SDS PODCAST EPISODE 767: OPEN-SOURCE LLM LIBRARIES AND TECHNIQUES, WITH DR. SEBASTIAN RASCHKA

Show Notes: http://www.superdatascience.com/767

Jon: 00:00:00 This is episode number 767 with Dr. Sebastian Raschka, Staff Research Engineer at Lightning AI. Today's episode is brought to you by the DataConnect Conference, and by Data Universe, the out-of-this-world data conference.

00:00:19 Welcome to the Super Data Science Podcast, the most listened-to podcast in the data science industry. Each week, we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today, and now, let's make the complex simple.

00:00:50 Welcome back to the Super Data Science Podcast. We've got another incredible guest for you today, and that's Dr. Sebastian Raschka. Sebastian is Staff Research Engineer at Lightning AI, the company behind the popular PyTorch Lightning, open-source library for training and deploying PyTorch models, including LLMs with ease. He's also author of the mega bestselling book, Machine Learning with PyTorch and Scikit-Learn. He's an iconic technical blogger with 50,000 subscribers and a really well-known technical social media contributor. He has over 350,000 combined followers, 350,000 across LinkedIn and Twitter. He was previously Assistant Professor of Statistics at the University of Wisconsin-Madison, and he holds a PhD in statistical data mining from Michigan State.

00:01:36 Sebastian's latest book, Machine Learning Q and AI, cute name, is available for pre-order now, and early chapters of his 2025 book, Build A Large Language Model (From Scratch) are also available online. As a treat for you, I will personally ship 10 physical copies of any of Sebastian's books to people who comment or reshare the LinkedIn post that I publish about Sebastian's episode from my personal LinkedIn account today. Simply mention in your comment or reshare which book you'd like. Is it ML with PyTorch and Scikit-Learn, Machine learning Q&A, Machine Learning Q and AI or Build an LLM (From

Scratch)? I'll hold a draw to select the 10 book winners next week. So you have until Sunday, March 24th to get involved with this book contest.

00:02:23 Today's episode is technical and will primarily be of interest to hands-on practitioners like data scientists, software developers, and machine learning engineers. In the episode, Sebastian details the many super helpful open-source libraries that PyTorch Lightning leads development of, the LLM related techniques and models that he's most excited about today, including DoRA parameter-efficient fine-tuning, Google's open-source Gemma models, multi-query attention, and alternatives to RLHF. He'll also fill us in on where he sees the next big opportunities in LLM development. All right, you ready for this most excellent episode? Let's go.

00:03:02 Sebastian, welcome to the Super Data Science Podcast. It's awesome to have you here. I've been trying to get you on the show for so long. It's so exciting for me to finally have it happening. How are you doing today? Where in the world are you calling in from?

Sebastian: 00:03:15 Thanks for the kind invitation, Jon, and it's my pleasure to be here. It's one of the biggest podcasts, I would say, and I was just waiting for the right moment to be on the show, and that happens to be today. So I'm currently in my simple office at home in Madison, Wisconsin. It's near Chicago, and I'm super looking forward to the Super Data Science Podcast.

Jon: 00:03:43 Nice. No doubt our audience is jazzed to have you here. I want to jump right into the technical stuff because there's so much to cover with you. It's amazing. So a lot of people know you from your book, Machine Learning with PyTorch and Scikit-Learn. There's a lot of ways that they could know you because you're also an iconic blogger. Everyone on my data science team is very familiar with

you from your highly technical posts and you sharing them on social media. You have a huge social media following, but you also, in addition to the book that you already have out, the Machine Learning with PyTorch and Scikit-Learn, you also have new books that are in the works. So the first one that's coming out that is actually available for pre-order now from No Starch Press, the book is Machine Learning Q and AI, which is a very cute name. There's people give Germans a hard time for not being very funny, but that is a very clever book title, Sebastian.

Sebastian: 00:04:42 Thank you for that. I was thinking very hard about this to make it interesting. It's essentially a Q&A book, a question and answer book, and it happens to be about AI, so hence the name Q and AI. This book has been a passion project of mine for a long time. So you mentioned, I'm writing a lot, I'm teaching a lot. I'm coding a lot, and there are certain topics that I often want to talk about, but they never really fit into other materials because either it's too short to have a standalone big article about it or it is too long to fit it into a semester's curriculum. So I thought, "Okay. Why don't I just collect all these interesting topics and questions and things and write a book about that?" So it's essentially 30 questions and answers about topics related to machine learning, deep learning, and AI.

00:05:40 It's really a wide range from what are techniques to avoid overfitting to what are the different multi-GPU training paradigms, how to evaluate large language models. There's a little bit of everything in there. It's these topics that I couldn't cover elsewhere. So it's basically almost outtakes, but in a good way, of other projects. So 30, I would say, fast-paced chapters. In that book, it's my first and only book where I don't have any code in the book. It's more like to make it, I would say, timeless, so it doesn't need to, let's say, get updated regularly. There is a

supplementary GitHub repository though with code examples.

00:06:26 My goal for that book was essentially to if you are traveling, you're on a plane or you sit in your favorite chair and drink your morning coffee, it's like that book that you can read without, let's say, stressing out about the code. You just read it to absorb, let's say, new knowledge. There are a few exercises in there though, so if you want to have, let's say, a little workout, there is also the exercises and solutions also in there, but essentially, it's, I would say, shorter, but pretty interesting book for those who are interested in the more intermediate to advanced topics in machine learning. So it's not for beginners, it's more the intermediate, I should say.

Jon: 00:07:10 That was my next question exactly, and I'd pieced that together from, well, looking at even just the table of contents and you mentioning that there's things like multi-GPU training paradigms is one of the 30 topics, one of the questions answered, so that makes it pretty clear that this isn't a book for beginners. So it sounds like something that's perfect for a lot of our listeners. The first chapter is on neural networks in deep learning. The second chapter is on computer vision. The third chapter is natural language processing. Then you get into production and real-world deployment scenarios in chapter four. Then the final chapter, chapter five, is on predictive performance and model evaluations.

00:07:47 So over the course of those five chapters, you have 30 questions. The shortest chapter in terms of number of questions has three, and they get up to about 10 questions in a given chapter. It's a really fun way to break up a book, as you say, into these byte-sized chunks, byte, of course, spelled with a Y, and those byte-sized chunks allow you to if you have 10 minutes or 20 minutes, maybe you can tackle just one of these questions or I imagine

you could also probably, maybe more so than other books, you could jump to a specific question without necessarily needing to have read the questions beforehand.

Sebastian: 00:08:28 That's exactly right. So you don't have to read it in order. You can actually just read it in any order. You can skip chapters. You can start from the last question and work up to the first one, although there is a bit of structure in it where I think with the first chapter, I start explaining what the difference is between representations, embeddings and so forth, and latent space as a general introduction, and that topic comes up later when we talk about other techniques like embeddings and so forth like when we talk about LLMs. So there is a certain sense of structure to it or order, but like you said, it can be read in any order.

00:09:05 One other thing I wanted to mention is also I try to make as many figures as possible because I think that's also useful for learning that it's not just text, it is figures. As a little fun fact, actually, I started writing this book before the first version of ChatGPT came out, and in the very first iteration of the book, I had actually, that was on Leanpub, a comparison between my answers and ChatGPT answers and then highlighting, let's say, what it got wrong and sometimes it was pretty good. I removed this for the final version because it was like, I would say almost like a running gag at the time, but it got a bit lame over the years to do these comparisons. I wanted to say that is something that I felt was very important to add also figures because that is also something that you can't just get from asking at chatbot, at least not yet while we are recording this. Who knows? Maybe next year we have something.

Jon: 00:10:00 Of course, another advantage of illustrations, my book is called Deep Learning Illustrated because I think for a lot

of people, a visual is one of the easiest ways to learn a concept, especially when it's complemented by words and equations. So I think it's a great way to learn, absolutely. Also, I think it's also been great for you when you are making social media posts. That visual can be an iconic anchor for the rest of the post. I think that that's been part of why your posts are so popular and so helpful.

Sebastian: 00:10:41 I think it's also for me a way to think about things. It's like, I don't know, there's something about it where if you think about how to put the figure together, it helps, really. It's a visual anchor for people to identify or see what it is about at a glance. Then also, it is essentially helping you to remember things. It's like if you remember context, it's usually the first thing that comes to mind is an image that is related to this. So I would say there is a reason why there's a saying, "A picture is worth a thousand words," right?

Jon: 00:11:14 Exactly. Exactly. All right. Beyond that book, beyond Machine Learning Q and AI, which we have just been talking about, you also have early chapters of your estimated 2025 release book, Build A Large Language Model (From Scratch), with from scratch in quotes, sorry, not in quote, in parentheses, so Build A Large Language Model (From Scratch). So the early chapters of that are already available. This is being published by Manning, which is a great publisher. They always do fantastic books. I imagine there's going to be a lot of visuals in that as well. What can people already get today? What can our listeners already access today from those early chapters that are available?

Sebastian: 00:12:01 So about that book, right now, this is a work in progress still, where I would say half of it is done. So I've almost all code written. This was my project last summer where I was writing most of the code, and now the fun task is to organize it into, let's say, an educational optimal way to

represent this information. It is essentially about building a large language model one step at a time from starting from the data input to pre-training the model, loading pre-trained weights, and then also fine-tuning, instruction fine-tuning, classification, fine-tuning and alignment with human preference labels. So it's a whole journey from data input to the model architecture of pre-training and the fine-tuning.

