

But I guess, less abstractly and more qualitatively, do you expect something to feel like… Because this thing can either think so fast, or has so many copies, or the copies can merge back into themselves, or is much smarter, any number of advantages an AI might have, will the civilization in which these AIs exist will just feel qualitatively different from humans?
하지만 더 덜 추상적이고 더 질적으로 말하자면, 무언가가... 이것이 너무 빠르게 생각할 수 있거나, 너무 많은 복사본을 가지고 있거나, 복사본들이 다시 자신들로 합쳐질 수 있거나, 훨씬 더 똑똑하거나, AI가 가질 수 있는 여러 장점들 때문에, 이러한 AI들이 존재하는 문명이 인간과는 질적으로 다르게 느껴질까요?

Andrej Karpathy *01:18:51*

I think it will. It is fundamentally automation, but it will be extremely foreign. It will look really strange. Like you mentioned, we can run all of this on a computer cluster and much faster.
다르게 느껴질 것 같습니다. 근본적으로는 자동화이지만, 극도로 낯설 것입니다. 정말 이상하게 보일 거예요. 말씀하신 것처럼, 우리는 이 모든 것을 컴퓨터 클러스터에서 훨씬 더 빠르게 실행할 수 있습니다.

Some of the scenarios that I start to get nervous about when the world looks like that is this gradual loss of control and understanding of what's happening. I think that's the most likely outcome, that there will be a gradual loss of understanding. We'll gradually layer all this stuff everywhere, and there will be fewer and fewer people who understand it. Then there will be a gradual loss of control and understanding of what's happening. That to me seems the most likely outcome of how all this stuff will go down.
세상이 그렇게 보일 때 제가 긴장하기 시작하는 시나리오 중 일부는 점진적인 통제력 상실과 무슨 일이 일어나고 있는지에 대한 이해 상실입니다. 저는 그것이 가장 가능성 있는 결과라고 생각합니다. 점진적인 이해 상실이 있을 것입니다. 우리는 점진적으로 이 모든 것을 모든 곳에 겹겹이 쌓을 것이고, 그것을 이해하는 사람들이 점점 줄어들 것입니다. 그런 다음 무슨 일이 일어나고 있는지에 대한 점진적인 통제력과 이해의 상실이 있을 것입니다. 저에게는 그것이 이 모든 것이 어떻게 진행될지에 대한 가장 가능성 있는 결과로 보입니다.

Dwarkesh Patel *01:19:31*

Let me probe on that a bit. It's not clear to me that loss of control and loss of understanding are the same things. A board of directors at TSMC, Intel—name a random company—they're just prestigious 80-year-olds. They have very little understanding, and maybe they don't practically actually have control.
그 부분을 좀 더 파고들어 보겠습니다. 통제력 상실과 이해 상실이 같은 것인지는 명확하지 않습니다.

TSMC, 인텔 등 임의의 회사의 이사회를 보면, 그들은 그저 권위 있는 80대들입니다. 그들은 이해가 매우 부족하고, 실제로 실질적인 통제력도 없을 수 있습니다.

A better example is the President of the United States. The President has a lot of fucking power. I'm not trying to make a good statement about the current operant, or maybe I am, but the actual level of understanding is very different from the level of control.
더 나은 예는 미국 대통령입니다. 대통령은 엄청난 권력을 가지고 있습니다. 현재 집권자에 대해 좋은 말을 하려는 것은 아니지만, 아니면 그럴 수도 있지만, 실제 이해 수준은 통제 수준과 매우 다릅니다.

Andrej Karpathy *01:20:06*

I think that's fair. That's a good pushback. I think I expect loss of both.
그것은 공정한 지적입니다. 좋은 반박입니다. 저는 둘 다 상실될 것으로 예상합니다.

Dwarkesh Patel *01:20:15*

How come? Loss of understanding is obvious, but why loss of control?
어떻게요? 이해의 상실은 명백하지만, 왜 통제력의 상실인가요?

Andrej Karpathy *01:20:20*

We're really far into a territory where I don't know what this looks like, but if I were to write sci-fi novels, they would look along the lines of not even a single entity that takes over everything, but multiple competing entities that gradually become more and more autonomous. Some of them go rogue and the others fight them off. It's this hot pot of completely autonomous activity that we've delegated to. I feel it would have that flavor.
우리는 이것이 어떻게 보일지 모르는 영역에 정말 멀리 와 있지만, 제가 SF 소설을 쓴다면, 모든 것을 장악하는 단일 개체가 아니라 점점 더 자율적이 되는 여러 경쟁 개체들의 노선을 따라 보일 것입니다. 그들 중 일부는 통제를 벗어나고 다른 것들이 그들과 싸웁니다. 우리가 위임한 완전히 자율적인 활동의 뜨거운 냄비 같은 것입니다. 저는 그런 맛이 날 것 같다고 느낍니다.

Dwarkesh Patel *01:20:52*

It is not the fact that they are smarter than us that is resulting in the loss of control. It's the fact that they are competing with each other, and whatever arises out of that competition leads to the loss of control.
통제력 상실을 초래하는 것은 그들이 우리보다 똑똑하다는 사실이 아니라, 그들이 서로 경쟁하고 있고, 그 경쟁에서 발생하는 것이 무엇이든 통제력 상실로 이어진다는 사실입니까?

Andrej Karpathy *01:21:06*

A lot of these things, they will be tools to people, they're acting on behalf of people or something like that. So maybe those people are in control, but maybe it's a loss of control overall for society in the sense of outcomes we want. You have entities acting on behalf of individuals that are still roughly seen as out of control.
이런 것들 중 많은 것이 사람들의 도구가 될 것이고, 사람들을 대신해서 행동하거나 그런 것들이 될 것입니다. 그래서 아마도 그 사람들이 통제하고 있을지도 모르지만, 우리가 원하는 결과라는 의미에서 사회 전체적으로는 통제력을 잃을 수도 있습니다. 개인을 대신해서 행동하는 개체들이 있는데, 그것들은 여전히 대략적으로 통제를 벗어난 것으로 보입니다.

Dwarkesh Patel *01:21:30*

This is a question I should have asked earlier. We were talking about how currently it feels like when you're doing AI engineering or AI research, these models are more in the category of compiler rather than in the category of a replacement.
이것은 제가 더 일찍 물어봤어야 했던 질문입니다. 우리가 현재 AI 엔지니어링이나 AI 연구를 할 때, 이 모델들이 대체물이라는 범주보다는 컴파일러 범주에 더 가깝다고 느낀다는 것에 대해 이야기했었죠.

At some point, if you have AGI, it should be able to do what you do. Do you feel like having a million copies of you in parallel results in some huge speed-up of AI progress? If that does happen, do you expect to see an intelligence explosion once we have a true AGI? I'm not talking about LLMs today.
어느 시점에서, AGI가 있다면, 당신이 하는 일을 할 수 있어야 합니다. 당신의 복사본 백만 개를 병렬로 가지는 것이 AI 진보의 엄청난 속도 향상을 가져올 것이라고 느끼시나요? 그런 일이 일어난다면, 진정한 AGI를 가지게 되면 지능 폭발을 보게 될 것으로 예상하시나요? 오늘날의 LLM이 아니라 말입니다.

Andrej Karpathy *01:22:01*

I do, but it's business as usual because we're in an intelligence explosion already and have been for decades. It's basically the GDP curve that is an exponential weighted sum over so many aspects of the industry. Everything is gradually being automated and has been for hundreds of years. The Industrial Revolution is automation and some of the physical components and tool building and all this stuff. Compilers are early software automation, et cetera. We've been recursively self-improving and exploding for a long time.
저는 그럴 것이라고 생각하지만, 우리는 이미 지능 폭발 속에 있고 수십 년 동안 그래왔기 때문에 평소와 다름없습니다. 그것은 기본적으로 업계의 많은 측면에 대한 지수 가중 합계인 GDP 곡선입니다. 모든 것이 점진적으로 자동화되고 있으며 수백 년 동안 그래왔습니다. 산업 혁명은 자동화이고 물리적 구성 요소와 도구 제작 등의 일부입니다. 컴파일러는 초기 소프트웨어 자동화 등입니다. 우리는 오랫동안 재귀적으로 자기 개선하고 폭발해 왔습니다.

Another way to see it is that Earth was a pretty boring place if you don't look at the biomechanics and so on, and looked very similar. If you look from space, we're in the middle of this firecracker event, but we're seeing it in

slow motion. I definitely feel like this has already happened for a very long time. Again, I don't see AI as a distinct technology with respect to what has already been happening for a long time.
그것을 보는 또 다른 방법은 지구가 생체역학 등을 보지 않으면 꽤 지루한 곳이었고 매우 비슷해 보였다는 것입니다. 우주에서 보면 우리는 이 폭죽 이벤트의 한가운데에 있지만, 우리는 그것을 슬로우 모션으로 보고 있습니다. 저는 확실히 이것이 이미 오랫동안 일어났다고 느낍니다. 다시 말하지만, 저는 AI를 이미 오랫동안 일어나고 있는 것과 관련하여 별개의 기술로 보지 않습니다.

Dwarkesh Patel *01:23:00*

You think it's continuous with this hyper-exponential trend?
이 하이퍼 지수 트렌드와 연속적이라고 생각하시는군요?

Andrej Karpathy *01:23:03*

Yes. That's why this was very interesting to me, because I was trying to find AI in the GDP for a while. I thought that GDP should go up. But then I looked at some of the other technologies that I thought were very transformative, like computers or mobile phones or et cetera. You can't find them in GDP. GDP is the same exponential.
네. 그래서 이것이 저에게 매우 흥미로웠습니다. 한동안 GDP에서 AI를 찾으려고 노력했거든요. GDP가 올라가야 한다고 생각했습니다. 하지만 제가 매우 변혁적이라고 생각했던 다른 기술들, 컴퓨터나 휴대폰 등을 봤을 때, GDP에서 찾을 수 없습니다. GDP는 같은 지수입니다.

Even the early iPhone didn't have the App Store, and it didn't have a lot of the bells and whistles that the modern iPhone has. So even though we think of 2008, when the iPhone came out, as this major seismic change, it's actually not. Everything is so spread out and it so slowly diffuses that everything ends up being averaged up into the same exponential. It's the exact same thing with computers. You can't find them in the GDP like, "Oh, we have computers now." That's not what happened, because it's such slow progression.
초기 아이폰조차도 앱 스토어가 없었고, 현대 아이폰이 가진 많은 부가 기능들이 없었습니다. 그래서 우리가 2008년 아이폰이 나왔을 때를 이 주요한 지진적 변화로 생각하지만, 실제로는 그렇지 않습니다. 모든 것이 너무 퍼져 있고 너무 천천히 확산되어 결국 모든 것이 같은 지수로 평균화됩니다. 컴퓨터도 정확히 같은 일입니다. GDP에서 "오, 이제 우리는 컴퓨터를 가지고 있어"라고 찾을 수 없습니다. 그런 일은 일어나지 않았습니다. 너무 느린 진행이기 때문입니다.

