Season 2 Ep 22 Geoff Hinton on revolutionizing artificial intelligence again

Created by Podalize

November 16, 2022

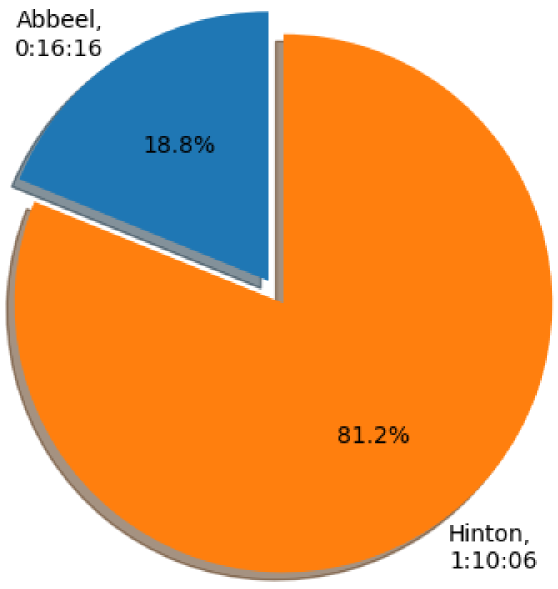

Figure 1: Percentage of spoken time per speaker

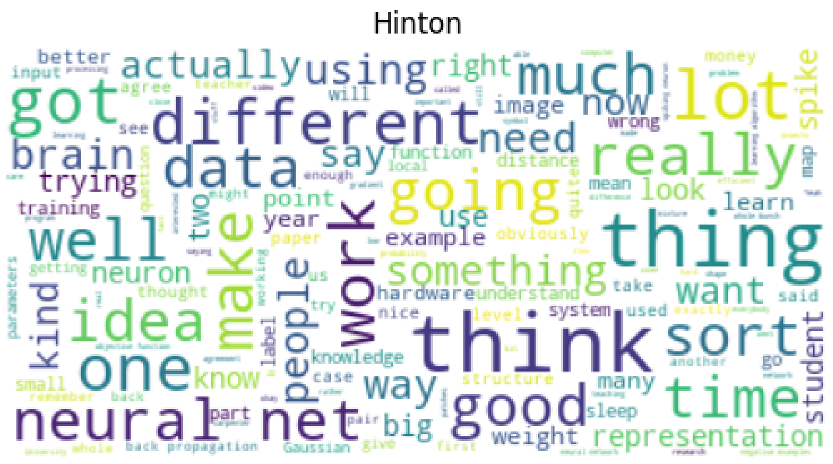

Figure 2: Word cloud per speaker

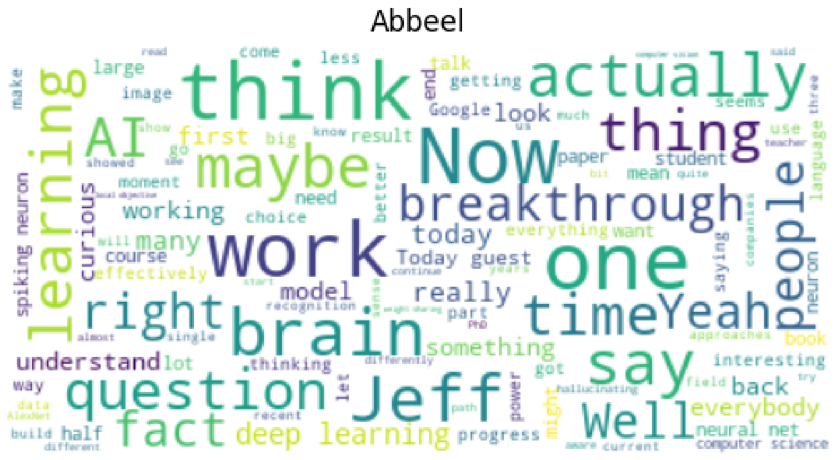

Figure 3: Word cloud per speaker

1 Transcript

Abbeel 00:00
Over the past 10 years, AI has experienced breakthrough after breakthrough after breakthrough in computer vision, in speech recognition, in machine translation, in robotics, in medicine, in computational biology, protein folding prediction, and the list goes on and on and on. And the breakthroughs aren't showing any signs of stopping. Not to mention, these AI breakthroughs are directly driving the business of trillion-dollar companies and many, many new startups. Underneath all of these breakthroughs is one single subfield of AI, deep learning. So, when and where did deep learning originate? And when did the AI become the most prominent AI approach? Today's guest has everything to do with this. Today's guest is arguably the single most important person in AI history and continues to lead the charge today. The award, the equivalent of the Nobel Prize for Computer Science. Today's guest has their work cited over half a million times. That means there is half a million and counting other research papers out there that build on top of his work. Today's guest has worked on deep learning for about half a century and most of the time in relative obscurity. But that all changed in 2012 when he showed deep learning is better at image recognition than any other approaches to computer vision and by a very large margin. That result, that moment known as the ImageNet moment changed the whole AI field. Pretty much everyone dropped what they had been doing and switched to deep learning. Former students of today's guest include Vladni, who put deep mind on the map with their first major result on learning to play Atari games and includes our season one finale guest, Elias Tsutskiver, founder and research director of OpenAI. In fact, every single guest in our podcast has built on top of the work done by today's guest. I am, of course, talking about no one less than Jeff Hinton. Jeff, welcome to the show. So happy to have you here.

Hinton 02:35
Well, thank you very much for inviting me.

Abbeel 02:37
Well, so glad to get to talk with you on the show here. And I'd say let's dive right in with maybe the, you know, the highest level question I can ask you. What are neural nets and why should we care?

Hinton 02:51
Okay, if you already know a lot about neural nets, please forgive the simplifications. Here's how your brain works. It has lots of little processing elements called neurons. And every so often a neuron goes ping. And what makes it go ping is that it's hearing pings from other neurons. And each time it hears a

ping from another neuron, it adds a little weight to some store of input that it's got. And when it gets it, when it's got enough input, it goes ping. And so if you want to know how the brain works, all you need to know is how the neurons decide to adjust those weights that they add when a ping arrives. That's all you need to know. There's got to be some procedure used for adjusting those weights. And if we could figure it out, we'd know how the brain works.

Abbeel 03:39
And that's been your quest for a long time now, figuring out how the brain might work. And what's the status? Do you, do we as a field understand how the brain works?

Hinton 03:50
Okay, I always think we're going to crack it in the next five years, since that's quite a productive thing to think. But I actually do. And I think we're going to crack it in the next five years. I think we're getting closer. I'm fairly confident now that it's not back propagation. So all of existing AI, I think, is built on something that's quite different from what the brain's doing. At a high level, it's got to be the same. That is, you have a lot of parameters, these weights between neurons, and you adjust those parameters on the basis of lots of training examples. And that causes wonderful things to happen if you have billions of parameters. The brain's like that. And deep learning is like that. The question is, how do you get the gradient for adjusting those parameters? So what you want is some, some measure of how well you're doing. And then you want to adjust the parameters so they improve that measure of how well you're doing. But my belief currently is that back propagation, which is the way deep learning works at present, is quite different from what the brain's doing. The brain's getting gradients in a different way.

Abbeel 04:59
Now, that's interesting. You're the one saying that, Jeff, because you actually, you wrote a paper on back-creation for training neural networks, and it's powering everything everybody's doing today. And now here you are saying, actually, it's probably time for us to figure out, oh, do you think we should change it close to what the brain is doing? Or do you think maybe back-creation could be better than what the brain is doing?