00:12:45 Right now, while we are recording this, I'm on chapter five, which gets to the pre-training. So chapter one, the general introduction, chapter two, the input, the tokenizer, chapter three is on masked multi-head attention, and chapter four is then putting together the LLM architecture, which is not pre-trained in chapter five. So it is currently being released one chapter at a time. I would say every one or two months I release a new chapter. So it's quite a lot of work because, like you mentioned, it also has a lot of figures.

00:13:20 So I think for chapter three I had like 26 figures or something. So that is also one of my focus areas to not only have the code, but also to illustrate the code with a lot of figures because LLMs, I would say, they are not super complicated because many, many concepts are repeated within an LLM architecture where you have the transformer block and you just keep repeating it, but it is still a complicated concept, so where you need a lot of figures and illustrations to make that accessible.

00:13:48 With that, I also have a GitHub repository where I also upload the Jupyter Notebooks with the code as it appears in the chapter. So if a listener wants to take a look and see if that's interesting to them, the GitHub repository is open-source available, and I also have notes in there. So the chapter goes into more details, but essentially, all the gist is also on GitHub, I would say.

Jon: 00:14:12 Two days, 50+ presenters, unlimited opportunities to connect with a global data audience, the Annual Data Connect Conference brings together industry leaders, technical experts, and entrepreneurs to discuss the latest trends, technologies, and innovations in data analytics, machine learning and AI, all while amplifying diverse voices in the space. Join us in Columbus, Ohio, July 11th and 12th to hear from incredible speakers like Carly Taylor, Megan Lieu, Olivia Gambelin, and more. Can't make the trip to Ohio? Join us for the Second Annual Data Connect Conference West hosted in Portland, Oregon on May 8th, where you'll hear from AI experts from the likes of Microsoft, Alteryx, and Women Defining AI. Save 15% on your Data Connect pass by registering with the code superdatascience.

00:15:00 Nice. So for our listeners who are interested, how does it work to get access to early chapters? Do you essentially pre-order the book today and that provides you access to whatever chapters you already have done and drafted?

Sebastian: 00:15:13 That is a good question. So that is through Manning. They have early access programs. So for those who pre-ordered the book, it's like a online pre-order version. I think as far as I know, you can get a digital version or you can get a version with the print pre-order, so you get both the e-version, and once the book is finished, there will be also the hardcover. For that, so you mentioned 2025 as the estimated release, so according to the publisher, that's correct, but I think ... So that's usually the timeline from finishing to print is usually a few months. So the estimated timeline for the last chapter is approximately August, September this year, so not too far away.

Jon: 00:16:01 Oh, wow.

Sebastian: 00:16:01 Then the release will be ... So it'll be all available, I hope if things go as planned, by August, September online as the

digital version, but then the final, let's say, production-ready book will be 2025 because there are usually a few steps from, let's say, the manuscript to the layout and final draft, but people who want to follow along could maybe check out the GitHub repository to see if that's for them and if they're interested in that.

00:16:30 I'm having a lot of fun writing this book because I like coding a lot, so that's like little passion project. For that, it's also this little journey from really taking the same LLM from pre-training to fine-tuning and so forth. That is quite interesting, I think. So I think it's also one of the greatest ways to learn about LLMs because there are a lot of articles that try to explain LLMs, and it is, to some extent, possible to explain what an LLM is doing and how it works, but there is a certain, I would say, clarity that comes when you actually code it from scratch, where you really see what goes in there, what happens inside, and what comes out there and how does it behave if you tweak a few things. So I think that is a good way to learn and also a lot of fun, for me at least. I hope for the readers as well.

Jon: 00:17:21 Sebastian, it sounds like one of those books that I am going to have to work through, I have so little time on top of my day job and hosting this podcast and content creation. Every guest that we have on, we ask for a book recommendation. Many guests that we have on are incredible authors writing amazing books, and even they'll send me copies or give me digital copies, and I want to read all of these things, but there's just I can't possibly make time to read everything that I would like to. This seems like one of those books that I will have to make time to do because knowing how well you package things up and knowing how much more there is for me to grasp around LLMs and transformer architectures, this seems like the perfect way to go. So I'm really looking forward to this coming up.

Sebastian:	00:18:11	Well, I appreciate it. I should send you a maybe personally signed copy then to just make sure you read it.
Jon:	00:18:17	Sure. I won't stop you, for sure. Happy to. Would be happy to do one of the quotes at the front if you want.
Sebastian:	00:18:28	We'll see if I can come up with a creative one. If not, I will ask the LLM, I guess.
Jon:	00:18:35	No, no, sorry, I meant ... Nevermind.
Sebastian:	00:18:38	Oh, I see what you mean, the cover code.
Jon:	00:18:41	Yeah, I forget what those are called. The endorsement, I guess is the word.
Sebastian:	00:18:45	Testimonial, endorsement, something like that.
Jon:	00:18:46	Testimonial, that's the best word. There you go. That's great. English is my first language and you're crushing me on it. So yes, beyond your book writing, which is obviously heavyweight, you're an absolutely world-class publisher, a writer of books in this space, so that in and of itself could be conceivably a full-time job, but you actually have a full-time job. You have a day job, which is at Lightning AI. So Lightning AI is the company behind PyTorch Lightning, and you have a staff research engineer role at Lightning AI. So maybe tell us a bit more about what Lightning AI does and what your work involves there as a staff research engineer.
Sebastian:	00:19:37	So at Lightning AI, I am currently a staff research engineer where I work mostly on open-source and research, the intersection between open-source research and LLMs. So about Lightning AI. We are a company, a startup company headquartered in New York City, and we have also a lot of meetups there, so if you are ever interested or in the area. We have multiple projects. Let's

start maybe at the open-source, on the open-source front. So PyTorch Lightning is essentially an open-source library on top of PyTorch, and it is this library that really makes PyTorch more convenient, I would say, where it is not really, I would say, changing your model.

00:20:20 So typically, what you do is you define a module that you put ... So you put a PyTorch model into this module, and then you can use a trainer. So the trainer, you may be familiar with the concept of a trainer from other packages, but that's like the original trainer in PyTorch Lightning where it provides you with a lot of free goodies. So once you use this trainer, you get something like checkpointing, logging, easy access to different loggers, also available ones like weights and biases, but then also with one line of code, you can change whether you want to use one or four GPUs, eight GPUs, multi-GPU training versus multi-node training, different precisions, mixed precision training, and so forth. So these little things where it is a lot of code if you would write this out in PyTorch, so it's like a lot of boilerplate code and PyTorch Lightning basically packages that for you so that you don't have to, let's say, repeat or redo all that long code. You have a way of organizing your code in a more concise way. So that is PyTorch Lightning.

00:21:26 So the other project is called Fabric. Fabric is essentially the engine that is used for multi-GPU training that plugs in, for example, DeepSpeed, FSTP from from PyTorch. It's essentially this project that is with a few lines of code, you can manipulate PyTorch code. I have a few tutorials I could maybe share where it's really just five lines of code you change, and then you can easily access more advanced multi-GPU paradigms and so forth.

00:21:55 This is especially useful when you also work with large models like LLMs, where you need to shard the models across multiple GPUs because they're too large to fit on a

single GPU. If you want to, let's say, lower the precision from 32-bit to 16-bit or beef load 16 -bit or mixed precision with 30-bit and 16-bit and so forth or even quantization. So that is where you can do that with a few lines of code instead of just changing 20 or 30 lines in your code where it becomes a bit hard if you want to just do comparisons. So that is basically to make our life also easier as researchers when we work on things so we can focus more on, I would say, the model itself rather than the training loop and so forth.

00:22:40 The third project is Lit-GPT. So Lit-GPT is using PyTorch Lightning and Fabric, and this is an open-source library to basically have an implementation of LLMs that is somewhat human-readable. So it is essentially mostly scripts or script-based, but it is meant to be easy to maintain and easy to read. For example, there are lots of LLM libraries out there, but as the researcher, when I try to, let's say, use them, they work great as a user, but then if you want to change a few things about the architecture, it becomes a bit more tricky because there are so many files and so many dependencies and a lot of code, basically, that you would have to understand to make changes, and that was meant more for, I would say, as a toolbox for us internally to experiment with LLMs. It's also open-source if others want to use it.

00:23:36 It was, for example, one of the most widely used packages also in our repositories in the LLM efficiency challenge at NeurIPS in 2023, but I would say it's not reinventing the wheel in terms of LLMs because it's essentially about loading existing LLMs like Llama, Falcon, and so forth. So you can use other LLMs in there, but it is like this certain way of making it hopefully a bit more easy to experiment with. So when you do something like LoRA, QLoRA, fine-tuning and so forth, that is all on the open-source side. That is usually what we use ourselves when we work on research projects, for example.

00:24:19 Then because we are a company, so on the, let's say, money-making side, we have this new platform. It's called Lightning Studios, and it's essentially a platform that I also use myself a lot for research. So basically, all my research is done on that platform and it would almost be easier to show it, but let me try to explain it in words. So it's essentially think about just when you do experiments on your computer, you use a visual studio code, for example, or another editor or Jupyter Notebooks and so forth. It's all that in the browser, basically. So you have similar to Google Colab, there is a Jupyter Notebook, but you can also use Visual Studio Code, which is usually what I use most of the time. So you can switch between those two.

00:25:08 Then really, the big feature is you do all the coding in the browser, but you can, if you need then, just simply with a few clicks switch the hardware. For example, my workflow is usually that I use a CPU machine for debugging or just setting up my code because CPU machines are free or cheap, so you don't waste your money doing coding, basic coding or debugging on an expensive A100 or H100 or something like that. Once I am confident that my code works, I just switch to the A100, H100, for example, and run my experiments. Then when the experiments are done, I can switch back to my CPU and analyze the results. So it's this smooth switching between hardware that makes it really useful for experiments by use, save a lot of money by not running on the expensive GPU all the time, basically.