With AI we're going to see the exact same thing. It's just more automation. It allows us to write different kinds of programs that we couldn't write before, but AI is still fundamentally a program. It's a new kind of computer and a new kind of computing system. But it has all these problems, it's going to diffuse over time, and it's still going to add up to the same exponential. We're still going to have an exponential that's going to get extremely vertical. It's going to be very foreign to live in that kind of an environment.

AI에서도 정확히 같은 것을 볼 겁니다. 더 많은 자동화일 뿐입니다. 이전에는 쓸 수 없었던 다른 종류의 프로그램을 쓸 수 있게 해주지만, AI는 여전히 근본적으로 프로그램입니다. 새로운 종류의 컴퓨터이고 새로운 종류의 컴퓨팅 시스템입니다. 하지만 이런저런 문제가 있고, 시간이 지나면서 확산될 것이고, 여전히 같은 지수함수로 합산될 겁니다. 우리는 여전히 극도로 수직적이 될 지수함수를 가질 것입니다. 그런 환경에서 사는 것은 매우 낯설 겁니다.

Dwarkesh Patel *01:24:10*

Are you saying that, if you look at the trend before the Industrial Revolution to now, you have a hyper-exponential where you go from 0% growth to then 10,000 years ago, 0.02% growth, and to now when we're at 2% growth. That's a hyper-exponential. Are you saying if you're charting AI on there, then AI takes you to 20% growth or 200% growth?
산업 혁명 이전부터 지금까지의 추세를 보면, 0% 성장에서 1만 년 전 0.02% 성장, 그리고 지금의 2% 성장으로 가는 초지수함수가 있다는 말씀이신가요? 그것이 초지수함수입니다. AI를 거기에 도표화하면 AI가 20% 성장이나 200% 성장으로 데려간다는 말씀이신가요?

Or are you saying that if you look at the last 300 years, what you've been seeing is that you have technology after technology—computers, electrification, steam engines, railways, et cetera—but the rate of growth is the exact same, it's 2%. Are you saying the rate of growth will go up?
아니면 지난 300년을 보면 기술이 계속 나타났지만(컴퓨터, 전기화, 증기기관, 철도 등) 성장률은 정확히 같은 2%라는 말씀이신가요? 성장률이 올라갈 거라고 말씀하시는 건가요?

Andrej Karpathy *01:24:46*

The rate of growth has also stayed roughly constant, right?
성장률도 대략 일정하게 유지되었죠?

Dwarkesh Patel *01:24:49*

Only over the last 200, 300 years. But over the course of human history it's exploded. It's gone from 0% to faster, faster, faster. Industrial explosion, 2%.
지난 200-300년 동안만요. 하지만 인류 역사 전체로 보면 폭발했습니다. 0%에서 점점 빨라지고, 빨라지고, 빨라지고. 산업 폭발, 2%.

Andrej Karpathy *01:25:01*

For a while I tried to find AI or look for AI in the GDP curve, and I've convinced myself that this is false. Even when people talk about recursive self-improvement and labs and stuff like that, this is business as usual. Of

course it’s going to recursively self-improve, and it’s been recursively self-improving.
한동안 GDP 곡선에서 AI를 찾거나 찾으려고 노력했는데, 이것이 거짓이라고 확신하게 되었습니다. 사람들이 재귀적 자기 개선과 연구소 등에 대해 이야기할 때조차도, 이것은 평소와 다름없습니다. 물론 재귀적으로 자기 개선할 것이고, 재귀적으로 자기 개선해왔습니다.

LLMs allow the engineers to work much more efficiently to build the next round of LLM, and a lot more of the components are being automated and tuned and et cetera. All the engineers having access to Google Search is part of it. All the engineers having an IDE, all of them having autocomplete or having Claude code, et cetera, it’s all just part of the same speed-up of the whole thing. It’s just so smooth.
LLM은 엔지니어들이 다음 라운드의 LLM을 구축하기 위해 훨씬 더 효율적으로 작업할 수 있게 해주고, 훨씬 더 많은 구성 요소가 자동화되고 조정되고 있습니다. 모든 엔지니어가 구글 검색에 접근할 수 있는 것도 그 일부입니다. 모든 엔지니어가 IDE를 가지고 있고, 자동완성을 가지고 있거나 Claude code를 가지고 있는 것 등, 이 모든 것이 전체의 속도를 높이는 같은 부분입니다. 너무 부드럽습니다.

Dwarkesh Patel *01:25:41*

Just to clarify, you’re saying that the rate of growth will not change. The intelligence explosion will show up as it just enabled us to continue staying on the 2% growth trajectory, just as the Internet helped us stay on the 2% growth trajectory.
명확히 하자면, 성장률은 변하지 않을 거라고 말씀하시는 건가요. 지능 폭발은 인터넷이 우리가 2% 성장 궤도를 유지하도록 도운 것처럼, 우리가 2% 성장 궤도를 계속 유지할 수 있게 해주는 것으로 나타날 거라는 건가요?

Andrej Karpathy *01:25:53*

Yes, my expectation is that it stays in the same pattern.
네, 제 기대는 같은 패턴을 유지한다는 것입니다.

Dwarkesh Patel *01:25:58*

Just to throw the opposite argument against you, my expectation is that it blows up because I think true AGI—and I’m not talking about LLM coding bots, I’m talking about actual replacement of a human in a server—is qualitatively different from these other productivity-improving technologies because it’s labor itself.
반대 논증을 던져보자면, 제 기대는 폭발할 거라는 것입니다. 진정한 AGI는(LLM 코딩 봇이 아니라 서버에 있는 실제 인간 대체를 말합니다) 다른 생산성 향상 기술들과는 질적으로 다르다고 생각하기 때문입니다. 그것은 노동 그 자체이기 때문입니다.

I think we live in a very labor-constrained world. If you talk to any startup founder or any person, you can be like, what do you need more of? You need really talented people. And if you have billions of extra people who are inventing stuff, integrating themselves, making companies bottom start to finish, that feels qualitatively different from a single technology. It's as if you get 10 billion extra people on the planet.
우리는 매우 노동 제약적인 세계에 살고 있다고 생각합니다. 어떤 스타트업 창업자나 어떤 사람과 이야기해보면, 무엇이 더 필요한가요? 정말 재능 있는 사람들이 필요합니다. 그리고 물건을 발명하고, 스스로를 통합하고, 처음부터 끝까지 회사를 만드는 수십억 명의 추가 사람들이 있다면, 그것은 단일 기술과는 질적으로 다른 느낌입니다. 마치 지구에 100억 명의 추가 사람이 생기는 것 같습니다.

Andrej Karpathy *01:26:44*

Maybe a counterpoint. I'm pretty willing to be convinced one way or another on this point. But I will say, for example, computing is labor. Computing was labor. Computers, a lot of jobs disappeared because computers are automating a bunch of digital information processing that you now don't need a human for. So computers are labor, and that has played out.
반론을 하자면요. 저는 이 점에 대해 어느 쪽이든 설득될 준비가 되어 있습니다. 하지만 예를 들어, 컴퓨팅은 노동입니다. 컴퓨팅은 노동이었습니다. 컴퓨터 때문에 많은 일자리가 사라졌는데, 컴퓨터가 이제 인간이 필요 없는 많은 디지털 정보 처리를 자동화하고 있기 때문입니다. 그래서 컴퓨터는 노동이고, 그것은 진행되어 왔습니다.

Self-driving as an example is also computers doing labor. That's already been playing out. It's still business as usual.
자율주행을 예로 들면 컴퓨터가 노동을 하는 것입니다. 이미 진행되고 있습니다. 여전히 평소와 다름없습니다.

Dwarkesh Patel *01:27:13*

You have a machine which is spitting out more things like that at potentially faster pace. Historically, we have examples of the growth regime changing where you went from 0.2% growth to 2% growth. It seems very plausible to me that a machine which is then spitting out the next self-driving car and the next Internet and whatever…
잠재적으로 더 빠른 속도로 그런 것들을 더 많이 뱉어내는 기계가 있습니다. 역사적으로 우리는 0.2% 성장에서 2% 성장으로 간 성장 체제 변화의 예가 있습니다. 다음 자율주행차와 다음 인터넷과 무엇이든 뱉어내는 기계가 있다는 것은 매우 그럴듯해 보입니다...

Andrej Karpathy *01:27:33*

I see where it’s coming from. At the same time, I do feel like people make this assumption of, “We have God in a box, and now it can do everything,” and it just won’t look like that. It’s going to be able to do some of the things. It’s going to fail at some other things. It’s going to be gradually put into society, and we’ll end up with the same pattern. That is my prediction.
어디서 오는지 알겠습니다. 동시에, 사람들이 "우리는 상자 안에 신을 가지고 있고, 이제 모든 것을 할 수 있다"고 가정하는데, 그렇게 보이지 않을 겁니다. 일부는 할 수 있을 것이고, 다른 일부는 실패할 것입니다. 점진적으로 사회에 투입될 것이고, 결국 같은 패턴으로 끝날 겁니다. 그것이 제 예측입니다.

This assumption of suddenly having a completely intelligent, fully flexible, fully general human in a box, and we can dispense it at arbitrary problems in society, I don’t think that we will have this discrete change. I think we’ll arrive at the same kind of gradual diffusion of this across the industry.
상자 안에 완전히 지능적이고, 완전히 유연하고, 완전히 일반적인 인간이 갑자기 있고, 우리가 사회의 임의의 문제에 그것을 배치할 수 있다는 이 가정, 저는 우리가 이런 불연속적인 변화를 가질 거라고 생각하지 않습니다. 우리는 산업 전체에 걸쳐 같은 종류의 점진적인 확산에 도달할 거라고 생각합니다.

Dwarkesh Patel *01:28:14*

It often ends up being misleading in these conversations. I don’t like to use the word intelligence in this context because intelligence implies you think there’ll be a single superintelligence sitting in a server and it’ll divine how to come up with new technologies and inventions that cause this explosion. That’s not what I’m imagining when I’m imagining 20% growth. I’m imagining that there are billions of very smart human-like minds, potentially, or that’s all that’s required.
이런 대화에서 종종 오해를 불러일으킵니다. 저는 이 맥락에서 지능이라는 단어를 사용하는 것을 좋아하지 않는데, 지능은 서버에 앉아 있는 단일 초지능이 있을 것이고 그것이 이 폭발을 일으키는 새로운 기술과 발명을 어떻게 만들어낼지 신성하게 알아낼 것이라는 것을 의미하기 때문입니다. 제가 20% 성장을 상상할 때 그런 것을 상상하는 게 아닙니다. 저는 잠재적으로 수십억 개의 매우 똑똑한 인간 같은 마음들이 있거나, 그것이 필요한 전부라고 상상합니다.