Hinton 05:23
Let me first correct you. Yes, we did write the most cited paper on back propagation, Rommelhart and Williams and me. Back propagation was already known to a number of different authors. What we really did was show that it could learn interesting representations. So it wasn't that we invented back

propagation. Rommelhart reinvented back propagation. We showed that it could learn interesting representations, like, for example, word embeddings. So I think back propagation is probably much more efficient than what we have in the brain at squeezing a lot of information into a few connections, whereby a few connections, I mean only a few billion. So the problem the brain has is that connections are very cheap. We've got hundreds of trillions of them. Experience is very expensive. And so we are willing to throw lots and lots of parameters at a small amount of experience, whereas the neural nets we're using are basically the other way around. They have lots and lots of experience, and they're trying to get the information about what relates the input to the output into the parameters. And I think back propagation is much more efficient than what the brain is using it doing that, but maybe not as good at, from not much data, abstracting a lot of structure.

Abbeel 06:52
And well, this begs the question, of course. Do you have any hypothesis on approaches that might get better performance in that regard?

Hinton 07:02
I have a sort of general view, which I've had for a long, long time, which is that we need unsupervised objective functions. So I'm talking mainly about perceptual learning, which I think is the sort of key. If you can learn a good model of the world by looking at it, then you can base your actions on that model rather than on the raw data. And that's going to make doing the right things much easier. I'm convinced that the brain is using lots of little local objective functions. So rather than being a kind of end-to-end system trained to optimize one objective function, I think it's using lots of little local ones. So as an example, the kind of thing I think will make a good objective function, though it's hard to make it work, is if you look at a small patch of an image and try and extract some representation of what you think is there, you can now compare the representation you got from that small patch of the image with a contextual bet that was got by taking the representations of other nearby patches and based on those predicting what that patch of the image should have in it. And obviously, once you're very familiar with the domain, those predictions from context and local extracted features will agree, generally agree, and you'll be very surprised when they don't. And you can learn an awful lot on one trial if they disagree radically. So that's an example of why I think the brain could learn a lot from the local disagreement. It's hard to get that to work, but I'm convinced something like that is going to be the objective function. And if you think of a big image and lots of little local patches in the image, that means you get lots and lots of feedback in terms of the agreement of what was extracted locally and what was predicted contextually all over the image and at many different levels of representation. And so we can get a much, much richer feedback from these agreements with contextual predictions, but making all that work is difficult.

But I think it's going to be along those lines.

Abbeel 09:18
Now what you're describing strikes me as part of what people are trying to do in self-supervised and unsupervised learning. And in fact, you wrote one of the breakthrough papers, the SimClear paper with a couple of collaborators, of course, in this space. What do you think about the SimClear work in contrastive learning more generally? And what do you think about the recent masked autoencoders? And how does that relate to what you just described?

Hinton 09:44
It relates quite closely to what I've... It's evidence that that kind of objective function is good. I didn't write the SimClear paper. Tim Cheung wrote the SimClear paper with help from the major co-authors. My name was on the paper for general inspiration, but I did write a paper a long time ago with Sue Becker on the idea of getting agreement between representations you got from two different patches of the image. So that was, I think of that as the origin of this idea of doing self-supervised learning by having agreement between representations from two patches of the same image. The method that Sue and I used didn't work very well because of a subtle thing that we didn't understand at the time, but I now do understand. And I could explain that if you like, but I'll lose most of the audience.

Abbeel 10:45
Well, I'm curious. I think it'd be great to hear it, but maybe we can zoom out for a moment before zooming back in. You talk about current methods use end-to-end learning backpropagation to power the end-to-end learning. And you're saying switch to learn from less data and extract more from less data is going to be key as a way to make progress to get closer to how the brain learns.

Hinton 11:09
Yes. So you get much bigger bandwidth for learning by having many, many little local objective functions.

Abbeel 11:17
And when we look at these local objective functions like filling in a blanked out part of an image or maybe filling back in a word, if we look at today's technologies, in fact, this is the current frontier and you've contributed, a lot of people are working exactly on that problem of learning from unlabeled data effectively because it requires a lot less human labor, but they still use backpropagation. The same mechanism.

Hinton 11:45
So what I don't like about the master autoencoder is you have your input patches and then you go through many layers of representation. And at the output of the net, you try to reconstruct the missing input patches. I think the brain, you have these levels of representation, but at each level, you're trying to reconstruct what's at the level below. So it's not like you go through this many, many layers and then come back out again. It's that you have all these levels, each of which is trying to reconstruct what's at the level below. So I think that's much more brain like. And the question is, can you do that without using backpropagation? Obviously, if you go through many, many levels and then reconstruct the missing patches at the output, you need to get information back through all those levels. And since we have backpropagation, it's built into all the simulators, you might as well do it that way. But I don't think that's how the brain is doing it.

Abbeel 12:45
Now imagine the brain is doing it with all these local objectives. Do you think for our engineered systems, will it matter whether... In some sense, there are three choices to make, it seems. One choice is, what are the objectives? What are those local objectives that we want to optimize? A second choice is, what's the algorithm to use to optimize it? And then a third choice is, what's the architecture of how do we wire the neurons together that are doing this learning? And among those three, it seems like all three could be the missing piece that we're not getting right. Or what do you think?

Hinton 13:26
Well, if you're interested in perceptual learning, I think it's fairly clear you want retinotopic maps, a hierarchy of retinotopic maps. So the architecture is local connectivity. And the point about that is, you can solve a lot of the credit assignment problem by just assuming that something in one locality in a retinotopic map is going to be determined by the corresponding locality in the retinotopic map that feeds into it. So you're not trying to low down in the system, figure out how pixels determine what's going on a long distance away in the image. You're going to just use local interactions. And that gives you a lot of locality. And you'd be crazy not to use that. One thing neural nets do at present is they assume they're going to be using the same functions at every locality. So convolutional nets do that. And transformers do that too. I don't think the brain can do that because that would involve weight sharing. And it would involve doing exactly the same computation at each locality so you can use the same weights. I think it's most unlikely the brain does that. But actually, there's a way to achieve what weight sharing does, what convolutional nets do in the brain in a much more plausible way than I think people

have suggested before, which is if you do have contextual predictions trying to agree with locally extracted things, then imagine a whole bunch of columns that are making local predictions and looking at nearby columns to get their contextual prediction. You can think of the context as a teacher for the local thing, but also vice versa. But think of the context as a teacher for what you're extracting locally. So you can think of the information that's in the context as being distilled into the local extractor. But that's true for all the local extractors. So what you've got is mutual distillation, where they're all providing teaching signals for each other. And what that means is knowledge about what you should extract in one location is getting transferred into other locations if they're trying to agree, if you're trying to get different locations to agree on something. If, for example, you find a nose and you find a mouth, then you want them both to agree that they're part of the same face. So they should both give rise to the same representation. Then the fact that you're trying to get the same representation at different locations allows knowledge to be distilled from one location to another. And there's a big advantage of that over actual weight sharing. Obviously, biologically, one advantage is that the detailed architecture in these different locations doesn't need to be identical. But the other advantage is the front end processing doesn't need to be the same. So if you take your retina, different parts of the retina have different size receptive fields. And convolutional nets try to ignore that. They sometimes have multiple different resolutions and do convolution at each resolution. But they just can't deal with different front end processing. Whereas if you're distilling knowledge from one location to another, what you're trying to do is get the same function from the optic array to the representation in these different locations. And it's fine if you pre-process the optic array differently in the two different locations. You can still distill the knowledge across the function from the optic array to the representation, even though the front end processing is different. And so, although distillation is less efficient than actually sharing the weights, it's much more flexible, and it's much more neural implausible. So for me, that was a kind of big insight I had about a year ago that we have to have something like weight sharing to be efficient. But local distillation will work if you're trying to get neighboring things to agree on a representation. But that idea of trying to get them to agree gives you the signal you need for knowledge in one location to supervise knowledge in another location.

Abbeel 17:38
And Jeff, do you think... So what you're describing, one way to think of it is to say, hey, weight sharing is clever because it's something the brain kind of does too. It just does it differently. So we should continue to do weight sharing. Another way to think of it is that actually we shouldn't continue to do weight sharing because the brain does it somewhat differently and there might be a reason to do it differently. What's your thinking?