00:26:01 So just to get it back also to the Google Colab experience, so it's basically like Google Colab, so just to make it maybe to have an analogy, but you can use multiple GPUs instead of just one, so you can also do multi-node training. The big difference is when you switch the hardware, you don't have to reinstall anything, so you don't have to install your packages again and your data is

also all there. So you basically have the same environment. You just switch the computer hardware, which is super useful if you run experiments.

Jon: 00:26:31 That's very cool. I do workshops several years running now at the Open Data Science Conference. Typically, at least in a given year, I'll do either east in Boston or West in San Francisco. This year, I'm doing East, just like I did last year. At last year's East, for the first time, I did a generative AI training and we used PyTorch Lightning, of course, in the training. It was a key library for us, but I ran into this limitation. When I wanted to teach multi-GPU, I had to leave Colab. All of the rest of my demos are Jupyter Notebooks available in GitHub that in the class I run in Colab in real time and so the students can as well. For anyone who's interested I'll include in the show notes, we edited and published the recording of me doing this generative AI training at ODSC East last year. So anyone can access that for free on YouTube.

00:27:32 But the main point was that it was inconvenient to have to switch to command line tools in order to be able to do the multi GPU stuff, although not a bad thing for people to learn as well. However, it's super cool to hear this, that Lightning Studios would allow me to have multi GPUs running in a Colab kind of environment. So I will now experiment with Lightning Studios for my ODSC East training coming up in April this year.

Sebastian: 00:28:00 If you have this workshop again, let me know. So I think this would be definitely the tool then that could make life simpler. And also you mentioned the command line. Yeah, so I do also still a lot of command line coding and you can do that also in that studio basically. So it's essentially like your computer. You have a terminal, you have Visual Studio Code, Jupiter lab, so all the things are there. And one thing about teaching is also I'm also... It's relatively new. We just launched that last month or two

months ago. But next time I'm teaching, what's also nice is that you can share this with people. So you can share Google Colab, you can share the whole environment so you don't have to spend time on the setup basically, because everyone can just duplicate your studio and gets the whole file environment set up and can also experiment and run the code themselves if they want to.

00:28:46 And there is also, if you want to experiment, there is a live coding feature where you can invite participants and you can also edit the same file for pair programming, which I did the other day with a colleague where I had it back in my code. We were working on some fine tuning experiments. So that was something also very useful for that because we also... We are a remote company. We have our headquarter in New York, but then sometimes it would be convenient if you are really, like you would sit together on a desk to pair program on a code file for example.

Jon: 00:29:20 This episode is brought to you by Data Universe, coming to New York's North Javits Center on April 10th and 11th. I myself will be at Data Universe providing a hands-on Generative AI tutorial, but the conference has something for everyone: Data Universe brings IT ALL together, helping you find clarity in the chaos of today's data and AI revolution. Uncover the leading strategies for AI transformation and the cutting-edge technologies reshaping business and society today. Data professionals, businesspeople and ecosystem partners — regardless of where you're at in your journey — there's outstanding content and connections you won't want to miss out on at Data Universe. Learn more at datauniverse2024.com.

00:29:59 For sure. It's something that we're starting to see more of these platforms enable that collaboration, the kind of thing that you expect when you're working on a Google Doc or Google Slides or Google sheet, where you can see

the person there and you can collaborate together. It's cool that you guys have figured that out for Lightning Studios as well. So Lightning Studios, to dig into that one a little bit more... I'm going to go back through these different products, open-source libraries projects that Lightning AI is involved with. So with Lightning Studios, is that running on the cloud or does somebody hook that up to their own infrastructure or does it support both?

Sebastian: 00:30:44 It's a good question. So the default, as a user, it would be running in the cloud. So you pay per usage. It depends on which GPU you use, but A100s are, for example, cheaper than H100s, but they're both more expensive than A10Gs, for example. So you pay hourly basically, but you can also connect your own cluster or own hardware. We currently only do that with select bigger companies, but we might also open that up more to individuals. But if you have, let's say a cluster, you're a university or a company, you can contact us and we can help you also to set this up for your cluster. Yeah, that is something a lot of people already have a lot of hardware, so in that sense it would make sense to have this nice interface sitting on top of that.

00:31:34 And one feature I also should mention is that you can submit jobs, like you can submit jobs on Swarm or something like that where you don't have to run this interactively. You can essentially experiment with your code and then select the machine you want to run the jobs on and just submit hyperparameter tuning sweeps, for example. So in that sense, I think if you have a cluster, that makes a lot of sense where this could then manage also... If you have multiple people working on it, like this allocation of the resources essentially. So we do allow that to currently select customers because it requires some extra setup steps. But yeah, that is something that is also I think super cool and super useful.

Jon: 00:32:13 Fantastic. I mean, I guess actually what I was hoping for was it was going to be the cloud-based approach that you described first, where you can hourly, where you can pick a GPU size. So then in that sense it is again kind of really like Colab. I mean Colab has a freemium tier with just your Gmail address or whatever, which is CPU only or slow GPUs. But this, to me, Lightning Studios sounds great because it's so easy to use. You can just bring your code into this environment and there's no setup headaches. I think that that's brilliant. I love that Lightning AI is doing that.

00:32:53 Going back to the project that you mentioned before, Lightning Studios, which is Lit-GPT, you described how the idea here is for it to be a human-readable LLM. So could you go into a bit more detail for me on what that means? Maybe go through a typical kind of user story where that human-readable element is part of the story?

Sebastian: 00:33:18 Yeah, that's a good point. So I should clarify human readable, maybe more for the developer rather than the user. So it's like the focus is to have more minimal code. So originally this project originated from Andre Karpathy's nanoGPT. So that was a repository for GPT-2. And Lit-GPT originated from that in that sense that this is more like a minimal implementation of LLMs compared to, let's say, Hugging Face transformers, which is a whole library that is huge and which is awesome. They have a lot of features, but if you want to experiment with an LLM, it is a bit more intimidating. So I think the user interface to Hugging Face transformers is great. It's easy to use, but then if you do research and you want to implement, for the first time, multi-query attention instead of regular multi-head attention, that is I think a bit trickier then.

00:34:18 And so Lit-GPT is more like, I would say minimal, in the sense that keeping the lines of code small and also

keeping the number of files small so that it is easier to inspect. Now I must say it started out very simple. It was actually called Lit-Llama, where it was only focused on the Llama architecture by Meta, but then it grew a bit into Lit-GPT A bit more complicated. So we still have both, we have Lit-Llama, which is I would say if you only care about Llama models, I would use that one because it's a bit even simpler. But since we then started to support also other models like Falcon and MPT, Pythia, and right now, I'm working on OLMo and Gemma. So since we are also supporting all these other types of models, you also have to make a few more modifications to these. For example, if this model, then do that, and for example, instead of multi-head attention use, multi query attention. And so in that sense it is a bit more complicated, but the aim is on minimalism, to keep things simple.

00:35:22 And then also for the fine-tuning, there are the normal full [inaudible 00:35:28] fine tuning, but also LoRA, QLoRA adapters, and I wanted to also add DoRA, which is this new weight decoupled LoRA version, which is very promising. So in that sense, it's a platform to easily do these things, to add these things. I mean " easy" in quotation marks because it is still not super trivial. It requires still work. So it's not for free in terms of you still have to work on it. But it is open-source and if you like this type of approach, it's just a slightly different approach. I think that might be a useful project to check out.

00:36:06 So people like that a lot when they worked on the NeurIPS efficiency challenge. But of course, there are so many other options too. I think what's cool though is that each library or repository has a completely different approach. There is the Hugging Face transformers, which is like a Python session basically that you run. Lit-GPT is more script based. Then there's something like Xlotl, for

example, where it's more configuration file based. And so you have all these different types and I think, I wouldn't even say one is better than the other. It really depends on the project.

00:36:39 And what's also nice is of course you can save the checkpoints and load them in the other libraries. The checkpoints we use in Lit-GPT are, for example, most of them from the Hugging Face model hub. So you can convert them back and forth. So it's not like... It doesn't lock you in. You can basically choose whatever tool you like for or need for your current project essentially.

Jon: 00:36:59 When you say human-readable, it isn't that the developer is literally writing natural language. It's that there's less code. It's easy to read code.

Sebastian: 00:37:11 Yeah, that's maybe the way to put it. So it's like DearPy or something like that where it's a human language format that goes into the LLM. It's more really like the code. Yeah, if you heard of nanoGPT by Andre Karpathy, it's basically like that, but with more advanced... It grew out of that, but it has support for other LMSs and also more optimization basically, like a multi GPU support, quantization, and these types of things. But in that sense, readable that it is minimal. But yeah.

Jon: 00:37:45 Very cool. Yeah, it sounds like a fantastic project for people to explore. It sounds like yet another thing that you've mentioned on the show that I feel like I need to be checking out now. Before Lit-GPT, you were talking about Fabric, which also sounds interesting. So this allows you to have advanced multi GPU tools for training and deploying, I imagine also as well, your LLMs. Now that description that I just gave that has a lot of overlap with PyTorch Lighting. So PyTorch Lighting is, of all of these projects that you've discussed, PyTorch Lighting is the only one that I have at this time used. Tomorrow, that

might be a very different story, but today it's just PyTorch Lighting. And so one of the great things for me about PyTorch Lightning is, yes, you mentioned the easy model training, but even, for example, my ODSE training from last year, I am using PyTorch Lightning code for multi-GPU deployment. And that is even one of the things that you mentioned that it's great for.