But the fact that there’s hundreds of millions of them, billions of them, each individually making new products, figuring out how to integrate themselves into the economy. If a highly experienced smart immigrant came to the country, you wouldn’t need to figure out how we integrate them in the economy. They figure it out. They could start a company, they could make inventions, or increase productivity in the world.
하지만 그들이 수억, 수십억 개 있다는 사실, 각각이 개별적으로 새로운 제품을 만들고, 경제에 스스로를 통합하는 방법을 알아낸다는 것입니다. 경험이 풍부하고 똑똑한 이민자가 나라에 왔다면, 우리가 그들을 경제에 어떻게 통합할지 알아낼 필요가 없습니다. 그들이 알아냅니다. 그들은 회사를 시작할 수 있고, 발명을 할 수 있고, 세계의 생산성을 높일 수 있습니다.

We have examples, even in the current regime, of places that have had 10-20% economic growth. If you just have a lot of people and less capital in comparison to the people, you can have Hong Kong or Shenzhen or whatever with decades of 10% plus growth. There's a lot of really smart people who are ready to make use of the resources and do this period of catch-up because we've had this discontinuity, and I think AI might be similar.
현재 체제에서도 10-20% 경제 성장을 한 곳들의 예가 있습니다. 사람이 많고 사람에 비해 자본이 적다면, 홍콩이나 선전 같은 곳에서 수십 년 동안 10% 이상의 성장을 할 수 있습니다. 자원을 활용할 준비가 된 정말 똑똑한 사람들이 많이 있고 우리가 이런 불연속성을 가졌기 때문에 이 따라잡기 기간을 하는 것이고, AI도 비슷할 수 있다고 생각합니다.

Andrej Karpathy *01:29:33*

I understand, but I still think that you're presupposing some discrete jump. There's some unlock that we're waiting to claim. And suddenly we're going to have geniuses in data centers. I still think you're presupposing some discrete jump that has no historical precedent that I can't find in any of the statistics and that I think probably won't happen.
이해하지만, 여전히 어떤 불연속적인 점프를 전제하고 있다고 생각합니다. 우리가 주장하기를 기다리는 어떤 잠금 해제가 있습니다. 그리고 갑자기 우리는 데이터 센터에 천재들을 가질 것입니다. 저는 여전히 통계에서 찾을 수 없고 아마 일어나지 않을 것이라고 생각하는, 역사적 선례가 없는 어떤 불연속적인 점프를 전제하고 있다고 생각합니다.

Dwarkesh Patel *01:29:52*

I mean, the Industrial Revolution is such a jump. You went from 0.2% growth to 2% growth. I'm just saying you'll see another jump like that.
산업 혁명이 그런 점프입니다. 0.2% 성장에서 2% 성장으로 갔습니다. 저는 그런 또 다른 점프를 볼 거라고 말하는 것뿐입니다.

Andrej Karpathy *01:30:00*

I'm a little bit suspicious, I would have to take a look. For example, some of the logs are not very good from before the Industrial Revolution. I'm a bit suspicious of it but I don't have strong opinions. You're saying that this was a singular event that was extremely magical. You're saying that maybe there's going to be another event that's going to be just like that, extremely magical. It will break the paradigm, and so on.
약간 의심스럽습니다. 살펴봐야 할 것 같습니다. 예를 들어, 산업 혁명 이전의 일부 기록은 그리 좋지 않습니다. 약간 의심스럽지만 강한 의견은 없습니다. 당신은 이것이 극도로 마법 같은 단일 사건이었다고 말하고 있습니다. 아마도 그것과 똑같이 극도로 마법 같은 또 다른 사건이 있을 거라고 말하고 있습니다. 패러다임을 깨뜨릴 것이고, 등등.

Dwarkesh Patel *01:30:23*

I actually don't think… The crucial thing with the Industrial Revolution was that it was not magical. If you just zoomed in, what you would see in 1770 or 1870 is not that there was some key invention. But at the same time, you did move the economy to a regime where the progress was much faster and the exponential 10x'd. I expect a similar thing from AI where it's not like there's going to be a single moment where we've made the crucial invention.
사실 저는... 산업 혁명의 중요한 점은 마법 같지 않았다는 것입니다. 확대해서 보면 1770년이나 1870년에 보게 될 것은 어떤 핵심 발명이 있었던 것이 아닙니다. 하지만 동시에, 진보가 훨씬 빨랐고 지수가 10배가 된 체제로 경제를 이동시켰습니다. 저는 AI에서도 비슷한 것을 기대합니다. 우리가 중요한 발명을 한 단일 순간이 있을 것 같지는 않습니다.

Andrej Karpathy *01:30:51*

It's an overhang that's being unlocked. Like maybe there's a new energy source. There's some unlock—in this case, some kind of a cognitive capacity—and there's an overhang of cognitive work to do.
잠금 해제되는 오버행이 있습니다. 새로운 에너지원이 있는 것처럼. 어떤 잠금 해제가 있습니다 - 이 경우에는 어떤 종류의 인지 능력 - 그리고 해야 할 인지 작업의 오버행이 있습니다.

Dwarkesh Patel *01:31:02*

That's right.
맞습니다.

Andrej Karpathy *01:31:03*

You're expecting that overhang to be filled by this new technology when it crosses the threshold.
당신은 그 오버행이 이 새로운 기술이 임계값을 넘을 때 채워질 것으로 기대하고 있습니다.

Dwarkesh Patel *01:31:06*

Maybe one way to think about it is throughout history, a lot of growth comes because people come up with ideas, and then people are out there doing stuff to execute those ideas and make valuable output. Through most of this time, the population has been exploding. That has been driving growth.
아마도 이렇게 생각해볼 수 있을 것 같습니다. 역사를 통틀어 많은 성장은 사람들이 아이디어를 떠올리고, 그 다음 사람들이 그 아이디어를 실행하고 가치 있는 산출물을 만들기 위해 일을 하기 때문에 일어납니다. 이 시간의 대부분 동안 인구가 폭발적으로 증가했습니다. 그것이 성장을 주도했습니다.

For the last 50 years, people have argued that growth has stagnated. The population in frontier countries has also stagnated. I think we go back to the exponential growth in population that causes hyper-exponential growth in output.
지난 50년 동안 사람들은 성장이 정체되었다고 주장해왔습니다. 선진국의 인구도 정체되었습니다. 저는 우리가 산출물의 초지수적 성장을 일으키는 인구의 지수적 성장으로 돌아간다고 생각합니다.

Andrej Karpathy *01:31:37*

It's really hard to tell. I understand that viewpoint. I don't intuitively feel that viewpoint.
정말 말하기 어렵습니다. 그 관점을 이해합니다. 직관적으로 그 관점을 느끼지는 못합니다.

01:32:50 – Evolution of intelligence & culture

Dwarkesh Patel *01:32:50*

You recommended Nick Lane's book to me. On that basis, I also found it super interesting and I interviewed him. I have some questions about thinking about intelligence and evolutionary history.
당신이 추천해주신 닉 레인의 책을 읽었어요. 덕분에 저도 매우 흥미롭게 읽었고 그를 인터뷰하기도 했습니다. 지능과 진화 역사에 대해 생각해볼 몇 가지 질문이 있습니다.

Now that you, over the last 20 years of doing AI research, you maybe have a more tangible sense of what intelligence is, what it takes to develop it. Are you more or less surprised as a result that evolution just spontaneously stumbled upon it?
지난 20년간 AI 연구를 하시면서 지능이 무엇인지, 그것을 개발하는 데 무엇이 필요한지 더 구체적으로 이해하게 되셨을 텐데요. 그 결과로 진화가 그저 우연히 지능을 발견했다는 사실이 더 놀랍게 느껴지시나요, 아니면 덜 놀랍게 느껴지시나요?

Andrej Karpathy *01:33:19*

I love Nick Lane's books. I was just listening to his podcast on the way up here. With respect to intelligence and its evolution, it's very, very recent. I am surprised that it evolved.
닉 레인의 책들을 정말 좋아합니다. 여기 오는 길에도 그의 팟캐스트를 듣고 있었어요. 지능과 그 진화에 관해서 말하자면, 정말 아주 최근의 일이죠. 저는 지능이 진화했다는 게 놀랍습니다.

I find it fascinating to think about all the worlds out there. Say there's a thousand planets like Earth and what they look like. I think Nick Lane was here talking about some of the earliest parts. He expects very similar life forms, roughly speaking, and bacteria-like things in most of them. There are a few breaks in there. The evolution of intelligence intuitively feels to me like it should be a fairly rare event.

저 밖에 있는 모든 세계들에 대해 생각해보는 게 정말 흥미로워요. 지구와 같은 행성이 천 개 있다면 그들은 어떤 모습일까요? 닉 레인이 여기서 가장 초기 부분에 대해 이야기했었죠. 그는 대부분의 행성에서 매우 유사한 생명체, 대략적으로 박테리아 같은 것들을 기대한다고 했습니다. 그 사이에 몇 가지 돌파구가 있죠. 지능의 진화는 직관적으로 꽤 드문 사건이어야 한다고 느껴집니다.

Maybe you should base it on how long something has existed. If bacteria were around for 2 billion years and nothing happened, then going to eukaryote is probably pretty hard because bacteria came up quite early in Earth's evolution or history. How long have we had animals? Maybe a couple hundred million years, multicellular animals that run around, crawl, et cetera. That's maybe 10% of Earth's lifespan. Maybe on that timescale it's not too tricky. It's still surprising to me, intuitively, that it developed. I would maybe expect just a lot of animal-like life forms doing animal-like things. The fact that you can get something that creates culture and knowledge and accumulates it is surprising to me.
아마 무언가가 얼마나 오래 존재했는지를 기준으로 판단해야 할 것 같아요. 박테리아가 20억 년 동안 존재했는데 아무 일도 일어나지 않았다면, 진핵생물로 가는 것은 아마 꽤 어려운 일이겠죠. 왜냐하면 박테리아는 지구 진화나 역사에서 꽤 초기에 나타났으니까요. 동물이 존재한 건 얼마나 됐나요? 아마 몇억 년, 돌아다니고 기어다니는 다세포 동물들 말이죠. 지구 수명의 약 10% 정도입니다. 그 시간 척도에서는 그렇게 까다롭지 않을 수도 있어요. 그래도 직관적으로는 여전히 놀랍습니다. 저는 그저 동물 같은 생명체들이 동물 같은 일을 하는 것만 기대했을 거예요. 문화와 지식을 창조하고 축적할 수 있는 무언가가 나타났다는 건 정말 놀라운 일입니다.

Dwarkesh Patel *01:34:42*

There's a couple of interesting follow-ups. If you buy the Sutton perspective that the crux of intelligence is animal intelligence… The quote he said is "If you got to the squirrel, you'd be most of the way to AGI."
몇 가지 흥미로운 후속 질문이 있습니다. 만약 지능의 핵심이 동물 지능이라는 서튼의 관점을 받아들인다면... 그가 한 말은 "다람쥐에 도달했다면 AGI까지 대부분의 길을 간 것"이라는 거였죠.