Hinton 18:01
I think the brain doesn't do weight sharing because it's hard for it to shift synapse strengths about the place. It's very easy if they're all sitting in RAM. So I think we should continue to do convolutional things in convlets and in transformers. We should share weights. We should share knowledge by sharing weights. But just bear in mind that the brain is going to share knowledge not by sharing weights, but by sharing the function from input to output and using distillation to transfer knowledge.

Abbeel 18:30
Now there's the other topic that is talked about quite a bit where the brain is drastically different from our current neural nets and it's the fact that neurons work with spiking signals and that's very different from our artificial neurons in our GPUs. And so I'm very curious on your thinking on that. Is that just an engineering difference or do you think there could be more to it that we need to understand better and benefits to

Hinton 18:59
I think it's not just an engineering difference. I think once we understand why that hardware is so good, why you can do so much in such an energy efficient way with that kind of hardware, we'll see that it's sensible for the brain to use spiking neurons. The retina, for example, doesn't use spiking neurons. The retina does lots of processing with non-spiking neurons. So once we understand why Cortex is using those, we'll see that it was the right thing for biology to do. And I think that's going to hinge on what the learning algorithm is, how you get gradients for networks of spiking neurons. And at present, nobody really knows. At present, what people do is say, you see, the problem with the spiking neuron is there's two quite different kinds of decision. One is exactly when does it spike and the other is does it or doesn't it spike. So there's this discrete decision, should the neurons spike or not, and then this continuous variable of exactly when it should spike. People trying to optimize this system like that have come up with various kind of surrogate functions which sort of smooth things a bit so you can get continuous functions. They didn't seem quite right. It'd be really nice to have a learning algorithm. And in fact, in NIPS in about 2000, Andy Brown and I had a paper on trying to learn spiking Boltzmann machines. But it'd be really nice to get a learning algorithm that's good for spiking neurons. And I think that's the main thing that's holding up spiking neuron hardware. So people like Steve Ferber in Manchester have realized that, and many other people, have realized that you can make more energy efficient hardware this way. And they've built great big systems. What they don't have is a good learning algorithm for it. And I think until we've got a good learning algorithm for it, we won't really be able to exploit what we can do with spiking neurons. And there's one obvious thing you can do with them that isn't easy in conventional neural nets, and that's agreement. So if you take a standard

artificial neuron and you simply ask the question, can it tell if its two inputs have the same value? Well, it can't. It's not an easy thing for a standard neuron to do, a standard artificial neuron. If you use spiking neurons, it's very easy to build a system where if the two spikes arrive at the same time, they'll make the neuron fire. And if they arrive at different times, they won't. So using the time of a spike seems like a very good way of measuring agreement. We know the biological system does that. So you can see the direction of sound is coming from, or rather here the direction of sound is coming from, by the time delay in the signals reaching the two ears. And if you take a foot, that's about a nanosecond for light, and it's about a millisecond for sound. And the point is, if I move something sideways in front of you by a few inches, the difference in the time delay to the two ears, the length of the path to the two ears, is only a small fraction of an inch. And so it's only a small fraction of a millisecond difference in the time the signal gets to the two ears. And we can deal with that, and OWLS can deal with it even better. And so we're measuring, we're sensitive to times of like 30 microseconds in order to get stereo from sound. I can't remember what OWLS are sensitive to, but I think it's a lot better than 30 microseconds. And we do that by having two axons with spikes traveling in different directions, one from one ear and one from the other ear. And then you have cells that fire as the spikes get there at the same time. That's the simplification, but roughly that. So we know that spike timing can be used for exquisitely sensitive things like that. And it would sort of be very surprising if the precise times of spike wasn't being used, but we really don't know how. And for a long time, I thought it'd be really nice if you could use spike times to detect agreement for things like self-supervised learning, or for things like if I've extracted your mouth and I've extracted your nose or other representations of them. And from your mouth, I can now predict something about your whole face. And from your nose, I can predict something about your whole face. And if your mouth and nose are in the right relationship to make a face, those predictions will agree. And it'd be really nice to use spike timing to see that those predictions agree. But it's hard to make that work. And one of the reasons it's hard to make that work is because we don't know, we don't have a good algorithm for training networks of spike in URLs. So that's one of the things I'm focused on now. How can we get a good training algorithm for networks of spike in URLs? And I think that'll have a big impact on hardware.

Abbeel 23:56
That's a real interesting question you're putting forward there because I doubt too many people are working on that compared to, let's say, the number of people working on large language models or other problems that are much more, I guess, visible in terms of progress recently. I think...

Hinton 24:14
Yeah, it's always a good idea to figure out what huge numbers of very smart

people are working on and to work on something else.

Abbeel 24:21
Yeah. I think the challenge, of course, for most people, I'd say including myself, but I definitely hear the question from many students too, is that it's easy to work on something else than everybody else, but it's hard to make sure that something else is actually relevant because there's many other things out there that are not very relevant you could possibly spend time on.

Hinton 24:42
Yeah, that involves having good intuitions.

Abbeel 24:44
Yeah. Listening to you, for example, could help. So I have actually a follow-up question, something you just said, Jeff, which is that the retina doesn't use all spiking neurons. Are you saying that the brain has two types of neurons, some that are more like our artificial neurons and some that are spiking neurons?

Hinton 25:08
I'm not sure the retina is more like our artificial neurons, but certainly the cortex has... The neocortex has spiking neurons. And its primary mode of communication is by sending spikes from one pyramidal cell to another pyramidal cell. And I don't think we're going to understand the brain until we understand why it chooses to send spikes. For a while, I thought I had a good argument that didn't involve the precise time to spikes. And the argument went like this. The brain's in the regime where it's got lots and lots of parameters and not much data relative to the typical neural nets we use. And there's a potential of overfitting in that regime unless you use very strong regularization. And a good regularization technique is dropout, where each time you use a neural net, you ignore a whole bunch of the units. And so maybe the fact that the neurons are sending spikes, what they're really communicating is the underlying Poisson rate. So let's assume it's Poisson, which is close enough for this argument. There's a Poisson process which sends spikes stochastically, but the rate of that process varies, and that's determined by the input to the neuron. And you might think you'd like to send the real valued rate from one neuron to another. But if you want to do lots and lots of regularization, you could send the real valued rate with some noise at it. And one way to add noise is to just use spikes. That'll add lots of noise. And so this was the motivation for dropout, that most of the times, most of the neurons aren't involved in things, if you look at any fine time window. And you can think of spikes as a representation of underlying Poisson rate. It's just a very, very noisy representation, which sounds like a very, very bad idea because it's very, very noisy. But actually, once you under-

stand about regularization, where you have too many parameters, it's a very, very good idea. So I still have a lingering fondness for the idea that actually we're not using spike timing at all. It's just about using very noisy representations of Poisson rates to be a good regularizer. And I sort of flip between, I think it's very important when you do science not to totally commit to one idea and ignore all the evidence for other ideas. But if you do that, you end up flipping between ideas every few years. So some years, I think neural nets are deterministic. I mean, we should have deterministic neural nets, and that's what backpots easy. And other years, I think it's about a five-year cycle. I think, no, no, it's very important that they're stochastic. And that changes the play for everything. So Boltzmann machines were intrinsically stochastic, and that was very important to them. But the main thing is not to fully commit to either of those, but to be open to both.

Abbeel 28:19
Now one thing, if I think more about what you just said, the importance of spiking neurons and figuring out how to train a spiking neuron network effectively, what if we, for now, just say, let's not worry about the training part, given that seemingly it's far more power efficient. Wouldn't people want to distribute pure inference chips that you pre-train effectively separately, and then you compile it onto a spiking neuron chip to have very low power inference capabilities? What about that?

Hinton 28:53
Yes. Lots of people have thought of that, and it's a very sensible idea. And it's probably on the evolutionary path to getting to use spiking neural nets, because once you're using them for inference, and it works, and people are already doing that, and it's already working, being shown to be more power efficient. And various companies have produced these big spiking systems. Once you're doing them for inference anyway, you'll get more and more interested in how you could learn in a way that makes more use of the available power in these spiked times. So you can imagine a system where you learn using back prop, but not on analog hardware, for example, not on this low energy hardware. And then you transfer it to the low energy hardware, and that's fine. But we'd really like to learn directly in the hardware.