00:38:48 So I guess what is the key difference between Fabric and PyTorch Lightning? What kinds of advanced things can I be doing on multiple GPUs with Fabric that I can't do with PyTorch Lightning? And it seems interesting to me that Lightning would even make it a separate project. So I don't know, maybe you have some insight into why.

Sebastian: 00:39:11 So I don't want to destroy your excitement here, but actually you asked what can you do with Fabric that you can't do with PyTorch Lightning? And the boring answer is nothing. So the reason is the explanation now is so if you're already using PyTorch Lightning, there is no need to use Fabric because you already have all the cool things available in PyTorch Lightning. Now, the reason why Fabric exists is some people don't need, let's say everything in PyTorch Lightning. PyTorch Lightning is a very big library and it does a lot of things for you. And Fabric would be more like the minimalistic approach to that. So in PyTorch Lightning, if you use PyTorch Lightning, you have to write this Lightning module where you define how to do a forward pass, a backward... backward is done for you, but the forward and the steps, evaluation steps, and so forth. So you have this lightning module and then you plug this lightning module into the trainer.

00:40:10 Now if you have, let's say, existing PyTorch code, you would need to kind of slightly reformat this. It's not, I would say a lot of work, but some people prefer something in between where let's say they can keep the original

PyTorch code as it is, like the training loop, and just have a very, very minimal addition of code like three, four or five lines of code. And that is what Fabric is for. So before you commit, let's say if you're a PyTorch user and you wonder, "Should I use PyTorch Lightning or maybe I don't need all of the features there," then Fabric is like your stepping stone from PyTorch to PyTorch Lightning. It's something in between. And now they are both sharing the same code base though. So the multi GPU processing or the tools we have support for like FSDP, DeepSpeed and so forth, the quantization and the precision options, this engine, they're all available in Fabric and they're available in PyTorch Lightning, but they share the same code base. So you can think of it as Fabric as a small subset of PyTorch Lightning, a decoupled subset.

00:41:17 So the boring story is essentially that, yeah, there's nothing you can do with Fabric that you can't do with PyTorch Lightning. But if you are thinking about... You have PyTorch and you are thinking about doing more GPU training and you don't want to fully commit to PyTorch Lightning, then Fabric is for example, the stepping stone. It's really just five or six lines of code that you have to apply to make all of that or enable all of that. And I also have, I think, an article where I really show a file diff on PyTorch Lightning, side by side. It's really just a few lines of code. So that is where it's really attractive, where if you have existing PyTorch code, you don't want to learn something completely new or make some bigger modification, yeah, Fabric is really focused on minimal changes and getting the best out of it.

Jon: 00:42:09 Eager to learn about Large Language Models and Generative A.I. but don’t know where to start? Check out my comprehensive two-hour training, which is available in its entirety on YouTube; yep, that means not only is it totally free, but it’s ad-free as well. It's a pure educational resource. In the training, we introduce deep learning

transformer architectures and how these enable the extraordinary capabilities of state-of-the-art LLMs. And it isn't just theory. My hands-on code demos, which feature the Hugging Face and PyTorch Lightning Python libraries, guide you through the entire lifecycle of LLM development—from training to real-world deployment. Check out my "Generative AI with Large Language Models: Hands-On Training" today on YouTube. We've got a link for you in the show notes.

00:42:51 Nice. Very cool. Well, thank you and everyone else at Lighting AI for bringing us so many amazing tools for us to use. And I think of these four projects, PyTorch Lightning, Fabric, Lit-GPT, those are 100% open-source as well. And so-

Sebastian: 00:43:06 That's correct. 100% open-source, no strings attached. Yeah.

Jon: 00:43:11 Amazing. Yeah, so thank you so much for that. And even Lightning Studios, which is something that is commercial, it's one of those commercial things where of course it has to be. I mean, you're giving access to GPU cloud compute, so it sounds like a really powerful tool. I'm sure lots of people are checking that out right now as they hear it. So a common thread across all of these projects is large language models. All four of these projects are helpful for getting large language models trained and deployed. So that obviously is a key area of focus for Lightning AI. And I know from our research that this is a key research area for you in particular, Sebastian-

Sebastian: 00:43:59 Yeah. Sorry, you were-

Jon: 00:44:00 Well, no, you can jump in right there. I was just basically going to ask if there's a particular angle on this, maybe unique challenges or opportunities that you see with large

language models or whatever you were just inhaled to say there. Whatever it is, I'm sure it's great.

Sebastian: 00:44:16 So yeah, I would say that is a good point. So there was a person on... I think it was [inaudible 00:44:23] this morning who asked, mentioned something about my articles that the person loves my articles and noticed that it's a certain tendency to cover LLMs, and this is totally true. This is why... The reason is really when I work on something or write about something, the things I'm excited about are usually what this is about. It's like I like to write about things that I find interesting and exciting and that's currently LLMs. So it's a big field and I would say LLMS are big models, big fields. We also got recently so many new techniques. It's interesting to pull it all together and say something about LLMS that encompasses it, but really almost its own research field from fine tuning to pre-training.

00:45:12 I mean, I was just reading the other day, OLMo, the paper by Allen AI, which is amazing. It's like an open model and it also comes with the training code, the lock files, high performer configurations, everything. And just looking at that, they made so many choices where you feel like studying just this one choice, like why did they use, for example, a linear decay for the learning rate instead of cosine the other people do? That's already a research project in itself and there's so much to study there. I don't know even where to begin to describe how much you can get out of these works and papers. And for me, I'm currently mostly focused on the fine tuning front. Pre-training is expensive and it's not only expensive, it takes a long time. So for me it's like this almost this instant gratification with fine tuning, you take an openly available LLM, you do fine tuning, and you usually get results in a few days rather than a few months.

00:46:16 So yeah, also what I find super exciting are techniques like LoRA, like low rank adaptation where you only essentially train smaller matrices that approximate the larger matrix, the weight update matrix. It's essentially like low rank decomposition. And there are just this weekend I read this paper, DoRA, weight decoupled LoRA essentially. So it's essentially applying weight normalization to it and they got so much better results. And that is something I was just implementing this weekend, because the authors didn't share the code yet. And yeah, I'm experimenting now with that. And to me it's just exciting to apply these techniques and see how much they can improve LLMs.

00:46:57 So I'm also a user of LLMs. I use of course Chat-GPT, like everyone. But for me the excitement is really working with LLMs. I don't know what is so particular about it, but it's super gratifying to code up some stuff, see it work, see it improve, and I get a lot of fun out of that. Yeah.

Jon: 00:47:19 Do you want to talk a little bit more about DoRA in particular? You've mentioned that now a couple of times. So I think you've given the key context on low rank adaptation, LoRA, which is a super powerful technique we use at my machine learning company, Nebula. We are using it all the time. So taking something like a Llama model, like the smallest Llama model is 7 billion parameters. You can fit this onto a single GPU, and then you can fine tune it using LoRA very inexpensively. Like you're talking, depending on the amount of data, it could be tens of dollars to be fine tuning a 7 billion parameter Llama model on a single GPU and be able to have that model perform at as good... Or we actually have some capabilities that we've been able to fine tune in say a 7 billion parameter model to be doing things that you can't get GPT-4 to do. Period.

00:48:18 And so that shows the power of this kind of idea. If you have some training data that you can be fine tuning with, maybe this is proprietary data that you have from your platform that you've collected, or you could even be using things like you could be using a GPT-4 to synthesize inputs and outputs that would be ideal and fine tune on those. And so incredibly powerful technique. We've done tons of episodes on this show in the past, kind of on the individual techniques, including LoRA itself. But so Sebastian, how is DoRA this new weight decomposed lower rank adaptation? How is that different or how's that different from LoRA?

Sebastian: 00:49:04 Yeah, so let me see if I can get this in a digestible form, correct on the fly on the podcast. So maybe to go one step back to just give a recap of LoRA, for those who don't have the other episode fresh in their mind. So if you train an LLM, you have different weight matrices in the linear layers and also the Q, K, and V matrices. And then during regular deep learning, you learn how to update these weights.

00:49:33 So if you have a weight matrix W, think of the weight update matrix as ΔW. It's basically how the weight changes. And if you keep them separate during learning, let's say you don't apply it during the forward pass, then LoRA, you can think of it as this ΔW, which has the same size as W as a lower rank decomposition into two smaller matrices. So if let's say your weight matrix is 10 by 10, then you have a hundred parameters and you can approximate that with two matrices, A times B, where A is 10 times two and the other one is two times 10. If you multiply them, you get 10 by 10 back, but you have now only 40 parameters instead of a hundred. And if you go larger, if you have 1000 by 2000, yeah, then you have, I think... If you have a rank of two, only 400 parameters for example. So you save a lot of parameters by using these smaller matrices. It's almost like an hour shaped glass

where you have a large input and output like the original dimensions. But in this small inner dimension, you save a lot of parameters and you only learn these matrices A and B.

00:50:44 Now the rank the size of the inner dimension, it's like a hyper parameter. So if you make it wider, if you make it as wide as the original matrix, then you don't save any parameters. But usually you choose something like R equals eight, like a rank of eight or something like 16, something small. And like you said, you can save a lot of time and money using that. And you get usually good results and usually it works pretty well. So you have to still though experiment with the rank size. Now DoRA is a method that sits on top of LoRA. It's actually very similar to LoRA. Now the only difference is they apply weight normalization to it. And also, so maybe the main idea is more not the weight normalization, but the decoupling of magnitude and directional component.