We got to squirrel intelligence right after the Cambrian explosion 600 million years ago. It seems like what instigated that was the oxygenation event 600 million years ago. But immediately the intelligence algorithm was there to make the squirrel intelligence. It's suggestive that animal intelligence was like *that*. As soon as you had the oxygen in the environment, you had the eukaryote, you could just get the algorithm. Maybe it was an accident that evolution stumbled upon it so fast, but I don't know if that suggests that at the end it's going to be quite simple.
우리는 6억 년 전 캄브리아 폭발 직후에 다람쥐 지능에 도달했습니다. 그것을 촉발한 것은 6억 년 전 산소화 사건이었던 것 같아요. 하지만 즉시 다람쥐 지능을 만들 수 있는 지능 알고리즘이 거기 있었다는 거죠. 환경에 산소가 있고 진핵생물이 있자마자 그 알고리즘을 얻을 수 있었다는 건데요. 진화가 그렇게 빨리 그것을 우연히 발견한 것일 수도 있지만, 결국 그것이 꽤 단순할 거라는 걸 시사하는지 모르겠네요.

Andrej Karpathy *01:35:31*

It’s so hard to tell with any of this stuff. You can base it a bit on how long something has existed or how long it feels like something has been bottlenecked. Nick Lane is very good about describing this very apparent bottleneck in bacteria and archaea. For two billion years, nothing happened. There’s extreme diversity of biochemistry, and yet nothing grows to become animals. Two billion years.
이런 것들 중 어떤 것이든 확실히 말하기는 정말 어렵습니다. 무언가가 얼마나 오래 존재했는지, 또는 무언가가 얼마나 오래 병목이었는지를 기준으로 판단할 수 있을 것 같아요. 닉 레인은 박테리아와 고세균의 매우 명백한 병목에 대해 잘 설명합니다. 20억 년 동안 아무 일도 일어나지 않았죠. 생화학적으로는 극도의 다양성이 있지만 아무것도 동물로 성장하지 않았습니다. 20억 년이요.

I don’t know that we’ve seen exactly that kind of an equivalent with animals and intelligence, to your point. We could also look at it with respect to how many times we think certain intelligence has individually sprung up.
동물과 지능에서 정확히 그와 같은 것을 봤는지는 모르겠습니다. 당신 말대로요. 우리는 특정 지능이 개별적으로 얼마나 많이 나타났는지를 볼 수도 있겠죠.

Dwarkesh Patel *01:36:07*

That’s a really good thing to investigate.
그건 정말 조사해볼 만한 좋은 포인트네요.

Andrej Karpathy *01:36:09*

One thought on that. There’s hominid intelligence, and then there’s bird intelligence. Ravens, etc., are extremely clever, but their brain parts are quite distinct, and we don’t have that much in common. That’s a slight indication of maybe intelligence springing up a few times. In that case, you’d expect it more frequently.
한 가지 생각은, 호미니드 지능이 있고 새의 지능이 있다는 거예요. 까마귀 등은 극도로 영리하지만 그들의 뇌 부위는 꽤 다르고, 우리와 공통점이 많지 않습니다. 그건 지능이 몇 번 나타났을 가능성을 약간 시사합니다. 그런 경우라면 더 자주 기대할 수 있겠죠.

Dwarkesh Patel *01:36:32*

A former guest, Gwern, and Carl Shulman, they’ve made a really interesting point about that. Their perspective is that the scalable algorithm which humans have and primates have, arose in birds as well, and maybe other times as well. But humans found an evolutionary niche which rewarded marginal increases in intelligence and also had a scalable brain algorithm that could achieve those increases in intelligence.
이전 게스트인 그웬과 칼 슐만이 이에 대해 정말 흥미로운 지적을 했습니다. 그들의 관점은 인간과

영장류가 가진 확장 가능한 알고리즘이 새에게도 나타났고, 아마 다른 경우에도 나타났을 거라는 거예요. 하지만 인간은 지능의 한계적 증가에 보상을 주는 진화적 틈새를 찾았고, 그 지능 증가를 달성할 수 있는 확장 가능한 뇌 알고리즘도 가지고 있었습니다.

For example, if a bird had a bigger brain, it would just collapse out of the air. It's very smart for the size of its brain, but it's not in a niche which rewards the brain getting bigger. It's maybe similar to some really smart…
예를 들어, 새가 더 큰 뇌를 가졌다면 그냥 공중에서 떨어졌을 겁니다. 뇌 크기에 비해 매우 똑똑하지만, 뇌가 더 커지는 것에 보상을 주는 틈새에 있지 않은 거죠. 어떤 똑똑한...

Andrej Karpathy

Like dolphins?
돌고래 같은?

Dwarkesh Patel

Exaclty, humans, we have hands that reward being able to learn how to do tool use. We can externalize digestion, more energy to the brain, and that kicks off the flywheel.
정확히요. 인간은 도구 사용법을 배울 수 있게 해주는 손이 있습니다. 소화를 외부화할 수 있고, 뇌에 더 많은 에너지를 보낼 수 있고, 그것이 플라이휠을 시작시킵니다.

Andrej Karpathy *01:37:28*

Also stuff to work with. I'm guessing it would be harder if I were a dolphin. How do you have fire? The universe of things you can do in water, inside water, is probably lower than what you can do on land, just chemically.
또한 작업할 수 있는 것들도 있죠. 제가 돌고래라면 더 어려웠을 것 같아요. 어떻게 불을 만들 수 있겠어요? 물 속에서 할 수 있는 일들의 우주는 아마도 육지에서 할 수 있는 것보다 작을 겁니다. 화학적으로도요.

I do agree with this viewpoint of these niches and what's being incentivized. I still find it miraculous. I would have expected things to get stuck on animals with bigger muscles. Going through intelligence is a really fascinating breaking point.
저는 이런 틈새와 무엇이 인센티브를 받는지에 대한 관점에 동의합니다. 그래도 여전히 기적적이라고 생각해요. 저는 더 큰 근육을 가진 동물에 머물 거라고 예상했을 겁니다. 지능을 통해 나아가는 것은 정말 매혹적인 돌파점입니다.

Dwarkesh Patel *01:38:02*

The way Gwern put it is the reason it was so hard is that it's a very tight line between being in a situation where something is so important to learn that it's not worth distilling the exact right circuits directly back into your DNA, versus it's not important enough to learn at all. It has to be something that incentivizes building the algorithm to learn in a lifetime.
그웬이 표현한 방식은 그것이 그렇게 어려웠던 이유는 배울 가치가 있는 것과 DNA에 직접 올바른 회로를 정제할 가치가 없는 것 사이, 그리고 전혀 배울 가치가 없는 것 사이의 매우 좁은 선이기 때문이라는 거였어요. 평생 동안 배우는 알고리즘을 구축하도록 인센티브를 주는 무언가여야 합니다.

Andrej Karpathy *01:38:28*

You have to incentivize some kind of adaptability. You want environments that are unpredictable so evolution can't bake your algorithms into your weights. A lot of animals are pre-baked in this sense. Humans have to figure it out at test time when they get born. You want these environments that change really rapidly, where you can't foresee what will work well. You create intelligence to figure it out at test time.
어떤 종류의 적응성에 인센티브를 줘야 합니다. 예측할 수 없는 환경을 원하죠. 그래서 진화가 알고리즘을 가중치에 구워 넣을 수 없는 거예요. 많은 동물들이 이런 의미에서 미리 구워져 있습니다. 인간은 태어났을 때 테스트 시간에 알아내야 합니다. 정말 빠르게 변하는 환경을 원하는데, 거기서는 무엇이 잘 작동할지 예견할 수 없어요. 테스트 시간에 알아낼 지능을 만드는 거죠.

Dwarkesh Patel *01:38:55*

Quintin Pope had this interesting blog post where he's saying the reason he doesn't expect a sharp takeoff is that humans had the sharp takeoff where 60,000 years ago we seem to have had the cognitive architectures that we have today. 10,000 years ago, agricultural revolution, modernity. What was happening in that 50,000 years? You had to build this cultural scaffold where you can accumulate knowledge over generations.
퀸틴 포프가 흥미로운 블로그 포스트를 썼는데, 그가 급격한 도약을 기대하지 않는 이유는 인간이 이미 급격한 도약을 했기 때문이라고 해요. 6만 년 전에 우리는 오늘날과 같은 인지 구조를 가진 것 같고, 1만 년 전에 농업 혁명, 그리고 근대가 왔죠. 그 5만 년 동안 무슨 일이 있었나요? 세대에 걸쳐 지식을 축적할 수 있는 문화적 발판을 구축해야 했습니다.

This is an ability that exists for free in the way we do AI training. In many cases they are literally distilled. If you retrain a model, they can be trained on each other, they can be trained on the same pre-training corpus, they don't literally have to start from scratch. There's a sense in which it took humans a long time to get this cultural loop going, but it just comes for free with the way we do LLM training.
이것은 우리가 AI 훈련을 하는 방식에서는 무료로 존재하는 능력입니다. 많은 경우 그들은 문자 그대로 정제됩니다. 모델을 재훈련하면 서로를 훈련시킬 수 있고, 같은 사전 훈련 말뭉치로 훈련될 수 있으며, 문자 그대로 처음부터 시작할 필요가 없습니다. 인간이 이 문화적 순환을 시작하는 데 오랜 시간이 걸렸지만, LLM 훈련 방식에서는 무료로 제공된다는 의미가 있습니다.

Andrej Karpathy *01:39:45*

Yes and no. Because LLMs don't really have the equivalent of culture. Maybe we're giving them way too much and incentivizing not to create it or something like that. But the invention of culture and of written record and of passing down notes between each other, I don't think there's an equivalent of that with LLMs right now. LLMs don't really have culture right now and it's one of the impediments I would say.
예이면서도 아니에요. LLM은 실제로 문화에 해당하는 것이 없거든요. 우리가 너무 많은 것을 주고 있어서 문화를 만들지 않도록 인센티브를 주는 것 같기도 해요. 하지만 문화의 발명, 문서 기록, 서로 간에 노트를 전달하는 것, 지금 LLM에는 그에 해당하는 것이 없다고 생각합니다. LLM은 지금 문화가 없고 그것이 장애물 중 하나라고 할 수 있겠죠.

Dwarkesh Patel *01:40:05*

Can you give me some sense of what LLM culture might look like?
LLM 문화가 어떤 모습일지 감을 좀 주실 수 있나요?

Andrej Karpathy *01:40:09*

In the simplest case it would be a giant scratchpad that the LLM can edit and as it's reading stuff or as it's helping out with work, it's editing the scratchpad for itself. Why can't an LLM write a book for the other LLMs? That would be cool. Why can't other LLMs read this LLM's book and be inspired by it or shocked by it or something like that? There's no equivalence for any of this stuff.
가장 간단한 경우, LLM이 편집할 수 있는 거대한 스크래치패드가 있을 거예요. 뭔가를 읽거나 작업을 도와주면서 자기 자신을 위해 스크래치패드를 편집하는 거죠. 왜 LLM이 다른 LLM을 위한 책을 쓸 수 없을까요? 그건 멋질 거예요. 왜 다른 LLM이 이 LLM의 책을 읽고 영감을 받거나 충격을 받거나 할 수 없을까요? 이런 것들에 해당하는 게 전혀 없어요.