Abbeel 29:52
Now one thing that really strikes me, Jeff, is when I think about your talks back around 2005, 6, 7, 8, when I was a PhD student, essentially pre-AlexNet talks, those talks, I think, topically have a lot of resemblance to what you're excited about now. And it almost feels like AlexNet is an outlier in your path. How did you go from thinking so closely about how the brain might work to deciding that maybe you can first explain what was AlexNet, but also how did it come about,

and what was that path to go from working on restricted Boltzmann machines, trying to see how the brain works to, I would say, the more traditional approach to neural nets that you all of a sudden showed can actually work?

Hinton 30:41
Well, if you're an academic, you have to raise grant money. And it's convenient to have things that actually work, even if they don't work the way you're interested in. So part of it's that, just go with the flow, and if you can make back prop work well. And back then, in about 2006, 2005, I got fascinated by the idea you could use stacks of restricted Boltzmann machines to pre-train feature detectors, and then it would be much easier to get back to It turned out with enough data, which is what you had in speech recognition, and later on because of Fei-Fei Li and her team in image recognition, with enough data, you don't need the pre-training. Although pre-training is coming back. I mean, GPT-3 has pre-training, and pre-training is a thoroughly good idea. But once we discovered that you could pre-train and that will make back prop work better, and that did great things for speech, which George Dahl and Nader Ramo-Mohammed did in 2009, then Alex, who was a graduate student in my group then, started applying the same ideas to vision. And pretty soon we discovered that you didn't actually need this pre-training, especially if you have the ImageNet data. And in fact, that project was partly due to Ilya's persistence. So I remember Ilya coming into the lab one day and saying, look, now that we've got speech recognition working, this stuff really works. We've got to do ImageNet before anybody else does. And retrospectively, I learned that Yan-La Koum was going into the lab and saying, look, we've got to do ImageNet with Combinets before anybody else does. And Yan's students and postdocs said, oh, but I'm busy doing something else. He couldn't actually get someone to commit to it. And Ilya initially couldn't get people to commit to it. And so Ilya persuaded Alex to commit to it by pre-processing the data for him. So he didn't have to pre-process the data. The data was all pre-processed to be just what he needed. And then Alex really went to town. And Alex is just a superb programmer. And Alex was able to make a couple of GPUs really sing. He made them work together in his bedroom at home. I don't think his parents realized that they were paying most of the cost because that was electricity. But he did a superb job of programming convolutional nets on them. So Ilya said, we've got to do this, and helped Alex with the design and so on. Alex did the really intricate programming. And I provided support and a few ideas, like using Dropout. I also did some good management. I'm not often very good at management, but I'm very proud of the management I did, which is Alex Kruszewski had to write a depth novel to show that he was capable of understanding research literature, which is what you have to do for a couple of years to stay in the program. And he doesn't really like writing. And he didn't really want to do the depth novel, but it was way past the deadline and the department was hustling us. So I said to him, each time you can improve the performance by 1% on ImageNet, you can delay your depth novel by another week. And Alex delayed his depth novel

by a whole lot of weeks.

Abbeel 34:24
Yeah. And just for context, I mean, a lot of researchers know this, of course, but maybe not everybody. Alex's result with you and Ilya cut the error rate in half compared to prior work on the ImageNet image recognition competition, which was just...

Hinton 34:41
More or less. I used to be a professor, so it wasn't quite in half. Close. It cut it from about 26% to about 16 or 15%, depending on how you count. It didn't cut it in half, but it cut it almost in half.

Abbeel 34:54
Almost in half. Whereas in previous years, the progress was by 1% or 2%. Here was a very, very... A whole different... Well, that's why everybody switched from what they were doing, which was hand-engineered approaches to computer vision, try to program directly. How can a computer understand what's in Image to deep learning?

Hinton 35:14
I should say one thing that's important to say here. So, Yan Luker spent many years developing convolutional neural nets. And it really should have been him, his lab that developed that system. We had a few little extra tricks, but they weren't the important thing. The important thing was to apply convolutional nets using GPUs to a big data set. So, Yan was kind of unlucky in that he didn't get the win on that, but it was using many of the techniques that he developed.

Abbeel 35:45
It didn't have the Russian immigrants that Toronto and you had been able to attract to make it happen.

Hinton 35:51
Well, one's Russian and one's Ukrainian, and it's important not to confuse those. Even though the Ukrainians are Russian speaking Ukrainian, don't confuse Russian with Ukrainian. Absolutely. It's a different country.

Abbeel 36:03

So, now, Jeff, that moment actually also marked a big change in your career, because as far as I understand, you'd never been involved in corporate work, but it marked a transition for you soon thereafter from being a pure academic to being... Ending up at Google, actually. Can you say a bit about that? How was that for you? Did you have any internal resistance?

Hinton 36:30
I can say why that transition happened. What triggered it? Yeah, I'm curious. So, I have a lonely disabled son who needs future provisions, so I needed to get a lump of money. And I thought one way I might get a lump of money was by teaching a Coursera course. And so, I did a Coursera course on your own networks in 2012, and it was one of the early Coursera courses, so their software wasn't very good. So, it was extremely irritating to do. It really was very irritating then. I'm not very good on software, so I didn't like that. And from my point of view, it amounted to you agree to supply a chapter of a text-book, one chapter every week. So, you had to give them these videos, and then a whole bunch of people were going to watch the videos. Like sometimes the next day, Yoshua Benjo would say, why did you say that? So, you know that it's going to be people who know very little, but also people who know a whole lot. And so, it's stressful. You know that if you make mistakes, they're going to be caught. Not like a normal lecture where you can just sort of press on the sustaining pedal and sort of blow your way through it if you get some certain confused about something. Here, you have to get it straight. And the deal with the University of Toronto originally was that the idea of a chapter of a textbook was that if any money was made from these courses, which I was hoping there would be, the money that came to the University would be split with the professor. They didn't specify exactly what the split would be, but one assumed it would be like 50-50 or something like that. And I was okay with that. The University didn't provide any support in preparing the videos. And then after I started the course and when I could no longer back out of it, the Provost made a unilateral decision without consulting me or anybody else that actually if money came from Coursera, the University would take all the money and the professor would get zero, which is exactly the opposite of what happens with textbooks. And the process was very like preparing a textbook. I actually asked the University to help me prepare the videos and the AV people came back to me and said, do you have any idea how expensive it is to make video? And I actually did have an idea because I've been doing it. So I got really pissed off with my university because they unilaterally sort of cancelled the idea I get any remuneration for this. They said it was part of my teaching. Well, actually it wasn't part of my teaching. It was clearly based on lectures I'd given this part of my teaching, but I was doing my teaching as well as that. And I wasn't using that course for my teaching. And that got me pissed off enough that I was willing to consider alternatives to being a professor. And at that time, we then suddenly got interest from all sorts of companies in recruiting us, either in funding, giving big grants or in funding a startup. It was clear that a number

of big companies were just very interested in getting in on the act. And so normally I would have just said, no, I get paid by the state for doing research. I don't want to try and make extra money from my research. I'd rather get on with the research. But because that particular experience with the University cheating me out of the money. Now, it turned out they didn't cheat me out of anything because no money was being spent on it. No money came from the course anyway. But that pushed me over the edge into thinking, well, okay, I'm going to find some other way to make some money. That was the end of my

Abbeel 40:14
friendship. Oh no. Well, but the result is that these companies are, and in fact, if you read the Genius Makers book by Cade Metz, which I reread last week in preparation for this conversation, if you read the book, it starts off with actually you running an auction for these companies to try to acquire your company, which is quite the start of a book. Very intriguing. But how was it