00:51:34 So maybe to explain this, think of a vector, like a regular 2D vector. A vector has a magnitude and a direction. So if you think about a vector in a 2D space, like an arrow pointing in a 2D space, the direction where this arrow points to, that's the directional component. And the magnitude is how long the vector is. So then you can normalize this vector to have unit length, length of one, and then this would be your directional component in a unit length form, and you can multiply it by the magnitude to get that original vector. So you essentially just decomposing a vector into a directional and the magnitude component.

00:52:17 And DoRA paper, they did some analysis where they found essentially that LoRA is very, let's say, rigid in how it determines certain updates during training compared to full fine tuning. And they had the idea, "Okay, if we decouple LoRA matrices into these directional and

magnitude components, it gives the model more flexibility to only change the direction, not let's say the magnitude or the other way around." So it's essentially adding slightly more parameters, but decoupling direction and magnitude in the lower matrices. So what they do is if you have a weight matrix, the columns, you can think of them as feature vectors essentially. And they're applying this technique, or this idea of decomposing each column in the matrix into a magnitude and a directional component. So they add a magnitude vector, because it's a matrix with, let's say 10 columns. So you have 10 magnitudes and then they learn that separately. And this is essentially all they do, except as some weight normalization happening from based on the weight normalization paper. But it is a technique that you can implement in three or four lines of code.

00:53:34 Also, the authors didn't share the code yet. I hope it will be soon, but in the meantime, if someone is interested, I can also share my re-implementation of that. And based on that, what they found is that this works much better than LoRA. And not only does it work better, it works better with a lower rank. So in LoRA, you need to have at least a rank of eight to get really okay performance. And with that method, I think they were even able to get the same or better performance with a rank of four. So you can even have the parameters basically and get even better performance than a standard LoRA, which is essentially great, because LoRA is a technique to make hyperparameter... Sorry, fine tuning, more parameter efficient. And with DoRA, you can make LoRA more parameter efficient. So it's essentially. It's almost a free add-on. So it adds a few parameters. So you have this magnitude vector now in addition. But they found that even if you reduce the total number of parameters to half of what you would use in LoRA, you still outperform LoRA.

00:54:38 So this is based on the paper. I haven't reproduced all... I mean I probably won't because yeah, I don't have, let's say, the time to reproduce all the experiments. But on early results, I mean it checks out. It sounds cool and it seems to work and trusting the results in the paper, I think this is a really great technique. I mean, it could be that it turns out maybe, for some reason, it doesn't perform so well. But all the indicators so far are exciting. So I'm actually quite excited about this technique and simple modification almost, but great results.

Jon: 00:55:13 Nice. Yeah. And so to make this maybe a little bit concrete for listeners, when we do this at Nebula, let's say we take Llama 7B. So we've got a 7 billion parameter model to start with. When we do LoRA to fine tune, you're adding in additional layers. You're adding in additional parameters that you have to tune. But it's a small portion. And so this rank that you're describing, like this rank of eight that would be common for ordinary LoRA would allow us, I can't remember the exact numbers, but it would be something like, let's say to make this math easy for me, we end up with 2 million parameters that we insert into this 7 billion parameter architecture. So a very small percentage, I now remember a specific figure with a lot of the fine-tuning that we do, it ends up being about half a percent of new, so whatever 7 billion is, whatever half a percent is of 7 billion, it's about that many parameters that we're adding in and fine-tuning just with those. You leave the original 7 billion in place, unchanged, and it's just these few million that you fine-tune and you get incredible results. You don't forget, you don't have the kind of catastrophic forgetting that you could if you were tuning all 7 billion at once.

00:56:34 And the key advantage is much, much, much cheaper and faster to train because you're training so many fewer parameters, it's less than a percent. And so what you're describing there with this innovation, DoRA, Sebastian, is

that with DoRA going to that rank of four, instead of say me using 2 million parameters, I could be using a million parameters with DoRA. So 2 million LoRA, 1 million DoRA, I'm again cutting in half the amount of compute that I need to be doing, the amount of time and money that I need to be spending training my LLM to get the same results.

Sebastian: 00:57:10 So it's essentially a way to get you really good results with fewer parameters compared to standard LoRA or if you want to keep that same number of parameters because it works for you, you get better performance, like better, let's say, modeling performance. And also what you mentioned is a good point regarding you don't modify the original parameters. So that is actually why LoRA is, I think, also so popular in practice is if you have different applications or different customers, you can still save only one base model. If your base model is Llama 7B, you never have to modify this model.

00:57:46 You can just keep it as it is. And then for each customer, you only have to change, let's say a few millions of parameters and save those instead of just updating the original weights. If you update the original weights, you now suddenly have 7 billion model sizes for each of the customers. If you have 10 customers, it's 70 billion, if you have a hundred, 700 billion. But with LoRA, you can really just save these small matrices and you can leave the original model untouched, which is I think why it's also really cool in practice, in inference.

Jon: 00:58:17 Yeah. A really great point there. That is another one of the amazing advantages of this approach where having those kind of fine-tuned models for each individual client starts to become so much cheaper to store and you can just swap those LoRA weights in. Yeah. Very, very cool concept and unsurprising that it's something that you have so much focus on. Now, before we started recording,

something that happened today at the time of recording, so this news is about a month old, but today at the time of recording a brand new starting point LLM that seems to potentially have a lot of value came out from Google today.

00:59:00 So kind of analogous to the way that Meta open-sourced their Llama architectures, open-sourced being in quotes because it's not as open as real open-source, but let's use the word literally like Meta does. So analogous to the way that Meta open-sourced Llama, Google has now open-sourced something comparable, Gemma. And Gemma is based on, I guess the naming convention is based on Gemini, their proprietary models. And Gemma is even smaller than Llama. So Llama comes, the families that were made public. There's a 7 billion, 70 billion and somewhere in between like 13.

Sebastian: 00:59:49 Yeah. I think-

Jon: 00:59:49 Thirteen was the one in between.

Sebastian: 00:59:51 Yeah. Mm-hmm.

Jon: 00:59:51 Yeah. Yeah. Yeah.

Sebastian: 00:59:51 I think there's also a 34 too, in addition to that. Yeah.

Jon: 00:59:55 But I don't know if they ever made that one public.

Sebastian: 00:59:57 That is a good point. Yeah.

Jon: 00:59:58 In the research paper they talk about that one, but for whatever reason, it didn't perform as well on safety metrics. And so at least at the time of recording, I don't think that that 34 has been published yet.

Sebastian: 01:00:09 Good point. Yeah. It was somehow a little bit awkward where you would expect it to be better than the 13 one,

but it didn't outperform it. And then there was something weird about this particular model, but yeah, right. So 7, 13 and 65 [inaudible 01:00:23] I think the common one.

Jon: 01:00:24 65. Right. Right. Right. And so with Gemma from Google, the biggest one in the family that they're releasing is 7, and the other one that they're releasing at this time at least is 2. And so I'd love for you to talk more about this since my only exposure to it is the 30 seconds before we started recording when you mentioned this to me something that they just published on their blog. For me at least with that tiny little bit of context, it's cool to have access to that 2 billion that's probably, I mean I've actually already written down from my data science stand up tomorrow morning I'm going to be bringing this up to see if we can be getting the same capabilities that we're currently using Llama 7B for. If we can use LoRA or maybe DoRA to fine-tune the 2 billion parameter Gemma and get comparable results, then that's a big win for us because that's big cost savings.

Sebastian: 01:01:16 Yeah. So that is a good point. So I think I would say 7 billion models, we have quite a lot of those. I think based on the paper, so this is something I just read two hours ago because it's brand new, so I may have not, let's say, read everything in detail, but skimming over it, it looked like it's also a grade 7 billion model, better than previous ones. But more interestingly, I would say, like you said is a 2 billion model. And the reason is, yeah, it's smaller. I do think it's almost a sweet spot where I feel like 1 billion is slightly too small. When you look at tiny Llama, it's a great model, but it's maybe a little bit too small to do cool things. And 7 billion is sometimes a bit too large. Of course, you can fine-tune it with parameter efficient LoRA methods, but then sometimes it's also just interesting to work with a full sized model like for pre-training or further pre-training or for comparing, let's say LoRA to full fine-tuning.

01:02:12 And 7 billion is almost like the size where it's slightly too large for one GPU, I mean, not an A100 maybe, but if you think about gaming graphic cards like with a lower RAM where, yeah, 7 billion is slightly too large, if you want to work with it, tinker with it. And 2 billion might be quite like the sweet spot where it fits into these cards conveniently, but then it's also pretty capable. And one indicator for that is the Phi model by Microsoft, which I think was 2.8 or something like that. I'm maybe misremembering, but around that size where I feel like, oh, this was actually quite a good model given that it was so small. And now with that 2 billion model from Gemma might be interesting also to do some comparison. And it might be the new almost default model for research because one thing is also always being able to get the most or best performing model, but the larger the model is, the longer it takes to run.

01:03:12 And if you want to iterate more quickly during research, maybe it's not necessary to always use these very big expensive models because, yeah, it's both expensive and time consuming. So maybe 2 billion is a cool new platform for experiments. And based on, so looking at it so far, it's only two or three hours, it looks like the architecture is very, very similar to Llama. So there is, I would say, not a big innovation on the architecture side, but maybe the training data was the reason why it performs so well. So I noticed, for example, so the 7 billion model, I think, just used regular multi-head attention. This 2 billion model used multi-query attention. So there was maybe also a slight difference or HyperAuto choice in terms of performance, why they used one over the other for one model and not the other. But other than that, it looks very similar to Llama.