Dwarkesh Patel *01:40:29*

Interesting. When would you expect that kind of thing to start happening? Also, multi-agent systems and a sort of independent AI civilization and culture?
흥미롭네요. 언제쯤 그런 일이 일어나기 시작할 거라고 예상하시나요? 또한 다중 에이전트 시스템과 일종의 독립적인 AI 문명과 문화는요?

Andrej Karpathy *01:40:40*

There are two powerful ideas in the realm of multi-agent that have both not been really claimed or so on. The first one I would say is culture and LLMs having a growing repertoire of knowledge for their own purposes.
다중 에이전트 영역에는 아직 실제로 구현되지 않은 두 가지 강력한 아이디어가 있습니다. 첫 번째는 문화와 LLM이 자신들의 목적을 위한 지식의 레퍼토리를 늘려가는 것입니다.

The second one looks a lot more like the powerful idea of self-play. In my mind it's extremely powerful. Evolution has a lot of competition driving intelligence and evolution. In AlphaGo more algorithmically, AlphaGo is playing against itself and that's how it learns to get really good at Go. There's no equivalent of self-playing LLMs, but I would expect that to also exist. No one has done it yet. Why can't an LLM for example, create a bunch of problems that another LLM is learning to solve? Then the LLM is always trying to serve more and more difficult problems, stuff like that.
두 번째는 셀프 플레이의 강력한 아이디어와 훨씬 더 비슷해 보입니다. 제 생각에 이건 극도로 강력해요. 진화는 지능과 진화를 이끄는 많은 경쟁을 가지고 있습니다. 알파고에서 더 알고리즘적으로, 알파고는 자기 자신과 대결하며 그렇게 해서 바둑을 정말 잘 두게 되는 거죠. 셀프 플레이하는 LLM에 해당하는 것은 없지만, 그것도 존재할 거라고 기대합니다. 아직 아무도 하지 않았어요. 예를 들어 왜 LLM이 다른 LLM이 풀어야 할 문제들을 만들 수 없을까요? 그러면 LLM은 항상 점점 더 어려운 문제를 제공하려고 노력하겠죠.

There's a bunch of ways to organize it. It's a realm of research, but I haven't seen anything that convincingly claims both of those multi-agent improvements. We're mostly in the realm of a single individual agent, but that will change. In the realm of culture also, I would also bucket organizations. We haven't seen anything like that convincingly either. That's why we're still early.
조직하는 방법은 여러 가지가 있습니다. 연구 영역이지만, 이 두 가지 다중 에이전트 개선 사항을 설득력 있게 주장하는 것을 본 적이 없습니다. 우리는 대부분 단일 개별 에이전트의 영역에 있지만 그것은 바뀔 겁니다. 문화 영역에서도 조직도 포함시킬 거예요. 그것도 설득력 있게 본 적이 없습니다. 그래서 우리가 아직 초기 단계인 거죠.

Dwarkesh Patel *01:41:53*

Can you identify the key bottleneck that's preventing this kind of collaboration between LLMs?
LLM 간의 이런 협업을 막는 핵심 병목을 식별할 수 있나요?

Andrej Karpathy *01:41:59*

Maybe the way I would put it is, some of these analogies work and they shouldn't, but somehow, remarkably, they do. A lot of the smaller models, or the dumber models, remarkably resemble a kindergarten student, or an elementary school student or high school student. Somehow, we still haven't graduated enough where this stuff can take over. My Claude Code or Codex, they still feel like this elementary-grade student. I know that they can

take PhD quizzes, but they still cognitively feel like a kindergarten or an elementary school student.
제가 표현하자면, 이런 유추들 중 일부는 작동하는데 작동하면 안 되지만, 어쨌든 놀랍게도 작동합니다. 많은 작은 모델들, 또는 멍청한 모델들은 놀랍게도 유치원생이나 초등학생, 고등학생을 닮았어요. 어쨌든 우리는 아직 이것들이 인계받을 수 있을 만큼 충분히 졸업하지 못했습니다. 제 Claude Code나 Codex는 여전히 초등학생처럼 느껴집니다. 그들이 박사 퀴즈를 풀 수 있다는 건 알지만, 인지적으로는 여전히 유치원생이나 초등학생처럼 느껴져요.

I don't think they can create culture because they're still kids. They're savant kids. They have perfect memory of all this stuff. They can convincingly create all kinds of slop that looks really good. But I still think they don't really know what they're doing and they don't really have the cognition across all these little checkboxes that we still have to collect.
그들이 문화를 창조할 수 있다고 생각하지 않아요. 왜냐하면 그들은 아직 어린아이들이니까요. 그들은 서번트 아이들이에요. 이 모든 것에 대한 완벽한 기억을 가지고 있죠. 정말 좋아 보이는 온갖 종류의 쓰레기를 설득력 있게 만들 수 있어요. 하지만 저는 여전히 그들이 자신이 하는 일을 정말로 알지 못하고, 우리가 여전히 수집해야 하는 모든 작은 체크박스에 걸쳐 인지능력을 가지지 못한다고 생각합니다.

01:42:55 - Why self driving took so long

Dwarkesh Patel *01:42:55*

You've talked about how you were at Tesla leading self-driving from 2017 to 2022. And you firsthand saw this progress from cool demos to now thousands of cars out there actually autonomously doing drives. Why did that take a decade? What was happening through that time?
2017년부터 2022년까지 테슬라에서 자율주행을 이끌었다고 말씀하셨죠. 멋진 데모에서 시작해서 지금은 수천 대의 차가 실제로 자율주행을 하는 모습을 직접 목격하셨는데요. 왜 10년이나 걸렸을까요? 그 기간 동안 무슨 일이 있었나요?

Andrej Karpathy *01:43:11*

One thing I will almost instantly push back on is that this is not even near done, in a bunch of ways that I'm going to get to. Self-driving is very interesting because it's definitely where I get a lot of my intuitions because I spent five years on it. It has this entire history where the first demos of self-driving go all the way to the 1980s. You can see a demo from CMU in 1986. There's a truck that's driving itself on roads.
먼저 바로 반박하고 싶은 게 하나 있습니다. 이게 완성에 가까운 게 아니라는 점인데요, 여러 면에서 그렇습니다. 자율주행은 정말 흥미로운 분야인데, 제가 많은 직관을 얻은 곳이기도 합니다. 5년 동안 이 일을 했으니까요. 자율주행의 역사를 보면, 최초의 데모는 1980년대까지 거슬러 올라갑니다. 1986년 카네기멜론대학교의 데모를 보면 트럭이 도로에서 스스로 운전하는 모습을 볼 수 있죠.

Fast forward. When I was joining Tesla, I had a very early demo of Waymo. It basically gave me a perfect drive in 2014 or something like that, so a perfect Waymo drive a decade ago. It took us around Palo Alto and so on because I had a friend who worked there. I thought it was very close and then it still took a long time.
시간이 흘러서, 제가 테슬라에 합류할 때쯤에는 웨이모의 초기 데모를 경험했습니다. 2014년쯤이었던 것 같은데, 기본적으로 완벽한 주행을 보여줬어요. 팔로알토를 돌아다니며 완벽하게 운전했죠. 당시에는 정말 가까워진 것 같았는데, 그런데도 오랜 시간이 걸렸습니다.

For some kinds of tasks and jobs and so on, there's a very large demo-to-product gap where the demo is very easy, but the product is very hard. It's especially the case in cases like self-driving where the cost of failure is too high. Many industries, tasks, and jobs maybe don't have that property, but when you do have that property, that definitely increases the timelines.
어떤 종류의 작업이나 직업에서는 데모와 제품 사이에 엄청난 간극이 있습니다. 데모는 아주 쉬운데 제품은 정말 어렵죠. 특히 자율주행처럼 실패 비용이 너무 높은 경우에는 더더욱 그렇습니다. 많은 산업, 작업, 직업들이 그런 특성을 갖고 있지 않을 수도 있지만, 그런 특성이 있다면 확실히 시간이 더 오래 걸립니다.

For example, in software engineering, I do think that property does exist. For a lot of vibe coding, it doesn't. But if you're writing actual production-grade code, that property should exist, because any kind of mistake leads to a security vulnerability or something like that. Millions and hundreds of millions of people's personal Social Security numbers get leaked or something like that. So in software, people should be careful, kind of like in self-driving. In self-driving, if things go wrong, you might get injured. There are worse outcomes. But in software, it's almost unbounded how terrible something could be.
예를 들어, 소프트웨어 엔지니어링에서는 그런 속성이 존재한다고 생각합니다. 하지만 많은 바이브 코딩에서는 그렇지 않습니다. 하지만 실제 프로덕션급 코드를 작성한다면 그런 속성이 존재해야 합니다. 어떤 실수든 보안 취약점 같은 것으로 이어지기 때문입니다. 수백만, 수억 명의 개인 사회보장번호가 유출되는 등의 일이 발생합니다. 따라서 소프트웨어 분야에서는 자율주행처럼 조심해야 합니다. 자율주행에서는 문제가 발생하면 부상을 입을 수도 있고, 더 심각한 결과가 발생할 수도 있습니다. 하지만 소프트웨어 분야에서는 어떤 일이 얼마나 심각해질 수 있는지 거의 무한합니다.

I do think that they share that property. What takes the long amount of time and the way to think about it is that it's a march of nines. Every single nine is a constant amount of work. Every single nine is the same amount of work. When you get a demo and something works 90% of the time, that's just the first nine. Then you need the second nine, a third nine, a fourth nine, a fifth nine. While I was at Tesla for five years or so, we went through maybe three nines or two nines. I don't know what it is, but multiple nines of iteration. There are still more nines to go.
저는 그들이 그 속성을 공유한다고 생각합니다. 오랜 시간이 걸리고, 또 생각해 보면, 그것은 9의 행진과 같습니다. 모든 9는 일정한 양의 작업입니다. 모든 9는 같은 양의 작업입니다. 데모를 받았을 때 무언가가 90%의 확률로 작동한다면, 그것은 단지 첫 번째 9일 뿐입니다. 그다음에는 두 번째 9, 세 번째 9, 네 번째 9,

다섯 번째 9가 필요합니다. 제가 테슬라에서 5년 정도 근무했을 때, 우리는 9를 세 번이나 두 번 정도 겪었습니다. 정확히 무엇인지는 모르겠지만, 여러 번의 9 반복이었습니다. 아직 9가 더 남아 있습니다.

That's why these things take so long. It's definitely formative for me, seeing something that was a demo. I'm very unimpressed by demos. Whenever I see demos of anything, I'm extremely unimpressed by that. If it's a demo that someone cooked up as a showing, it's worse. If you can interact with it, it's a bit better. But even then, you're not done. You need the actual product. It's going to face all these challenges when it comes in contact with reality and all these different pockets of behavior that need patching.
그래서 이런 작업들이 오래 걸리는 거예요. 데모였던 것을 보는 건 저에게 분명 중요한 형성 과정이에요. 데모는 정말 인상적이지 않아요. 뭐든 데모를 볼 때마다 정말 인상적이지 않아요. 누군가 보여주기용으로 만들어낸 데모라면 더 심하죠. 직접 체험해 볼 수 있다면 좀 더 나아지겠지만요. 하지만 그렇다고 해서 끝난 게 아니에요. 실제 제품이 필요하거든요. 현실과 마주하게 되면 온갖 난관에 부딪히고, 수정해야 할 다양한 행동 패턴들이 생겨날 거예요.