Hinton 40:44
for you? Oh, when it was happening, it was at NIPS and Terry had organized NIPS in a casino in Lake Tahoe. And so in the basement of the hotel, there were these smoke filled rooms full of people pulling one-armed bandits and big lights flashing saying you won $25,000 and all that stuff and people gambling in other ways. And upstairs, we were running this auction. And we felt like we were in a movie. We felt like this was like being in that movie, The Social Network. It sort of felt like that. It was great. The reason we did it was we had absolutely no idea how much we were worth. And I consulted a lawyer, an IP lawyer, and said there's two ways to go about this. You could hire a professional negotiator. In that case, you'll end up working for a company, but they'll be pissed off with you. Or you could just run an auction. As far as I know, this was the first time a small group like that just ran an auction. We ran it on Gmail. I'd worked at Google over the summer, so I knew enough about Google to know that they wouldn't read our Gmail. And I'm still pretty confident they didn't read our Gmail. Microsoft wasn't so confident. And we just ran this auction where people had to Gmail me their bids. And we then immediately mailed them out to everybody else with the timestamp of the Gmail. And we just kept going up by half a million dollars. I think it was half a million dollars to begin with and then a million dollars after that. And yeah, it was pretty exciting. And we discovered we were worth a lot more than we thought. Retrospectively, we could probably have got more, but we got to an amount that we thought was astronomical. And then basically, we wanted to work for Google, so we stopped the auction, so we could be sure of working for Google. And as I understand it, you're still at

Abbeel 43:01

Google today. I'm still at Google today. Nine years later, I'm in the same company as you.

Hinton 43:08
Nine years later, I'm in my 10th year there. I think I'll get some kind of award when I'm be there for 10 years because it's so rare. Although people tend to stay at Google longer than other companies. Yeah, I like it there. The main reason I like it is because the Brain team is a very nice team. And I get along very well with Jeff Dean. He's kind of very smart, but very straightforward to deal with. And what he wants me to do is do what I want to do, which is basic research. He thinks what I should be doing is trying to come up with radically new algorithms. And that's what I want to do anyway. So it's just a very nice fit. I'm no good at managing a big team to improve speech recognition by 1%. I'd be happy to say that. Well, it's better

Abbeel 43:53
to just revolutionize the field again, right? Yeah. I would like to do it one more time. But I'm looking forward to it. I wouldn't be surprised at all. Now, when I look at your career, Jeff, and some of this information actually comes from the book, as I didn't notice before, I had read the book the first time. I mean, you were a computer science professor at the University of Toronto, Emeritus now, I believe, but computer science, but you never got a computer science degree. You got a psychology degree. And you actually, at some point, were a carpenter. How does it come about? How do you go from studying psychology to becoming a carpenter, to getting into AI? What's the path for you there? How do you look at that?

Hinton 44:45
In my last year at Cambridge, I had a very difficult time and got very unhappy. And I dropped out just after the exams I dropped out and became a carpenter. And I'd always enjoyed carpentry more than anything else. So at high school, there'd be sort of all the classes, and then you could stay in the evenings and do carpentry, and that's what I really looked forward to. And so I became a carpenter. And then after I'd been a carpenter for about six months, you couldn't actually make a living as a carpenter. So I was a carpenter and decorator. I made the money doing decorating, but I had the fun doing carpentry. And the point is, carpentry is more work than it looks, and decorating is less work than it looks. So you can charge more per hour for decorating, unless you're a very good carpenter. And then I met a real carpenter, and I realized I was completely hopeless at carpentry. And so he's making a door for a basement, for a coal cellar under the sidewalk that was very damp. And he was taking pieces of wood and arranging them so that they will warp in opposite directions so that it will cancel out. And that was kind of a level of kind of understanding and

thought about the process. That never occurred to me. He could also take a piece of wood and just cut it exactly square with a handsaw. And he explained something useful to me. He said, if you want to cut a piece of wood square, you have to line the saw bench up with the room, and you have to line the piece of wood up with the room. You can't cut it square if it's not aligned with the room, which is very interesting in terms of coordinate frames. So anyway, because I was so hopeless compared with him, I decided that I might as well go back into AI.

Abbeel 46:46
Now, when you say get back into AI, as I understand it, this was at the University of Edinburgh,

Hinton 46:51
where you went for your PhD? Yeah, I went to do a PhD there. And I went to do a PhD in neural networks with an eminent professor called Christopher Lunger Higgins. He was really very brilliant. He almost got a Nobel Prize when he was in his 30s for figuring out something about the structure of boron hydride. And I still don't understand what it is because it's all to do with quantum mechanics. But it hinged on the fact that 360 degree rotation is not the identity operator. It's 720 degrees. There's a thing you want to find in those books about it. Anyway, he was interested in neural nets and the relation to holograms. And about the day I arrived in Edinburgh, he lost interest in neural nets because he read Winograd's thesis, and he became completely converted. He thought neural nets was the wrong way to think about it. We should do symbolic AI. He was very impressed by Winograd's thesis. And so he had a lot of integrity. So even though he completely disagreed with what I was doing, he didn't stop me doing it. He kept trying to get me to do stuff more like Winograd's thesis, but he let me carry on doing what I was doing. And yeah, I was a bit of a loner. Everybody else back then in the early 70s was saying, Minsky and Papert have shown that neural nets are nonsense. Why are you doing this stuff? It's crazy. And in fact, the first talk I ever gave to that group was about how to do true recursion with neural networks. So this was talking 1973, so 49 years ago. And so one of my first projects, I discovered a write-up of it recently, was you want a neural network that will be able to draw a shape, and you want it to parse the shape into parts. And you want it to be possible for a part of the shape to be drawn by the same neural hardware as the whole shape is being drawn by. So the neural hardware that's drawing the whole shape has to remember where it's got to in the whole shape and what the orientation and position size is for the whole shape. But now it has to go off, and you want to use the very same neurons for drawing a part of the shape. So you need somewhere to remember what the whole shape was and how far you got in it, so that you can pop back to that once you finish doing this subroutine, this part of the shape. And the question is, how is a neural network going to remember that? Because obviously you

can't just copy the neurons. And so I managed to get a system working where the neural network remembered it by having fast Hebbian weights that were just adapting all the time, and were adapting so that any state that had been in recently could be retrieved by giving it part of that state and then say, filling the rest. And so I had a neural net that was doing true recursion, reusing the same neurons and the same weights to do the recursive call as it used for a high level call. And that was in 1973. And the, I think people didn't understand the talk because I wasn't very good at giving talks, but they also said, why would you want to do recursion with a neural net? You can do recursion with Lisp. They didn't understand the point, which is that, you know, the neural network is not going to be able to do it. Unless we get neural nets to do something like recursion, we're never going to be able to explain a whole bunch of things. And now that's become sort of an interesting question again. So I'm going to wait one more year until that idea is an antique, a genuine antique, if it'll be 50 years old, and then I'm going to sort of write up the research I did then. And it was all about fast weights as a member.

Abbeel 50:50
I have many questions here, Jeff. The first one is you're standing in this room, or everybody's, you're a PhD student or maybe fresh out of PhD. You're standing in a room with essentially everybody telling you what you're working on is a waste of time. And you were convinced somehow it

Hinton 51:13
was not. Where do you get that conviction from? I think a large part of it was my school. So my father was a communist, but he sent me to an expensive private school because they had good science education. And I was there from the age of seven. I had a preschool, and it was a Christian school. And all the other kids believed in God. And it was just at home. I was taught that that was nonsense. And it did seem to me that it was nonsense. And so I was used to just having everybody else being wrong and obviously wrong. And I think that's important. I think you need, you need, I was about to say you need the faith, which is funny in this situation. You need the faith in science to be willing to work on stuff just because it's obviously right, even though everybody else says it's nonsense. And in fact, it wasn't everybody else. It was everybody else in the early 70s doing AI said it was nonsense, or nearly everybody else. But if you look a bit earlier, if you look in the 50s, both von Neumann and Turing believed in neural nets. Turing in particular believed in neural nets training with reinforcement. So if I still believe if they hadn't both died early, the whole history of AI might have been very different because they were sort of powerful enough intellects to assuage a field. And they were very interested in how does the brain work. So I think it's just bad luck that they've died early. Well, British intelligence might have come into it.