01:04:04 I think they had also an interesting activation function called GeGLU, which is something to interesting to look into. It looks like GLU but only applied to half of the

inputs in the fully connected layers. So that's just something I noticed while I was porting it with my colleagues to Lit-GPT or still in the process. So these are little differences, but it seems like a great alternative to Llama and we will see what other people say because when we record this and until this episode is out, it's maybe a month and then maybe time will tell how cool Gemma actually really is. But I think it's exciting to get all these new LLMs to play with.

Jon: 01:04:46 I think so too. I did a quick fact check on your Microsoft Phi there, which is something that I really didn't know anything about. So I looked it up. Phi-2 is 2.7 billion parameters and Phi-1 was 1.3 billion.

Sebastian: 01:04:59 Ah.

Jon: 01:04:59 So thank you.

Sebastian: 01:05:00 Yes.

Jon: 01:05:02 And, yeah, let's dig into it a little bit more. You've mentioned this multi-query attention several times in this episode, but I don't know what that is. So how is that different from the multi-head attention that is typical in transformer architectures?

Sebastian: 01:05:18 So multi-query attention would essentially be just sharing the query for multiple keys and values. It's essentially just making it slightly more parameter efficient in the sense that you have fewer parameters in the multi-head attention module by sharing the same query matrix for different heads, like multi-head attention heads. So it's essentially, if you think about one query head or one query per attention head, this would be multi-head attention. But now you can say, I use the same query for two heads, or three or four, not three maybe, but two or four heads, and then you basically save a few parameters.

And honestly, I have never done any comparison between the two. I only know also from papers like Llama 2 used multi-query attention. I think Falcon did. Palm maybe also, but I don't know if they did share any results, what the trade-off is essentially. So I'm assuming it's probably plus minus the same performance, but you save parameters essentially. That's my assumption, but I don't know for sure if it's maybe a bigger trade-off. Yeah.

Jon: 01:06:32 That was a great little snippet. I love that. We'll probably cut that exact, this is just be a short couple minute clip for YouTube. We do like we take five kind of shorter clips from the episode and I think that little introduction to multi-query attention is perfect as a clip. Nice. All right. So we have gone super far off-piste in terms of like, I had this kind of structure of questions. Our researcher, Serge Masis, had tons of amazing topics to cover, but we've ended off just from the things you've been talking about and me being interested in them going completely off-piste almost for this entire hour of the podcast so far, which is really cool to see. So coming back a little bit to the structure that I had, which last ended with me mentioning LLMs, you have mentioned research on alternatives to reinforcement learning from human feedback, so RLHF. So RLHF is today a very common practice for often the final step of training a large language model to have outputs aligned with the kinds of outputs that humans would like to see.

01:07:42 So this could include listeners are probably familiar within ChatGPT, you can give a thumbs up or thumbs down to a particular output and that thumbs up, thumbs down can be used in this reinforcement learning from human feedback to fine-tune the model to have the kinds of outputs that humans are expecting given a particular input. It can also be useful for safety purposes, for safety alignment, and there are alternatives. So I think the only one that I have any kind of immediate familiarity with is

reinforcement learning from AI feedback, RLAIF, which Anthropic favors. And so yeah, so you've previously mentioned alternatives to RLHF. I had love for you to take the floor and fill us in on more.

Sebastian: 01:08:34 Yeah. So also a nice spontaneous segue here. So these alternatives, so first of all, RLHF is, I would say it's labeled as reinforcement learning, but I would say it's almost a bit different from a reinforcement learning, let's say, applied to robots. But the idea is essentially that you train, you have your multiple steps. So you first have your pre-trained LLM, and then you follow up with an instruction fine-tuning step where you have a data set where you have instructions like summarize this text or translate this text and then a desired response. And so you teach the model to better follow these instructions. And this is just regular supervised fine-tuning. So in pre-training, you have this next token prediction task where the language model always learns to predict the next token and the correct response. So if you, during pre-training have a input data set, the target data set is essentially the input data set shifted by one position. So you always try to predict the next token and instruction fine-tuning is essentially the same thing except it's one step.

01:09:44 So if your input is the instruction plus the desired response, the target is that same thing shifted by one position. So the model, I would say, it doesn't soak up as much knowledge as during pre-training because there are fewer training steps involved because you give it the full input, basically, as one step. But it learns like the structure or the instructions. It worked surprisingly well. I think the first project was Alpaca. There might have been other ones. But Alpaca was a popular one where people tried this. And surprisingly the LLMs were capable of following instructions pretty well. But this was usually, I think this was done because it was convenient to do it

compared to implementing RLHF, which is more work. But RLHF is what researchers at OpenAI did for Chat-GPT, but also Llama, Llama 2 at least, the chat version. So the process is then that you take the supervised fine-tuned model, so you still have to do this step, but then you train a reward model. So you collect, like you said, these thumbs down and thumbs up and thumbs down answers where you get human preferences.

01:10:58 So you have the model generate always two responses. You take that supervised model and this generates two responses, or you take one, you don't have to have it, let's say generated, but let's say you have two responses in a data set and one is better than the other. There's the chosen and the rejected response essentially. And then you train a reward model to predict which one is the preferred one with a score essentially. And then you use that reward model to train the LLM to give preferred answers. And I'm just thinking about RLAF, which is the reinforcement learning with AI feedback you mentioned. I've written about this last year, but it's already such a long time ago when I remember correctly, but please correct me if I'm wrong, it's the same process except that instead of humans providing the feedback, it's something like GPT that provides the feedback. Is that correct?

Jon: 01:11:56 That's right. That's my understanding of it as well, where it's not a, the approach in terms of the mathematics is not different. It's just that you are getting the references from an AI system as opposed to from a human.

Sebastian: 01:12:12 Yeah. And I think that's done because it's very expensive to get these human responses and thinking about it, maybe labeling responses like thumbs down and thumbs up is not the most expensive task for human. It's maybe something you can do quickly. But if you think about writing the instruction in the first place, that's quite time-consuming. And I think you have to fact check me on

that, but I think Llama 2 they used 1 million responses or something, which is a crazy large number to do the RLHF and yeah, and I think if you use then something like GPT-4, I mean, yeah, it's all automated. Then you don't need to have someone do the laborious task of writing these responses and rating them. And I think also then that would be, so one thing also related to that is a lot of people, so one challenge is how to evaluate large language models. And one is of course they are automated matrix. For example, you have a question and then you check whether the model gives the correct response.

01:13:17 And usually that works well if you have a multiple choice quiz, like you have four answer possibilities and the LLM gives you the correct answer or math where you have what is two plus plus two, and the model ideally answers four, but this only can cover so much. It can cover correctness. And if you want to evaluate a LLM based on the conversational abilities, I think it's also tricky. The gold standard is really still asking humans which answer do you prefer? And because that's quite expensive, especially if you think about academic research, also a lot of people use GPT-4 for that like having a human preference. So it's essentially asking GPT-4 which answer do you prefer? And then you compute the percentage of, if you have two LLMs, you basically ask it which LLM gives the better answer. And then you compute a win percentage. Like and let's say 80% of the time this LLM produced better responses than the other LLM given Chat GPT-4 is the judge. And I think, yeah, this is also then I think pretty closely aligned to how humans respond.

01:14:30 If some people do sometimes the comparison between human and GPT-4 judges, they're very close. And it tells me that, okay, maybe GPT-4 is actually quite good at providing the preferred response that most humans would prefer. And then something like reinforcement learning with AI feedback that I think would work well

then also because it is pretty closely aligned with humans. It's maybe a checking egg problem because GPT-4 was trained to prefer human responses. So this would be one alternative to RLHF. But what I wanted to get to was actually more like mathematically different approaches. So RLHF requires this reward model that you have to train, which then is used to further align the LLM. And some researchers were wondering, is this actually necessary? Why do we have to have this extra reward model step? I forgot the authors of the paper, but there was a paper called DPO, which stands for, I think it's direct preference optimization, where it's an approach essentially where you skip the reward model step. You directly provide the correct and incorrect response and directly train on that with cross-entropy loss, and that is one alternative.

01:15:45 And there's another one, KTO, and I forgot what it stands for, I think it was Kahneman something. And this is interesting, it's kind of similar to this DPO method in that you don't have to have a reward model, but both DPO and RLHF, what they have in common is that, yeah, you have thumbs up and thumbs down answers. So you have to have a comparison. So for each data point, you have to have always these two answers, and sometimes it's maybe not the most convenient thing to collect. So if you think about maybe even a search engine, when you have a Google search engine and you type responses, you only click on one answer, or if there's a movie, you only select one answer.

01:16:33 You don't have always a pair and then say, oh, this is better than the other, or something like that. You maybe have an absolute score. And I think KTO works like that where it only requires, let's say, a single score instead of having a pair of answers, which can be beneficial in certain applications where you can't collect direct comparison. So yeah, I would say these are the three

methods that come to mind that are right now the most promising ones. And yeah, they have different trade-offs, I guess, but all are good or interesting methods to look into, I think.

Jon: 01:17:08 Nice. Yeah. And I guess so as the kind of baseline there, RLHF, I guess we would typically call that PPO...

Sebastian: 01:17:14 Mm-hmm. Yeah. That's...

Jon: 01:17:15 ... that exists today. Yeah. That's a good point. So the PPO algorithm is essentially the algorithm that then takes the rewards and applies the updates to the model. Yeah. And I did also, I hadn't heard of KTO before, but I did a quick real-time fact check here. And you're right, the K is Kahneman, and it's interesting, the T is Tversky, Kahneman and Tversky are famous social scientists and economists who collaborated on tons of super high impact papers over several decades. Kahneman is still alive and is famous, for example, for the book Thinking Fast and Slow, and he's a Nobel Prize winning economist and so on. And so, yeah, it is interesting. I haven't had time in real time here to look up the link as to why they're calling it Kahneman-Tversky Optimization, but it must be inspired somehow by research that they've done, something like that.