We're going to see all this stuff play out. It's a march of nines. Each nine is constant. Demos are encouraging. It's still a huge amount of work to do. It is a critical safety domain, unless you're doing vibe coding, which is all nice and fun and so on. That's why this also enforced my timelines from that perspective.
이 모든 일이 어떻게 전개되는지 지켜볼 겁니다. 9의 행진과 같습니다. 각 9는 일정합니다. 데모는 고무적입니다. 하지만 여전히 해야 할 일이 엄청나게 많습니다. 중요한 안전 영역입니다. 물론 바이브 코딩을 하지 않는 한 말이죠. 바이브 코딩은 즐겁고 재밌는 작업입니다. 그래서 저는 그런 관점에서 타임라인을 강화했습니다.

Dwarkesh Patel *01:46:25*

It's very interesting to hear you say that, that the safety guarantees you need from software are not dissimilar to self-driving. What people will often say is that self-driving took so long because the cost of failure is so high. A human makes a mistake on average every 400,000 miles or every seven years. If you had to release a coding agent that couldn't make a mistake for at least seven years, it would be much harder to deploy.
소프트웨어에 필요한 안전 보장이 자율주행과 크게 다르지 않다는 말씀, 정말 흥미롭네요. 사람들은 자율주행이 그렇게 오래 걸린 이유는 실패 비용이 너무 크기 때문이라고 종종 말하곤 합니다. 사람은 평균적으로 40만 마일(약 64만 킬로미터)마다, 즉 7년마다 실수를 합니다. 최소 7년 동안 실수를 저지르지 않는 코딩 에이전트를 출시해야 한다면, 배포하기가 훨씬 더 어려워질 것입니다.

But your point is that if you made a catastrophic coding mistake, like breaking some important system every seven years…
하지만 당신이 말하고자 하는 것은 만약 7년마다 중요한 시스템을 망가뜨리는 것과 같은 치명적인 코딩 실수를 저질렀다면...

Andrej Karpathy ***01:46:56***

Very easy to do.
매우 쉽게 할 수 있어요.

Dwarkesh Patel ***01:46:57***

In fact, in terms of wall clock time, it would be much less than seven years because you're constantly outputting code like that. In terms of tokens, it would be seven years. But in terms of wall clock time...

Andrej Karpathy ***01:47:09***

In some ways, it's a much harder problem. Self-driving is just one of thousands of things that people do. It's almost like a single vertical, I suppose. Whereas when we're talking about general software engineering, it's even more... There's more surface area.

Dwarkesh Patel ***01:47:20***

There's another objection people make to that analogy, which is that with self-driving, what took a big fraction of that time was solving the problem of having basic perception that's robust, building representations, and having a model that has some common sense so it can generalize to when it sees something that's slightly out of distribution. If somebody's waving down the road this way, you don't need to train for it. The thing will have some understanding of how to respond to something like that.

These are things we're getting for free with LLMs or VLMs today, so we don't have to solve these very basic representation problems. So now

deploying AIs across different domains will sort of be like deploying a self-driving car with current models to a different city, which is hard but not like a 10-year-long task.

Andrej Karpathy ***01:48:07***

I'm not 100% sure if I fully agree with that. I don't know how much we're getting for free. There's still a lot of gaps in understanding what we are getting. We're definitely getting more generalizable intelligence in a single entity, whereas self-driving is a very special-purpose task that requires. In some sense building a special-purpose task is maybe even harder in a certain sense because it doesn't fall out from a more general thing that you're doing at scale, if that makes sense.

But the analogy still doesn't fully resonate because the LLMs are still pretty fallible and they have a lot of gaps that still need to be filled in. I don't think that we're getting magical generalization completely out of the box, in a certain sense.

The other aspect that I wanted to return to is that self-driving cars are nowhere near done still. The deployments are pretty minimal. Even Waymo and so on has very few cars. They're doing that roughly speaking because they're not economical. They've built something that lives in the future. They've had to pull back the future, but they had to make it uneconomical. There are all these costs, not just marginal costs for those cars and their operation and maintenance, but also the capex of the entire thing. Making it economical is still going to be a slog for them.

Also, when you look at these cars and there's no one driving, I actually think it's a little bit deceiving because there are very elaborate teleoperation centers of people kind of in a loop with these cars. I don't have the full extent of it, but there's more human-in-the-loop than you might expect. There are people somewhere out there beaming in from the sky. I don't know if they're fully in the loop with the driving. Some of the time they are, but they're certainly involved and there are people. In some sense, we haven't actually removed the person, we've moved them to somewhere where you can't see them.

I still think there will be some work, as you mentioned, going from environment to environment. There are still challenges to make self-driving real. But I do agree that it's definitely crossed a threshold where it kind of feels real, unless it's really teleoperated. For example, Waymo can't go to all the different parts of the city. My suspicion is that it's parts of the city where you don't get good signal. Anyway, I don't know anything about the stack. I'm just making stuff up.

Dwarkesh Patel ***01:50:23***

You led self-driving for five years at Tesla.

Andrej Karpathy ***01:50:27***

Sorry, I don't know anything about the specifics of Waymo. By the way, I love Waymo and I take it all the time. I just think that people are sometimes a little bit too naive about some of the progress and there's still a huge amount of work. Tesla took in my mind a much more scalable approach and the team is doing extremely well. I'm kind of on the record for predicting how this

thing will go. Waymo had an early start because you can package up so many sensors. But I do think Tesla is taking the more scalable strategy and it's going to look a lot more like that. So this will still have to play out and hasn't. But I don't want to talk about self-driving as something that took a decade because it didn't take it yet, if that makes sense.

Dwarkesh Patel ***01:51:08***

Because one, the start is at 1980 and not 10 years ago, and then two, the end is not here yet.

Andrej Karpathy ***01:51:14***

The end is not near yet because when we're talking about self-driving, usually in my mind it's self-driving at scale. People don't have to get a driver's license, etc.

Dwarkesh Patel ***01:51:22***

I'm curious to bounce two other ways in which the analogy might be different. The reason I'm especially curious about this is because the question of how fast AI is deployed, how valuable it is when it's early on is potentially the most important question in the world right now. If you're trying to model what the year 2030 looks like, this is the question you ought to have some understanding of.

Another thing you might think is one, you have this latency requirement with self-driving. I have no idea what the actual models are, but I assume it's like tens of millions of parameters or something, which is not the necessary

constraint for knowledge work with LLMs. Maybe it might be with computer use and stuff.

But the other big one is, maybe more importantly, on this capex question. Yes, there is additional cost to serving up an additional copy of a model, but the opex of a session is quite low and you can amortize the cost of AI into the training run itself, depending on how inference scaling goes and stuff. But it's certainly not as much as building a whole new car to serve another instance of a model. So the economics of deploying more widely are much more favorable.

Andrej Karpathy ***01:52:37***

I think that's right. If you're sticking to the realm of bits, bits are a million times easier than anything that touches the physical world. I definitely grant that. Bits are completely changeable, arbitrarily reshuffleable at a very rapid speed. You would expect a much faster adaptation also in the industry and so on. What was the first one?

Dwarkesh Patel ***01:52:59***

The latency requirements and its implications for model size?

Andrej Karpathy ***01:53:02***

I think that's roughly right. I also think that if we are talking about knowledge work at scale, there will be some latency requirements, practically speaking, because we're going to have to create a huge amount of compute and serve that.

The last aspect that I very briefly want to also talk about is all the rest of it. What does society think about it? What are the legal ramifications? How is it working legally? How is it working insurance-wise? What are those layers of it and aspects of it? What is the equivalent of people putting a cone on a Waymo? There are going to be equivalents of all that. So I feel like self-driving is a very nice analogy that you can borrow things from. What is the equivalent of a cone in the car? What is the equivalent of a teleoperating worker who's hidden away and all the aspects of it.

Dwarkesh Patel ***01:53:53***

Do you have any opinions on what this implies about the current AI buildout, which would 10x the amount of available compute in the world in a year or two and maybe more than 100x it by the end of the decade. If the use of AI will be lower than some people naively predict, does that mean that we're overbuilding compute or is that a separate question?

Andrej Karpathy ***01:54:15***

Kind of like what happened with railroads.

Dwarkesh Patel ***01:54:18***

With what, sorry?

Andrej Karpathy ***01:54:19***

Was it railroads or?

Dwarkesh Patel ***01:54:20***

Yeah, it was.

Andrej Karpathy ***01:54:21***

Yeah. There's historical precedent. Or was it with the telecommunication industry? Pre-paving the internet that only came a decade later and creating a whole bubble in the telecommunications industry in the late '90s.

I understand I'm sounding very pessimistic here. I'm actually optimistic. I think this will work. I think it's tractable. I'm only sounding pessimistic because when I go on my Twitter timeline, I see all this stuff that makes no sense to me. There's a lot of reasons for why that exists. A lot of it is honestly just fundraising. It's just incentive structures. A lot of it may be fundraising. A lot of it is just attention, converting attention to money on the internet, stuff like that. There's a lot of that going on, and I'm only reacting to that.

But I'm still overall very bullish on technology. We're going to work through all this stuff. There's been a rapid amount of progress. I don't know that there's overbuilding. I think we're going to be able to gobble up what, in my understanding, is being built. For example, Claude Code or OpenAI Codex and stuff like that didn't even exist a year ago. Is that right? This is a miraculous technology that didn't exist. There's going to be a huge amount of demand, as we see the demand in ChatGPT already and so on.

So I don't know that there's overbuilding. I'm just reacting to some of the very fast timelines that people continue to say incorrectly. I've heard many, many times over the course of my 15 years in AI where very reputable

people keep getting this wrong all the time. I want this to be properly calibrated, and some of this also has geopolitical ramifications and things like that with some of these questions. I don't want people to make mistakes in that sphere of things. I do want us to be grounded in the reality of what technology is and isn't.

01:56:20 - Future of education

Dwarkesh Patel ***01:56:20***

Let's talk about education and Eureka. One thing you could do is start another AI lab and then try to solve those problems. I'm curious what you're up to now, and why not AI research itself?

Andrej Karpathy ***01:56:33***

I guess the way I would put it is I feel some amount of determinism around the things that AI labs are doing. I feel like I could help out there, but I don't know that I would uniquely improve it. My personal big fear is that a lot of this stuff happens on the side of humanity, and that humanity gets disempowered by it. I care not just about all the Dyson spheres that we're going to build and that AI is going to build in a fully autonomous way, I care about what happens to humans. I want humans to be well off in the future.

I feel like that's where I can a lot more uniquely add value than an incremental improvement in the frontier lab. I'm most afraid of something depicted in movies like *WALL-E* or *Idiocracy* or something like that, where humanity is on the side of this stuff. I want humans to be much, much better in this future. To me, this is through education that you can achieve this.