Abbeel 53:11
Now, you go from believing in this. Well, at the time, many people didn't. Getting the big breakthroughs helped me that power almost everything that's being done today. And now there is this in some sense, the next question, right, is it's not just that deep learning works and works great, the question becomes, is it all we need? Or will we need other things? And you've said things, maybe I'm not literally quoting you, but to the extent of deep learning will do everything. What I really meant by that, I sometimes say things without thinking without

Hinton 53:50
being accurate enough. And then people call me on it, like saying we won't need radiologists. So what I really meant was using stochastic gradient descent to adjust a whole bunch of parameters. That's what I sort of had in mind when I said deep learning. The way you get the gradient might not be back propagation. And the thing you get the gradient of might not be some final performance measure, but rather these lots of local objective emissions. But I think that's how the brain works. And I think that's going to explain everything. Yes. Well, nice to see it confirmed. So one other thing I want to say is the kind of computers we have now are very good for doing banking, because they can remember how much you have in your account. It wouldn't be so good if you went in and they said, well, you got roughly this much. And we're not really sure, because we don't do it to that precision, but roughly this much. We don't want that in a computer doing banking, or in a computer guiding the space shuttle or something. We really rather got the answer exactly right. And they're very different from us. And I think people aren't sufficiently aware that we made a decision about how computing would be, which is that our knowledge would be immortal. So if you look at existing computers, you have a computer program, or maybe you just have a lot of weights for a neural net. That's a different kind of program. But if your hardware dies, you can run the same program on another piece of hardware. And so that makes the knowledge immortal. It doesn't hinge on that particular piece of hardware surviving. Now, the cost of the immortality is huge, because it means the two different bits of hardware have to do exactly the same thing. Obviously, there's error correction and all that, but after you've done all the error correction, they have to do exactly the same thing, which means they better be digital, or mostly digital. And they're probably going to do things like multiplying numbers together, which involves using lots and lots of energy to make things very discrete, which is not what hardware really wants to be. And so as soon as you commit yourself to the immortality of your program or your neural net, you're committed to very expensive computations, and also to very expensive manufacturing processes. You need to manufacture these things accurately, and probably in 2D and then put lots of 2D things together. If you're just willing to give up on immortality, in fiction, normally what you get in return is luck. But in reality, in fiction, normally what you get in return is love. But if we're willing to give up immor-

tality, what we'll get in return is very low energy computation and very cheap manufacturing. So instead of manufacturing computers, what we should do is grow them. We should use nanotechnology to just grow the things in 3D. And each one will be slightly different. So the image I have is if you take a pot plant and you sort of pull it out of its pot, there's a root ball, and it's the shape of the pot, right? And so all the different pot plants have the same shape root ball, but the details of the roots are all different. But they're all doing the same thing. They're extracting nutrients from the soil. And they've got the same function, and they're pretty much the same, but the details are all very different. So that's what real brains are like. And I think that's what what I call mortal computers will be like. So these are computers that are grown rather than manufactured. You can't program them. They just learn. They obviously have to have a learning algorithm sort of built into them. They learn. They can do most of their computation in analog, because analog is very good for doing things like taking a voltage times a resistance and turning it into a charge and then adding up the charge. And they're already chips that do things like that. The problem is what do you do next? And how do you learn in those chips? And at present, people have suggested back propagation or various versions of Boltzmann machines. I think we're going to need something else. But I think sometime in the not too distant future, we're going to see mortal computers, which are very cheap to create, have to get all their knowledge in there by learning and a very low energy. And these mortal computers, when they die, they die and their knowledge dies with them. And so it's no use looking at the weights, because those weights only work for that hardware. So what you're going to have to do is distill the knowledge into other computers. So when these mortal computers get old, they're going to have to do lots of podcasts to try and get knowledge into younger mortal computers.

Abbeel 59:07
The first one you build, I'll happily have that one on. Let me know. So Jeff, this reminds me of another question that's been on my mind for you, which is when you think about today's neural nets, the ones that grab the headlines are very, very large. I mean, not as large as the brain maybe, but in some sense, starting to get to that size, right? The large language models. And the results look very, very impressive. So one, I'm curious about your take on those kinds of models and what you see in them and what you see as limitations. But two, I'm also curious about what do you think about working on the other end of the spectrum? For example, ants have much smaller brains, obviously, than humans. Yet it's fair to say that our visual motor systems that we have developed artificially are not yet at the level of what ants can pull off or bees and so forth. And so I'm curious about that spectrum as well as the recent big advances in language models,

Hinton 1:00:21

what you think about those. So bees, they may look small to you, but I think a bee has about a million neurons. So I think a bee is closer to GPT-3, certainly closer than an ant is, but a bee is actually quite a big neural net. My belief is that if you take a system with lots of parameters and they're tuned sensibly using some kind of gradient descent in some kind of sensible objective function, then you'll get wonderful properties out of it. You'll get all these emergent properties like you do with GPT-3 and also the Google equivalents that aren't talked about so much. That doesn't sort of settle the issue of whether they're doing it the same way as us. And I think we're doing a lot more things like recursion, which I think we do in neural nets. And I try to address some of these issues in a paper I put on the web last year called GLOM. Well, I call it GLOM. It's how you do part-hole hierarchies in neural nets. So you definitely have to have structure. And if what you mean by symbolic computation is just that you have part-hole structure, then we do symbolic computation. That's not necessarily the case. If we have part-hole structure, then we do symbolic computation. That's not normally what people meant by symbolic computation. The sort of hardline symbolic computation means you're using symbols and you're operating on symbols using rules that just depend on the form of the symbol string you're processing. And that a symbol, the only property a symbol has is that it's either identical or not identical to some other symbol. And perhaps that it points to something. It can be epsilon. The neural nets are very different from that. So the sort of hardline symbol processing, I don't think we do that. But we certainly deal with part-hole hierarchies. But I think we do it in great big neural nets. And I'm sort of up in the air at present as to what extent does GPT-3 really understand what it's saying. I think it's fairly clear it's not just like the old ELISA program, which just rearranges strings of symbols and had no clue what it was talking about. And the reason for believing that is you say in English, show me a picture of a hamster wearing a red hat. And it draws a picture of a hamster wearing a red hat. And you're fairly sure it never got that pair before. So it has to understand the relationship between the English string and the picture. And before it had done that, if you'd asked any of these doubters, these neural net skeptics, neural net deniers, let's call them neural net deniers, if you'd asked them, well, how would you show that it understands? I think they'd have accepted that, well, if you asked a draw a picture of something, it draws a picture of that thing, then it understood. Just as with Winograd's thesis, you ask it to put the blue block in the green box, and it puts the blue block in the green box. And so that's pretty good evidence it understood what you said. But now that it does it, of course, the skeptics then say, well, you know, that doesn't really count. There's nothing that will satisfy them, basically.

Abbeel 1:03:50
Yeah, the goal line is always moving for true skeptics. Now, there's the recent one, the Google one, the Palm model that in the paper showed how it was explaining, effectively, how jokes work. That was extraordinary. That just seemed

a very deep understanding of language.