Sebastian: 01:18:12 Yeah. That's a good point. It is currently top of my head. I don't have the answer to that, but yeah, or you can put the paper in the show notes.

Jon: 01:18:21 Yeah. Exactly. We've given the listeners enough context you can look it up. Cool. So, yeah, RLHF topic that you've been interested in. And, yeah, I don't know if there's anything else that you want to talk about with respect to LLMs, like big challenges, for example. You know, what are the kinds of things that we should be doing over the

coming years to make LLMs more efficient, more practical, more accessible?

Sebastian: 01:18:49 Yeah. That is a good question. It's, I think, a million dollar question literally, because if you make them more efficient, it will probably save you millions of dollars if you're a company. But if I knew the answer, I don't even know if I would give it away here. But I think there are interesting directions. There are the parameter efficient fine-tuning techniques, which are essentially patching LLMs, but it doesn't make the LLMs smaller. It only makes the fine-tuning smaller. Right? But we have seen so far in the last year that we can really go down from, you know, 70 billion models to one or 2 billion and still get good results. And I think one direction is always that these LLMs are still general models. They are still performing or tested on various different tasks from language translation to question answering to summarization and maybe something else, so they're very, very general. And maybe if we want to make them smaller, we don't need to have them that general.

01:19:50 Maybe that's one avenue where going more towards specialized large language models or something like that, sure, we can specialize them in the fine-tuning stage, but you still have that originally big model to work with. Maybe during pre-training there is something where we can use a smaller architecture. One way that was promising, I would say, is the mixture of expert approaches by Mistral, where it's essentially one way of maybe not making the model smaller, but making the inference costs smaller by having multiple parallel feet forward layers and then only using few of them during forward pass.

01:20:31 And maybe there's also some way where we don't need, or we can select certain experts and only use those for certain applications where we don't have to save the

whole model. But that would require, of course, more analysis for in terms of which experts are actually used for which tasks, and maybe having some constraints. Like maybe for math, I only need expert one and eight, and then this way we can construct these smaller models. Other avenues are also really alternatives to LLMs. I mean, I would still call them LLMs, but let's say alternatives to transformer LLMs. I think there was at least two RWKV. It's a recurrent neural network approach. And there is Hyena.

Jon: 01:21:14 Mamba.

Sebastian: 01:21:16 [inaudible 01:21:17] Yeah. Mamba, I wanted to get to this. Yeah. Good point.

Jon: 01:21:18 Sorry. I jumped in too fast.

Sebastian: 01:21:21 Yeah. But yeah, Mamba is, I think, the hot topic. So Hyena was last year. It was like a structured state space model where Mamba is I think a selective state space model. And I'm honestly not an expert in state space models. But what I understand correctly, the difference between Hyena and Mamba is that the hidden layers or the hidden units in Hyena are more independent from the input, you know, like in attention, the inputs influence. So you always have an influence of inputs to whatever comes out of the network. But if you think about a multilayer perceptron, the weights are static, whereas in a transformer on the inputs dynamically change the attention values. So you have the same weights and the matrices, but due to the DOT products, between the values and the queries, you have dynamically based on the input of different attention scores, and then it pays different attention to different parts of the input sequence. And when I understand correctly, the Hyena model didn't have that, which is the main selling point, because then you save a lot of time because you can do

all the token computations all in parallel, instead of having this sequential step also during the attention computation. But I think the Mamba had some modifications that was needed because Hyena only performed well if it was not language related. For example, if you had DNA sequences, I think there was the big paper on a million token long DNA sequences and this performed really well, but when you applied it to language, to actual human language, it didn't perform so well. So Mamba, they kind of updated that, I would say, and came up with a selective state space model where it's more, I would say, somewhere in between Hyena and a transformer, but it is still much more efficient than a transformer. The only question is what happens if you scale it up?

01:23:26 Because it is competitive with small LLMs that are transformer based on the maybe 1 billion parameter space scale. But what happens if you look at 7 billion, 70 billion, 100 billion? So that will be still an open research question, but I think that is also something to have on the radar. I don't know if it will ever replace transformers, but I'm glad that people, smart people, are looking into this because I think, yeah, that's usually... It's always interesting to see what alternatives are out there. And transformers themselves, they originated because people thought, okay, we are working with RNNs for two decades and somehow we maybe are getting stuck here. We don't really know how to make them better. We had LSTMs. We had GRUs. But what's the next thing?

01:24:19 So what they did is... So before that, it was a paper, I think in 2014, by Yoshua Bendura's group where they added this attention mechanism to RNNs. But then researchers asked, do we even need the RNN? Maybe we can have an architecture just based on attention. And then they came up with a transformer and I mean... We know that this was actually the right thing to do. And

now maybe Mamba, it looks like maybe it's not better now, but maybe with some other tweaks it could be the next thing. Maybe it's not even meant to be better than transformers. Maybe it is better on the small scale. And that would be also a nice alternative where you have an application like ChatGPT where this is an application that is a general chatbot. I need the maximum modeling performance. I use a transformer. Or here I have a specialized tool. I only care about, let's say, language translation. Maybe I should use this smaller Mamba architecture or something like that. So where there is maybe place for a different architecture, so it doesn't have to be all the same.

Jon: 01:25:21 Yeah, it's interesting how you started your answer here with, I don't know if I have much to say about future interest in research directions, but you nailed two great ones here, a mixture of experts and alternatives to the transformer. And something that just popped into my head, because you happen to put those topics side by side, is this interesting... And particularly with the way that you ended it just there by talking about there might be some situations where you want a transformer versus maybe another situations you want a Mamba. I wonder if anyone has yet done a mixture of experts where some of the experts are transformer architecture and some are striped Hyenas, some are Mamba because maybe you're blending genomics tasks with natural language tasks. And so the mixture of experts could support top performance in all these different kinds of areas.

Sebastian: 01:26:04 Yeah, I think time will tell. Maybe, I mean, when this podcast is released next month, we will already have a completely new thing. Who knows? I mean, things are happening so fast.

Jon: 01:26:17 Yeah, super fast. Although it's interesting to me, you talked about right at the beginning of the episode how

you wrote your Q&A iBook to last longer by not having code demos in it. And it's interesting to me how despite things moving very quickly, there are things... I mean, I had the same kind of notion with the mathematical foundations of machine learning content that I've been creating. It's on linear algebra, calculus, probability theory, these underlying fundamentals, which are very unlikely to change in our lifetimes, I suspect. I think that even when I die, machine learning will still involve linear algebra and calculus. I think this is very likely. And similarly, even things like neural networks reaching back to the 1950s. And so this kind of approach is still... We don't say the word deep learning as often anymore, but that is still what's happening almost invisibly now behind all the LLMs that we talk about in so many episodes, including in today's. And so it's interesting that despite things moving very quickly, there are still these foundational elements that are worth learning and that seem to have a very long lifespan.

Sebastian: 01:27:33 I totally agree. I think also if you just think of transformers, everything in the transformer can essentially be implemented with fundamental deep learning building blocks, like linear layers, even though if you disable the bias, the query [inaudible 01:27:49] and values, they can be implemented with linear layers and activation functions. Still the same you would use in deep learning like ReLU, GeLU, [inaudible 01:27:58] and so forth, and LayerNorm and so forth. LayerNorm originated from I think, convolutional networks. So it's all coming together there and it's still the same foundations. And also interestingly, well, I really appreciate that also, people are still, of course, improving PyTorch. They're always new releases, but they're always so far as I can tell, backwards compatible. So the code that we wrote five years ago, mostly, I think it should still work because it's all backward compatible. And I think also frameworks wise, we pretty much converged all to PyTorch.

	01:28:29	I mean, of course there are people who use Jacks or TensorFlow with Keras, but I think most people nowadays really use PyTorch. And I think it's great that more people use the same framework. Of course, multiple frameworks are also good because they can inspire each other. But for code readability, for me, it helps if I'm very familiar with one framework and everything is in that framework, so I have a easier time to see what's going on and so forth. So I think right now we are in a time where we all kind of use the same basis for everything. It's just stacking up the layers slightly differently. It's still, of course not trivial, but at least it helps that we have these building blocks. And then also mathematically, like you mentioned, a lot of stuff is still calculus standard back propagation, and then also the linear algebra you need for the matrix multiplications. But that's really it, I would say. So still luckily the same building blocks. I mean, I love learning new things, but the day only is 24 hours, right?
Jon:	01:29:34	Yeah. You mentioned there PyTorch contributions five years ago, and that tipped me on to, I think probably the last question that I'm really going to ask you, the last technical question I'm going to ask you before we get to audience questions, of which there were many for you. So you've done tons of open-source contributions. Our researcher Serge has a note here that he thinks you're involved in over a hundred repositories.
Sebastian:	01:30:01	Wow.
Jon:	01:30:02	And some of those like your ML Extend package are very popular. ML Extend has about 5,000 GitHub stars and over 20,000 downloads a day. So it's likely been used millions of times now, downloaded millions of times. So I guess you could tell us a tiny little bit about ML Extend maybe since I mentioned it, but in general, a kind of bigger question that I have for you is over all of these years doing this open-source contributions, are there any

particular achievements or contributions that you think are really, I don't know, you still have a really warm feeling about that? It feels like it was something really special?