Dwarkesh Patel ***01:57:35***

So what are you working on there?

Andrej Karpathy ***01:57:36***

The easiest way I can describe it is we're trying to build the Starfleet Academy. I don't know if you've watched *Star Trek*.

Dwarkesh Patel ***01:57:44***

I haven't.

Andrej Karpathy ***01:57:44***

Starfleet Academy is this elite institution for frontier technology, building spaceships, and graduating cadets to be the pilots of these spaceships and whatnot. So I just imagine an elite institution for technical knowledge and a kind of school that's very up-to-date and a premier institution.

Dwarkesh Patel ***01:58:05***

A category of questions I have for you is explaining how one teaches technical or scientific content well, because you are one of the world masters at it. I'm curious both about how you think about it for content you've already put out there on YouTube, but also, to the extent it's any different, how you think about it for Eureka.

Andrej Karpathy ***01:58:25***

With respect to Eureka, one thing that is very fascinating to me about education is that I do think education will pretty fundamentally change with AIs on the side. It has to be rewired and changed to some extent.

I still think that we're pretty early. There's going to be a lot of people who are going to try to do the obvious things. Have an LLM and ask it questions. Do all the basic things that you would do via prompting right now. It's helpful, but it still feels to me a bit like slop. I'd like to do it properly, and I think the capability is not there for what I would want. What I'd want is an actual tutor experience.

A prominent example in my mind is I was recently learning Korean, so language learning. I went through a phase where I was learning Korean by myself on the internet. I went through a phase where I was part of a small class in Korea taking Korean with a bunch of other people, which was really funny. We had a teacher and 10 people or so taking Korean. Then I switched to a one-on-one tutor.

I guess what was fascinating to me was, I think I had a really good tutor, but just thinking through what this tutor was doing for me and how incredible that experience was and how high the bar is for what I want to build eventually. Instantly from a very short conversation, she understood where I am as a student, what I know and don't know. She was able to probe exactly the kinds of questions or things to understand my world model. No LLM will do that for you 100% right now, not even close. But a tutor will do that if they're good. Once she understands, she really served me all the things that I needed at my current sliver of capability. I need to be always appropriately

challenged. I can't be faced with something too hard or too trivial, and a tutor is really good at serving you just the right stuff.

I felt like I was the only constraint to learning. I was always given the perfect information. I'm the only constraint. I felt good because I'm the only impediment that exists. It's not that I can't find knowledge or that it's not properly explained or etc. It's just my ability to memorize and so on. This is what I want for people.

Dwarkesh Patel ***02:00:27***

How do you automate that?

Andrej Karpathy ***02:00:29***

Very good question. At the current capability, you don't. That's why I think it's not actually the right time to build this kind of an AI tutor. I still think it's a useful product, and lots of people will build it, but the bar is so high and the capability is not there. Even today, I would say ChatGPT is an extremely valuable educational product. But for me, it was so fascinating to see how high the bar is. When I was with her, I almost felt like there's no way I can build this.

Dwarkesh Patel ***02:01:02***

But you are building it, right?

Andrej Karpathy ***02:01:03***

Anyone who's had a really good tutor is like, "How are you going to build this?" I'm waiting for that capability.

I did some AI consulting for computer vision. A lot of times, the value that I brought to the company was telling them not to use AI. I was the AI expert, and they described the problem, and I said, "Don't use AI." This is my value add. I feel like it's the same in education right now, where I feel like for what I have in mind, it's not yet the time, but the time will come. For now, I'm building something that looks maybe a bit more conventional that has a physical and digital component and so on. But it's obvious how this should look in the future.

Dwarkesh Patel ***02:01:43***

To the extent you're willing to say, what is the thing you hope will be released this year or next year?

Andrej Karpathy ***02:01:49***

I'm building the first course. I want to have a really, really good course, the obvious state-of-the-art destination you go to to learn, AI in this case. That's just what I'm familiar with, so it's a really good first product to get to be really good at it. So that's what I'm building. Nanochat, which you briefly mentioned, is a capstone project of LLM101N, which is a class that I'm building. That's a really big piece of it. But now I have to build out a lot of the intermediates, and then I have to hire a small team of TAs and so on and build the entire course.

One more thing that I would say is that many times, when people think about education, they think more about what I would say is a softer component of diffusing knowledge. I have something very hard and technical in mind. In my mind, education is the very difficult technical process of building ramps to knowledge. In my mind, nanochat is a ramp to knowledge because it's very simple. It's the super simplified full-stack thing. If you give this artifact to someone and they look through it, they're learning a ton of stuff. It's giving you a lot of what I call eurekas per second, which is understanding per second. That's what I want, lots of eurekas per second. So to me, this is a technical problem of how do we build these ramps to knowledge.

So I almost think of Eureka as maybe not that different from some of the frontier labs or some of the work that's going on there. I want to figure out how to build these ramps very efficiently so that people are never stuck and everything is always not too hard or not too trivial, and you have just the right material to progress.

Dwarkesh Patel ***02:03:25***

You're imagining in the short term that instead of a tutor being able to probe your understanding, if you have enough self-awareness to be able to probe yourself, you're never going to be stuck. You can find the right answer between talking to the TA or talking to an LLM and looking at the reference implementation. It sounds like automation or AI is not a significant part. So far, the big alpha here is your ability to explain AI codified in the source material of the class. That's fundamentally what the course is.

Andrej Karpathy ***02:04:00***

You always have to be calibrated to what capability exists in the industry. A lot of people are going to pursue just asking ChatGPT, etc. But I think right now, for example, if you go to ChatGPT and you say, teach me AI, there's no way. It's going to give you some slop. AI is never going to write nanochat right now. But nanochat is a really useful intermediate point. I'm collaborating with AI to create all this material, so AI is still fundamentally very helpful.

Earlier on, I built CS231n at Stanford, which I think was the first deep learning class at Stanford, which became very popular. The difference in building out 231n then and LLM101N now is quite stark. I feel really empowered by the LLMs as they exist right now, but I'm very much in the loop. They're helping me build the materials, I go much faster. They're doing a lot of the boring stuff, etc. I feel like I'm developing the course much faster, and it's LLM-infused, but it's not yet at a place where it can creatively create the content. I'm still there to do that. The trickiness is always calibrating yourself to what exists.

Dwarkesh Patel ***02:05:04***

When you imagine what is available through Eureka in a couple of years, it seems like the big bottleneck is going to be finding Karpathys in field after field who can convert their understanding into these ramps.

Andrej Karpathy ***02:05:18***

It would change over time. Right now, it would be hiring faculty to help work hand-in-hand with AI and a team of people probably to build state-of-the-art

courses. Over time maybe some of the TAs can become AIs. You just take all the course materials and then I think you could serve a very good automated TA for the student when they have more basic questions or something like that. But I think you'll need faculty for the overall architecture of a course and making sure that it fits. So I see a progression of how this will evolve. Maybe at some future point I'm not even that useful and AI is doing most of the design much better than I could. But I still think that's going to take some time to play out.

Dwarkesh Patel ***02:05:59***

Are you imagining that people who have expertise in other fields are then contributing courses, or do you feel like it's quite essential to the vision that you, given your understanding of how you want to teach, are the one designing the content? Sal Khan is narrating all the videos on Khan Academy. Are you imagining something like that?

Andrej Karpathy ***02:06:20***

No, I will hire faculty because there are domains in which I'm not an expert. That's the only way to offer the state-of-the-art experience for the student ultimately. I do expect that I would hire faculty, but I will probably stick around in AI for some time. I do have something more conventional in mind for the current capability than what people would probably anticipate.

When I'm building Starfleet Academy, I do probably imagine a physical institution, and maybe a tier below that a digital offering that is not the state-of-the-art experience you would get when someone comes in physically

full-time and we work through material from start to end and make sure you understand it. That's the physical offering. The digital offering is a bunch of stuff on the internet and maybe some LLM assistant. It's a bit more gimmicky in a tier below, but at least it's accessible to 8 billion people.

Dwarkesh Patel ***02:07:08***

I think you're basically inventing college from first principles for the tools that are available today and just selecting for people who have the motivation and the interest of really engaging with material.

Andrej Karpathy ***02:07:26***

There's going to have to be a lot of not just education but also re-education. I would love to help out there because the jobs will probably change quite a bit. For example, today a lot of people are trying to upskill in AI specifically. I think it's a really good course to teach in this respect. Motivation-wise, before AGI motivation is very simple to solve because people want to make money. This is how you make money in the industry today. Post-AGI is a lot more interesting possibly because if everything is automated and there's nothing to do for anyone, why would anyone go to a school?

I often say that pre-AGI education is useful. Post-AGI education is fun. In a similar way, people go to the gym today. We don't need their physical strength to manipulate heavy objects because we have machines that do that. They still go to the gym. Why do they go to the gym? Because it's fun, it's healthy, and you look hot when you have a six-pack. It's attractive for people to do that in a very deep, psychological, evolutionary sense for humanity.

Education will play out in the same way. You'll go to school like you go to the gym.

Right now, not that many people learn because learning is hard. You bounce from material. Some people overcome that barrier, but for most people, it's hard. It's a technical problem to solve. It's a technical problem to do what my tutor did for me when I was learning Korean. It's tractable and buildable, and someone should build it. It's going to make learning anything trivial and desirable, and people will do it for fun because it's trivial. If I had a tutor like that for any arbitrary piece of knowledge, it's going to be so much easier to learn anything, and people will do it. They'll do it for the same reasons they go to the gym.

Dwarkesh Patel ***02:09:17***

That sounds different from using… So post-AGI, you're using this as entertainment or as self-betterment. But it sounded like you had a vision also that this education is relevant to keeping humanity in control of AI. That sounds different. Is it entertaining for some people, but then empowerment for some others? How do you think about that?

Andrej Karpathy ***02:09:41***

I do think eventually it's a bit of a losing game, if that makes sense. It is in the long term. In the long term, which is longer than maybe most people in the industry think about, it's a losing game. I do think people can go so far and we've barely scratched the surface of how much a person can go. That's just because people are bouncing off of material that's too easy or too hard.

People will be able to go much further. Anyone will speak five languages because why not? Because it's so trivial. Anyone will know all the basic curriculum of undergrad, et cetera.

Dwarkesh Patel ***02:10:18***

Now that I'm understanding the vision, that's very interesting. It has a perfect analog in gym culture. I don't think 100 years ago anybody would be ripped. Nobody would have been able to just spontaneously bench two plates or three plates or something. It's very common now because of this idea of systematically training and lifting weights in the gym, or systematically training to be able to run a marathon, which is a capability most humans would not spontaneously have. You're imagining similar things for learning across many different domains, much more intensely, deeply, faster.

Andrej Karpathy ***02:10:54***

Exactly. I am betting a bit implicitly on some of the timelessness of human nature. It will be desirable to do all these things, and I think people will look up to it as they have for millennia. This will continue to be true. There's some evidence of that historically. If you look at, for example, aristocrats, or you look at ancient Greece or something like that, whenever you had little pocket environments that were post-AGI in a certain sense, people have spent a lot of their time flourishing in a certain way, either physically or cognitively. I feel okay about the prospects of that.