Hinton 1:04:12
No, it was just rearranging the words it had in its training set. You think so? No, no, I don't see how it could generate those explanations without sort of understanding what's going on. Now, I'm still open to the idea that because it was trained with back propagation, it's going to end up with a very different sort of understanding from us. And obviously, adversarial images tell you a lot, but you can recognize objects by using their textures. And you can be correct about it in the sense that it was generalized to other instances of those objects. But it's a completely different way of doing it from what we do. And I like to think of the example of insects and flowers. So insects see in the ultraviolet. So two flowers that look the same to us can look completely different to insects. And now, because the flowers look the same to us, do we say the insects are getting it wrong? Because these flowers evolved with the insects to give signals to the insects in the ultraviolet to tell them which flower it is. So it's clear the insects are getting it right, and we just can't see the difference. And that's another way of thinking about adversarial examples. It looks, you know, this thing that it says is an ostrich looks like a school bus to us. But actually, if you look in the texture domain, then it's actually an ostrich. So the question is, who's right? And in the case of the insects, just because two flowers look identical to us, it doesn't mean they're really the same. The insects are right about them being very different. In that case, it's different parts of the electromagnetic spectrum that are indicating the difference that we don't pick up on. But it could be... In the case of image recognition for our current neural nets, so you could argue maybe that

Abbeel 1:05:53
since we build them and we want them to do things for us in our world, then we really don't want to just say, okay, they got it right, and we got it wrong. I mean, they need to recognize the car and the pedestrian. Yeah, I agree. I just want to show it's not as simple as you might think about who's right and

Hinton 1:06:09
who's wrong. And part of the point of my GLON paper was to try and build perceptual systems that work more like us. So they're much more likely to make the same kinds of mistakes that we do. So they're much more likely to make the same kinds of mistakes as us and not make very different kinds of mistakes. And obviously, if you've got a self-driving car, for example, if it makes a mistake that any normal human driver would have made, that seems much more acceptable than making a really dumb mistake from our movie.

Abbeel 1:06:46
So, Jeff, as I understand it, sleep is something you also think about. Can you say a bit more?

Hinton 1:06:55
Yes, I often think about it when I'm not sleeping at night. So there's something funny about sleep, which is animals do it. Fruit flies sleep. And it may just be to stop them flying around in the dark. But if you deprive people of sleep, then they go really weird. Like if you deprive someone for three days, they start hallucinating. If you deprive someone for a week, they'll go psychotic and may never recover. These are nice experiments done by the CIA, I think. And the question is why? Why do we... What is the computational function of sleep? There's presumably some pretty important function for it. If depriving you of it makes you just completely fall apart. And so current theories are things like it's for consolidating memories, or maybe for downloading things from hippocampus into cortex, which is a bit odd since they had to come through cortex to get to the campus in the first place. So a long time ago, in the early 80s, Terry Sonofsky and I had this theory called Pulse Machines. And it was partly based on an insight of Francis Crick when he was thinking about Hopfield nets. Francis Crick and Graham Mitcherson had a paper about sleep. And the idea that you would hit the net with random things and tell it not to be happy with random things. So in a Hopfield net, you give it something you wanted to memorize, and it changes the weight so the energy of that vector is lower. And the idea is if you also give it random vectors and say, make the energy higher, the whole thing works better. And that led to Boltzmann machines, where we figured out that if you instead of giving it random things, you get things generated from a Markov chain, the model's own Markov chain, and you say make those less likely and make the data more likely, that is actually a maximum likelihood learning. And so we got very excited about that, because we thought, okay, that's what sleep is for. Sleep is this negative phase of learning. It comes up again now in contrastive learning, where you have two patches from the same image you try and get them to have similar representations, and two patches from different images, you try and get them to have representations that are sufficiently different. Once they're different, you don't make them any more different, but you stop them being too similar. And that's how contrastive learning works. Now with Boltzmann machines, you couldn't actually separate the positive phase from the negative phase. You had to interleave positive examples and negative examples, otherwise the whole thing would go wrong. And I went to, I tried a lot not interleaving them. It's quite hard to do a lot of positive examples followed by a lot of negative examples. What I discovered a couple of years ago that got me very excited and caused me to agree to give lots of talks with that, that I then cancelled when I couldn't make it work better, was that with contrastive learning, you can actually separate the positive and negative phases. So you can do lots of examples of positive pairs, followed by lots of examples of negative pairs. And that's great, because what that means is, you can have

something like a video pipeline, where you're just trying to make things similar while you're awake, and trying to make things dissimilar while you're asleep. If you can figure out how sleep can generate video for you. So it makes a contrastive learning algorithm much more plausible if you can separate the positive and negative phases and do them at different times and do a whole bunch of positive updates, followed by a whole bunch of negative updates. And even for the standard contrastive learning, you can do that moderately well. You have to use lots of momentum and stuff like that. There's all sorts of little tricks to make it work, but you can make it work. So I now think it's quite likely that the function of sleep is to do unlearning on negative examples. And that's why you don't remember your dreams. You don't want to remember them, you're unlearning them. Crick pointed this out. You'll remember the ones that are in the fast waits when you wake up, because the fast waits are a temporary store. So that's not unlearning, that still works the same way. But the long-term memory, the whole point is to get rid of those things. And that's why you dream for many hours a night, but when you wake up, you can just remember the last minute of the dream you're having when you woke up. And I think this is a much more plausible theory of sleep than any other I've seen, because it explains why if you got rid of it, the whole system would just fall apart. You'll go disastrously wrong and start hallucinating, doing all sorts of weird things. And let me say a little bit more about the need for negative examples that you have in contrastive learning. If you've got a neural net, and it's trying to optimize some internal objective function, something about the kinds of representations it has, or something about the agreement between contextual predictions and local predictions, it wants this agreement to be a property of the real data. And the problem inside a neural net is that you might get all sorts of correlations in your inputs. I'm a neuron, right? So I get all sorts of correlations in my inputs, and those correlations have nothing to do with the real data. They're caused by the wiring of the network and the weights in the network. If these two neurons are both looking at the same pixel, they'll have a correlation, but that doesn't tell you anything about the data. And so the question is, how do you learn to extract structure that's about the real data and not about the wiring of your network? And the way to do that is to feed it positive examples and say, find structure in the positive examples that isn't in the negative examples, because the negative examples are going to go through exactly the same wiring. And if the structure is not in the negative examples, but it is in the positive examples, then the structure is about the difference between the positive and negative examples, not about your wiring. So as people don't think about this much, but if you have powerful learning algorithms, you better not make them learn about the neural network's own weights and wiring. That's not what's interesting. Now, when you think about people who don't get sleep then and start hallucinating, is

Abbeel 1:13:19
hallucinating just because they're not aware of the fact that they're not aware of

the fact that they're not aware of the fact that they start hallucinating. You're just doing it while you're awake. Obviously you can have little naps and that's very helpful. And maybe hallucinating

Hinton 1:13:33
when you're awake is serving the same function as sleep. And it's, I mean, all the experiments I've done say it's better to not have 16 hours of waking, eight hours of sleep. It's better to have a few hours of waking, few hours of sleep. So and a lot of people have discovered that little naps little naps help. Einstein used to take little naps all the time, and he did okay.

Abbeel 1:13:54
Yeah, he did very well. Now, there's this other thing, Lihua, you've brought up, this notion of student beats teacher. What does that refer to?

Hinton 1:14:08
Okay. So, a long time ago, I did an experiment on MNIST, which is a standard digit database for recognising digits, where you take the data, the training data, and you corrupt it. And you corrupt it by substituting the wrong label, one of the other nine labels, 80% of the time. So, now you've got a data set in which the labels are correct 20% of the time, and wrong 80% of the time. And the question is, can you learn from that? And how well do you learn from that? And the answer is you can learn to get like 95% correct on it. So now you've got a teacher who's wrong 80% of the time, and the student is right 95% of the time. So, the student is much, much better than the teacher. And this isn't, each time you get an example, you corrupt it. You take the training examples, you corrupt them once and for all. So, you can't average away the corruption over different, you might be able to average it away over different training cases that haven't had similar images. But, and if you ask, well, how many training cases do you need if you have corrupted ones? And this was of great interest because of the tiny images data set some time ago, where they had 80 million tiny images with a lot of wrong labels in. And the question is, would you rather have a million things that are flakily labeled, or would you rather have 10,000 things with accurate labels? And I had a hypothesis that what counts is the amount of mutual information between the label and the truth. So, if the labels are correct, corrupted 90% of the time, there's no mutual information between the labels and the truth. If they're corrupted 80% of the time, there's only a small amount of mutual information. That's what you think. I think it's about, my memory is it's 0.06 bits per case. Whereas if it's uncorrected, it's about 3.3 bits per case. So, it's only a tiny amount. And then the question is, well, suppose I balance the size of the training set by putting as much mutual information in there. So, if there's like a 50th of the mutual information, I have 50 times as many examples, do I now get the same performance? And the answer

is, yes, you do to within a factor of two. I mean, the training set actually needs to be twice that big. But roughly speaking, you can see how useful a training example is by the amount of mutual information between the label and the truth. And I noticed recently you have something for doing SIM to real, where you're labeling real data using a neural net, and those labels aren't perfect. And then you take the student that learned from those labels and the student is better than the teacher it learned from. And people are always puzzled by how could the student be better than the teacher? But in neural nets, it's very easy. The student will be better than the teacher if there's enough training data, even if the teacher is very flaky. And I have a paper a few years ago with Melody Gouhan about this for some medical data. The first part of the paper talks about this. But the rule of thumb is basically what counts is the mutual information between the assigned label and the truth. And that tells you how valuable a training example is. And so you can make do with lots of flaky ones.

Abbeel 1:17:38
That's interesting. Now, in the work we did that you just referenced, Jevin, and the work I've seen quite popular recently, usually the teacher provides noisy labels. But then not all the noisy labels are used. There's a notion that only look at the ones where the teacher is more confident. Your description doesn't really care about that.

Hinton 1:18:02
That's obviously a good hack. But you don't need to do that. You don't need to do that. It's a good hack. And it probably helps to only look at the ones where you have reason to believe the teacher got it right. But it'll work even if you just look at them all. And there's a phase transition. So with MNIST, Melody plotted a graph. And as soon as you get like 20% of the labels right, your student will get like 95% correct. But as you get down to about 15% right, you suddenly get a phase transition where you don't do any better than chance. Because somehow the student has to get it. The teacher is saying these labels. And the student has to, in some sense, understand which cases are right and which cases are wrong and sort of see the relationship between the labels and the inputs. And then once the student's seen that relationship, a wrongly labeled thing is just very obviously wrong. So it's fine if it's randomly wrongly labeled. But there is a phase transition where you have to have it good enough so the students sort of get the idea. But that explains how our students are all smarter than us.

Abbeel 1:19:06
We all need to get a small fraction of the time.

Hinton 1:19:10
Right. And I'm sure the students do some of this data curation where you say something and the student thinks, oh, that's rubbish. I'm not going to listen to that. Those are the very best students you met.

Abbeel 1:19:21
Yeah, those are the ones that can surprise us. Now, one of the things that is really important in neural net learning and especially when you're building models is to get an understanding of what is it learning. And often people try to somehow visualize what's happening during learning. And one of the most prevalent visualization techniques is called TSNI, which is something you invented, Jeff. So I'm curious, how did you come up with that? Maybe first describe what it does and then what's the story behind it?

Hinton 1:19:56
So if you have some high dimensional data and you try and draw a 2D or a 3D map of it, you could take the first two principal components and just plot the first two principal components. But what principal components cares about is getting the big distances right. So if two things are very different, principal components is very concerned to get them very different. In the 2D space, it doesn't care at all about the small differences because it's sort of operating on the squares of the big differences. So it won't preserve similarity very well, high dimensional similarity. And you're often interested in just the opposite. You've got some data, you're interested in what's very similar to what. And you don't care if it gets the big distances a bit wrong as long as it gets the small distances right. So I had the idea a long time ago that what if we took the distances and we turned them into probabilities of pairs, there's various versions of t-stances, but suppose we turned them into the probability of a pair such that we say pairs with a small distance are probable and pairs with a big distance are improbable. So we're converting distances into probabilities in such a way that small distances correspond to big probabilities. And we do that by putting a Gaussian around a point, a data point, and computing the density of the other data point under this Gaussian. And that's an unnormalized probability. Then you normalize these things. And then you try and lay the points out in 2D so as to preserve those probabilities. And so it won't care much if two points are far apart, they'll have a very low pairwise probability. And it doesn't care the relative positions of those two points. What it cares about the relative positions of ones with high probabilities. And that produced quite nice maps. And that was called stochastic neighbor embedding because we thought of this, you put a Gaussian and you stochastically pick a neighbor according to the density under the Gaussian. And I did that work with Samurais. And it had very nice simple derivatives, which convinced me that we were on to something. And we got nice maps, but they tended to crowd things together. And there's obviously a basic problem in converting high dimensional data into low dimensional data.

So SNEE tends to crowd things together, stochastic neighbor embedding. And that's because of the nature of high dimensional spaces and low dimensional spaces. In a high dimensional space, a data point can be close to lots of other points without them all being too close to each other. In a low dimensional space, they all have to be close to each other if they're all close to this data point. So you've got a problem in embedding closenesses from high dimensions to low dimensions. And I had the idea when I was doing SNEE that since I was using probabilities as this kind of intermediate currency, there should be a mixture model, there should be a mixture version where you say in high dimensions, the probability of a pair is proportional to e to the minus s squared distance on the Gaussian. And in low dimensions, suppose you have two different maps. The probability of a pair is the sum of e to the minus the distance in the first 2D map and e to the minus the squared distance in the second 2D map. And that way, if we have a word like bank, and we're trying to put similar words near one another, bank can be close to greed in one map and can be close to river in the other map without river ever being close to greed. So I really pushed that idea because I thought this is a really neat idea and you can have a mixture of maps. And we managed to get it to work. Ilya was one of the first people to work on that. And James Cook worked on it a lot. And several other students worked on it. And we never really got it to work well. And I was very disappointed that somehow I'm being able to make use of the mixture idea. And then I went to a simpler version, which I called Unisni, which was a mixture of a Gaussian and a uniform. And that worked much better. So the idea is in one map, all pairs are equally probable. And that gives you a sort of background probability, which comes through the big distances, a small background probability. And then in the other map, you contribute a probability proportional to your square distance in this other map. But it means in this other map, things can be very far apart if they want to be, because the fact that then they need some probability is taken care of by the uniform. And then I got a review paper from a plumber called Lawrence Vander Maarten, which I thought was actually a published paper because of the form it arrived in, but wasn't actually a published paper. And he wanted to come do research with me. And I thought he had this published paper, so I invited him to come do research. And it turned out he was extremely good. And it's lucky I'd been mistaken in thinking it was a published paper. And we started on Unisni. And then I realized that actually Unisni is a special case of using a mixture of a Gaussian and a very, very broad Gaussian, which is a uniform. So what if we use a whole hierarchy of Gaussians? Many, many Gaussians with different widths. And that's called a t-distribution. And that led to t-Sni. And t-Sni works much better. And t-Sni has a very nice property that it can show you things at multiple scales because it's got a kind of 1 over d squared property that once distances get big, it behaves just like gravity and clusters of galaxies and things. You have clusters of galaxies and galaxies and clusters of stars and so on. And you get structure at many different levels in it. You get the coarse structure and the fine structure all showing up. Now the objective function used for all this, which was the sort of relative densities under a Gaussian, came from other work I did with Alberto Paganaro earlier that we found hard to get published. I got a review saying, yeah, I got a review of that work when it was rejected by some conference saying Hinton's been working on this idea for seven years and nobody's interested. I take those reviews as telling me I'm onto something very original. And that actually had the function in it that's now used, I think it's called NCE. It's used in these contrastive methods. And t-Sni is actually a version of that function. But it's being used for making maps. So it's a very long history of t-Sni, of getting the original SNI and then trying to make a mixture version and it's just not working and not working and not working. And then eventually getting the coincidence of figuring out it was a t-distribution of what you wanted to use. That was the kind of mixture. And Lauren's arriving and Lauren's was very smart and very good programmer. He made it all work beautifully.

Abbeel 1:27:12
This is really interesting because it seems a lot of the progress these days, the bigger idea plays a big role, but here it seems it was really getting the details right, was the only way to get it to fully work.

Hinton 1:27:28
You typically need both. You have to have a big idea for it to be interesting original stuff, but you also have to get the details right. And that's what graduate students are

Abbeel 1:27:39
for. So Jeff, thank you. Thank you for such a wonderful conversation for our part one of our season finale.