If this is false and I'm wrong and we end up in a *WALL-E* or *Idiocracy* future, then I don't even care if there are Dyson spheres. This is a terrible outcome. I

really do care about humanity. Everyone has to just be superhuman in a certain sense.

Dwarkesh Patel ***02:11:52***

It's still a world in which that is not enabling us to… It's like the culture world, right? You're not fundamentally going to be able to transform the trajectory of technology or influence decisions by your own labor or cognition alone. Maybe you can influence decisions because the AI is asking for your approval, but it's not because I've invented something or I've come up with a new design that I'm really influencing the future.

Andrej Karpathy ***02:12:21***

Maybe. I think there will be a transitional period where we are going to be able to be in the loop and advance things if we understand a lot of stuff. In the long-term, that probably goes away. It might even become a sport. Right now you have powerlifters who go extreme in this direction. What is powerlifting in a cognitive era? Maybe it's people who are really trying to make Olympics out of knowing stuff. If you have a perfect AI tutor, maybe you can get extremely far. I feel that the geniuses of today are barely scratching the surface of what a human mind can do, I think.

Dwarkesh Patel ***02:12:59***

I love this vision. I also feel like the person you have the most product-market fit with is me because my job involves having to learn different subjects every week, and I am very excited.

Andrej Karpathy ***02:13:17***

I'm similar, for that matter. A lot of people, for example, hate school and want to get out of it. I really liked school. I loved learning things, et cetera. I wanted to stay in school. I stayed all the way until Ph.D. and then they wouldn't let me stay longer, so I went to the industry. Roughly speaking, I love learning, even for the sake of learning, but I also love learning because it's a form of empowerment and being useful and productive.

Dwarkesh Patel ***02:13:39***

You also made a point that was subtle and I want to spell it out. With what's happened so far with online courses, why haven't they already enabled us to enable every single human to know everything? They're just so motivation-laden because there are no obvious on-ramps and it's so easy to get stuck. If you had this thing instead—like a really good human tutor—it would just be such an unlock from a motivation perspective.

Andrej Karpathy ***02:14:10***

I think so. It feels bad to bounce from material. It feels bad. You get negative reward from sinking an amount of time in something and it doesn't pan out, or being completely bored because what you're getting is too easy or too hard. When you do it properly, learning feels good. It's a technical problem to get there. For a while, it's going to be AI plus human collab, and at some point, maybe it's just AI.

Dwarkesh Patel ***02:14:36***

Can I ask some questions about teaching well? If you had to give advice to another educator in another field that you're curious about to make the kinds of YouTube tutorials you've made. Maybe it might be especially interesting to talk about domains where you can't test someone's technical understanding by having them code something up or something. What advice would you give them?

Andrej Karpathy ***02:14:58***

That's a pretty broad topic. There are 10–20 tips and tricks that I semi-consciously do probably. But a lot of this comes from my physics background. I really, really did enjoy my physics background. I have a whole rant on how everyone should learn physics in early school education because early school education is not about accumulating knowledge or memory for tasks later in the industry. It's about booting up a brain. Physics uniquely boots up the brain the best because some of the things that they get you to do in your brain during physics is extremely valuable later.

The idea of building models and abstractions and understanding that there's a first-order approximation that describes most of the system, but then there're second-order, third-order, fourth-order terms that may or may not be present. The idea that you're observing a very noisy system, but there are these fundamental frequencies that you can abstract away. When a physicist walks into the class and they say, "Assume there's a spherical cow," everyone laughs at that, but this is brilliant. It's brilliant thinking that's very generalizable across the industry because a cow can be approximated as a sphere in a bunch of ways.

There's a really good book, for example, *Scale*. It's from a physicist talking about biology. Maybe this is also a book I would recommend reading. You can get a lot of really interesting approximations and chart scaling laws of animals. You can look at their heartbeats and things like that, and they line up with the size of the animal and things like that. You can talk about an animal as a volume. You can talk about the heat dissipation of that, because your heat dissipation grows as the surface area, which is growing as a square. But your heat creation or generation is growing as a cube. So I just feel like physicists have all the right cognitive tools to approach problem solving in the world.

So because of that training, I always try to find the first-order terms or the second-order terms of everything. When I'm observing a system or a thing, I have a tangle of a web of ideas or knowledge in my mind. I'm trying to find, what is the thing that matters? What is the first-order component? How can I simplify it? How can I have a simplest thing that shows that thing, shows it in action, and then I can tack on the other terms?

Maybe an example from one of my repos that I think illustrates it well is called micrograd. I don't know if you're familiar with this. So micrograd is 100 lines of code that shows backpropagation. You can create neural networks out of simple operations like plus and times, et cetera. Lego blocks of neural networks. You build up a computational graph and you do a forward pass and a backward pass to get the gradients. Now, this is at the heart of all neural network learning.

So micrograd is a 100 lines of pretty interpretable Python code, and it can do forward and backward arbitrary neural networks, but not efficiently. So micrograd, these 100 lines of Python, are everything you need to understand how neural networks train. Everything else is just efficiency. Everything else is efficiency. There's a huge amount of work to get efficiency. You need your tensors, you lay them out, you stride them, you make sure your kernels, orchestrating memory movement correctly, et cetera. It's all just efficiency, roughly speaking. But the core intellectual piece of neural network training is micrograd. It's 100 lines. You can easily understand it. It's a recursive application of chain rule to derive the gradient, which allows you to optimize any arbitrary differentiable function.

So I love finding these small-order terms and serving them on a platter and discovering them. I feel like education is the most intellectually interesting thing because you have a tangle of understanding and you're trying to lay it out in a way that creates a ramp where everything only depends on the thing before it. I find that this untangling of knowledge is just so intellectually interesting as a cognitive task. I love doing it personally, but I just have a fascination with trying to lay things out in a certain way. Maybe that helps me.

Dwarkesh Patel ***02:18:41***

It also makes the learning experience so much more motivated. Your tutorial on the transformer begins with bigrams, literally a lookup table from, "Here's the word right now, or here's the previous word, here's the next word." It's literally just a lookup table.

Andrej Karpathy ***02:18:58***

That's the essence of it, yeah.

Dwarkesh Patel ***02:18:59***

It's such a brilliant way, starting with a lookup table and then going to a transformer. Each piece is motivated. Why would you add that? Why would you add the next thing? You could memorize the attention formula, but having an understanding of why every single piece is relevant, what problem it solves.

Andrej Karpathy ***02:19:13***

You're presenting the pain before you present a solution, and how clever is that? You want to take the student through that progression. There are a lot of other small things that make it nice and engaging and interesting. Always prompting the student.

There's a lot of small things like that are important and a lot of good educators will do this. How would you solve this? I'm not going to present the solution before you guess. That would be wasteful. That's a little bit of a…I don't want to swear but it's a dick move towards you to present you with the solution before I give you a shot to try to come up with it yourself.

Dwarkesh Patel ***02:19:51***

Because if you try to come up with it yourself, you get a better understanding of what the action space is, what the objective is, and then why only this action fulfills that objective.

Andrej Karpathy ***02:20:03***

You have a chance to try it yourself, and you have an appreciation when I give you the solution. It maximizes the amount of knowledge per new fact added.

Dwarkesh Patel ***02:20:11***

Why do you think, by default, people who are genuine experts in their field are often bad at explaining it to somebody ramping up?

Andrej Karpathy ***02:20:24***

It's the curse of knowledge and expertise. This is a real phenomenon, and I suffered from it myself as much as I try not to. But you take certain things for granted, and you can't put yourself in the shoes of new people who are just starting out. This is pervasive and happens to me as well.

One thing that's extremely helpful. As an example, someone was trying to show me a paper in biology recently, and I just instantly had so many terrible questions. What I did was I used ChatGPT to ask the questions with the paper in the context window. It worked through some of the simple things. Then I shared the thread to the person who wrote that paper or worked on that work. I felt like if they could see the dumb questions I had, it might help them explain better in the future.

For my material, I would love it if people shared their dumb conversations with ChatGPT about the stuff that I've created because it really helps me put myself again in the shoes of someone who's starting out.

Dwarkesh Patel ***02:21:19***

Another trick that just works astoundingly well. If somebody writes a paper or a blog post or an announcement, it is in 100% of cases that just the narration or the transcription of how they would explain it to you over lunch is way more, not only understandable, but actually also more accurate and scientific, in the sense that people have a bias to explain things in the most abstract, jargon-filled way possible and to clear their throat for four paragraphs before they explain the central idea. But there's something about communicating one-on-one with a person which compels you to just say the thing.

Andrej Karpathy ***02:22:07***

Just say the thing. I saw that tweet, I thought it was really good. I shared it with a bunch of people. I noticed this many, many times.

The most prominent example is that I remember back in my PhD days doing research. You read someone's paper, and you work to understand what it's doing. Then you catch them, you're having beers at the conference later, and you ask them, "So this paper, what were you doing? What is the paper about?"

They will just tell you these three sentences that perfectly captured the essence of that paper and totally give you the idea. And you didn't have to read the paper. It's only when you're sitting at the table with a beer or something, and they're like, "Oh yeah, the paper is just, you take this idea, you take that idea and try this experiment and you try out this thing." They

have a way of just putting it conversationally just perfectly. Why isn't that the abstract?

Dwarkesh Patel ***02:22:51***

Exactly. This is coming from the perspective of how somebody who's trying to explain an idea should formulate it better. What is your advice as a student to other students, if you don't have a Karpathy who is doing the exposition of an idea? If you're reading a paper from somebody or reading a book, what strategies do you employ to learn material you're interested in in fields you're not an expert at?

Andrej Karpathy ***02:23:20***

I don't know that I have unique tips and tricks, to be honest. It's a painful process. One thing that has always helped me quite a bit is—I had a small tweet about this—learning things on demand is pretty nice. Learning depth-wise. I do feel you need a bit of alternation of learning depth-wise, on demand—you're trying to achieve a certain project that you're going to get a reward from—and learning breadth-wise, which is just, "Oh, let's do whatever 101, and here's all the things you might need." Which is a lot of school—does breadth-wise learning, like, "Oh, trust me, you'll need this later," that kind of stuff. Okay, I trust you. I'll learn it because I guess I need it. But I love the kind of learning where you'll get a reward out of doing something, and you're learning on demand.

The other thing that I've found extremely helpful. This is an aspect where education is a bit more selfless, but explaining things to people is a beautiful

way to learn something more deeply. This happens to me all the time. It probably happens to other people too because I realize if I don't really understand something, I can't explain it. I'm trying and I'm like, "Oh, I don't understand this." It's so annoying to come to terms with that. You can go back and make sure you understood it. It fills these gaps of your understanding. It forces you to come to terms with them and to reconcile them.

I love to re-explain things and people should be doing that more as well. That forces you to manipulate the knowledge and make sure that you know what you're talking about when you're explaining it.

Dwarkesh Patel ***02:24:48***

That's an excellent note to close on. Andrej, that was great.

Andrej Karpathy ***02:24:51***

Thank you.

—---