Sebastian: 01:30:47 Oh yeah. That is a good question. I would say my time working with and helping out with scikit-learn was really great. I think that's also what really got me into open-source... So back then I was an R, Perl and Bash script user, but that was way my time before doing open-source contributions really. I started doing more open-source contributions when I started using PyTorch. And then specifically when I got started with deep learning and used the scikit-learn Library, and that is also related to ML Extend because ML Extend stands for machine learning extensions. And that was essentially a playground to implement things that are not yet in scikit-learn or may never be there, but they are useful because I needed them at the time, for example. So that was, for example, back then the majority vote classifier, an ensemble method, the feature selector, sequential feature selection, and both of them, they made it also into scikit-learn.

01:31:51 And that I think was... I was really happy about contributing to scikit-learn because at that time I was a user for many years, and I think it's just an awesome community with all the contributors. It has been also really popular back then where everyone was using scikit-learn, and the team was working really hard to make sure everything is rock solid for all these people who rely on scikit-learn. There's a lot of people in the industry too. And so yeah, I think this was a great time when I was working with scikit-learn. I still work with it occasionally. It's just that I work more on deep learning right now at the moment. But that is one project I think that is really nice. And it's still around, it's rock solid and there are still new features always all the time. I think that is definitely

something... A really cool open-source project that I'm really thankful for that it exists. And I think without scikit-learn, I probably wouldn't be so much into open-source and machine learning right now. So I think it's maybe what inspired me or still inspires me.

Jon: 01:32:56 It's so cool to hear you focus on scikit-learn there. Not just the project itself, but the community around it. That is something that we talked a lot about in episode number 737 with Gaël Varoquaux, who's one of the co-founders of scikit-learn project. And so I had the amazing opportunity to meet with him in person in Paris and film an episode of the Sorbonne where... I mean he just completely blew my mind and you could see in so many of his answers, it was the community aspect that came out is what's so special about the project.

01:33:30 Nice. Well, let's move on to audience questions now. So we had tons of engagement at the time of us recording, which is just five days after I posted. There's 10,000 impressions, several hundred reactions on LinkedIn and dozens of comments. Some great questions came through from listeners. So having reviewed the questions, a lot of these questions we've touched on already in this episode, which is cool. So things around where you see things going and AI research papers, and in LLM innovation. There is a question here from Daniel [inaudible 01:34:13]. He's a data scientist. He's working on machine learning solutions in particular, and he asks, as an AI educator, which you do a ton of, he says, how do you earn money and get support for that work? Do you have any tips on how someone can become more visible to other AI practitioners?

01:34:33 So those are two questions which could be treated separately, but he put them in the same bullet amongst many other questions and separate bullets. So I think he's asking those, so they're thematically related. The

idea here is how do you support... As somebody who wants to be educating people on AI and probably doing a lot of open-source development like you are, how do you support that? It seems like in your case you have a very specific answer which you can go into, but maybe you have also some general ideas on how people more generally, other than landing amazing roles like staff research roles at one of the hottest AI companies out there, what are other ways that people can earn money in support being AI educators or doing open-source projects?

Sebastian: 01:35:19 Yeah, that's an interesting question. I think I would say that's maybe also something I'm not particularly good at because I usually do what I'm excited about and I shared with people, and I'm lucky that people are also excited about these topics. So I'm always, I think, lucky that my interests overlap with other people's interests in that sense. But in general, I think for me, I have a main job, a day job. There's a lot of stuff that happens on the weekend where I get to tinker on my projects. I also tend to get up very early in the morning and sometimes go to bed too late to do certain things, but it's also like if I wouldn't do them, they would keep me up at night. So it's like where I get the answers to some interesting research questions or when I read that [inaudible 01:36:07] paper, I was like, well, I actually wanted to maybe take it more easy this weekend, but hey, I need to implement this now because it's so exciting. I want to see if it actually works, for example.

01:36:15 It's like that stuff where I guess people... Instead of me reading something, I would just code something or instead of playing video games, which I also like, I sometimes do other things, but other than that, how to do this for a living, I would say for me it's still a side project, but I have my books, so if someone wants to support me there, you would consider purchasing a copy of my book

or one of my books that would be really appreciated. And other than that, I think also there are other great ways, not for me right now, but for other people. I've seen Substack, for example, as a great alternative for educators where you can selectively share content for free, but you also have the option that people can have a paid subscription to support the writer basically in that sense.

01:37:07 Or I think other ways are on YouTube where people have corporations with people who I think sponsor content. I have not looked into this so far, but I think there are these options out there if you want to consider this route. And other than that, I would say, I'm maybe not the best person to optimize on that front, but I would say if you do what you like or what you love, then you will enjoy the ride. And if others support... I think there are always people who would like to support work that is useful to them. So if you work on something that is exciting and useful, I think there's nothing you can do wrong there.

Jon: 01:37:53 Yeah, great answer. And I think in terms of those kinds of sponsorship platforms, I guess Patreon, that kind of thing is out there for people or content creators. A completely unrelated question, but that's kind of how it works sometimes. I don't have great segues between topics when I'm working with audience questions. This is from Ravi [inaudible 01:38:20], I'm not sure I'm pronouncing his name correctly, but he's a data and analytics architect. And Ravi is curious about use cases for LLMs beyond just as the kind of typical use case we have today as an API. So he's curious about a future where LLMs could be used not just as an API, but maybe more integral, like the idea of it as a platform or even operating system.

Sebastian: 01:38:52 Honestly, this is a very interesting question. It's something I have not thought about how to use LLMs besides let's say language and images as more like a

platform that runs things. Because I think one thing about it is always to also remember it's quite expensive. So it's like when you have mission-critical stuff happening in real-time, might not be the best choice for an LLM, but there are interesting applications of the transformer architecture that is inside the LLMs where maybe this could be used in, I can see maybe integrated on a chip where it reroutes certain gates as a gating mechanism. Another thing kind of unrelated related is AlphaFold by DeepMind, where it's a model that is used to do protein structure predictions. And so essentially the challenge has always been the protein folding problem that people want to understand given a protein sequence, how this protein sequence folds into a 3D structure.

01:40:03 Because once you understand this 3D structure or have this structure, you can design, let's say certain drugs for diseases, for that type of thing, like drug discovery or just study certain diseases. And usually the traditional way is crystallography where you crystallize these experimentally, but you can't do that easily for all kinds of proteins, especially like membrane proteins. So physicists have been always developing these molecular dynamics simulations. It's a physical process to fold up a protein sequence into a 3D structure. And then DeepMind developed AlphaFold, which is a deep learning model that predicts the structure directly without the folding process. And for that, I think in the second iteration, they also used transformers. So it's essentially mapping from an input text format, it's a protein sequence, it's like a string of letters into a three-dimensional structure with 3D coordinates and so forth. And I think they also added physical constraints like impossible geometric angles and so forth.

01:41:10 And I think this still is the most advanced prediction method for protein structures. And as far as I know, it's based on a transformer architecture inside. So that is I

think also an interesting usage of this type of model that is not, let's say the traditional LLM where it's not output text, it's on output structures. And I can see also making transformers maybe or developing them for other types of applications. Which applications in particular? I'm not sure yet, but I think there are... I mean, it's essentially a great prediction engine that you can use for other things as well.

Jon: 01:41:50 Nice. Yeah, interesting question. Not one that I would've... I'm not sure what I would've said if someone had asked me. And I think that was a great answer, Sebastian, very cool. All right, well you've been super generous with your time today. I really appreciate all of it and the amazing insights that you had. Interviews from my perspective, don't get better than this. Audience, please give us some feedback on the episode, but it seems to me like this is an absolute kind of home run of an episode. You provided us with tons of packages that we can be exploring on our own time, mostly open-source and tons of models, innovations all cutting edge for us to dig our teeth into. So many branching points from this episode and explained so well. So lucky to have you on the show, Sebastian. Thank you. But before I let you go, I always ask our guests for a book recommendation. So do you have one for us?

Sebastian: 01:42:44 So yeah, that is a good one. Actually, if you want a technical book, I enjoyed reading, I think it's called Machine Learning Systems by Chip Huyen, which is a great overview of putting machine learning models into production, which is I would say my weak spot as a researcher. So I learned a lot there about the different things to put things into production.

Jon: 01:43:08 Iconic book. We had her on for episode 661 to talk about her book. So if people want to get a little taste of it before

buying, you can do that. Anyway, sorry, your other recommendation.

Sebastian: 01:43:17 And the other one, I picked it up at an airport, it was called Tomorrow, Tomorrow, Tomorrow, which is a fiction book. It's a story about a couple who develops or starts a video game startup. And for some reason it was a really nice book that I enjoyed that is not technical. So if you are looking for some little fun read, that was something I can recommend.

Jon: 01:43:43 Very cool. Great recommendations. And how should people follow you after this episode? Obviously you have a crazy amount of insights and open-source code that you can be sharing. What are the best ways to follow you?

Sebastian: 01:43:58 Yeah, I would say the classic ways, Twitter/X, or LinkedIn are two ways where I share my thoughts. It's usually, I live in the middle of Wisconsin essentially where it's pretty quiet, so this is my way to communicate with people is almost like that. But then the other one is I also have a Substack where I share about once a month some research summaries. Usually it's relatively research focused, but I try to have an article once a month because it's usually I read these articles and then... It helps me also as my personal note-taking system to just write about them and I like sharing them. So that is, I would say another way you can see what I'm up to.

Jon: 01:44:44 All right, well Sebastian, again, thank you so much for being on the show and hopefully we can catch up with you again in a couple of years and see how things have been coming along. This has been such an exceptional episode.

Sebastian: 01:44:54 Yeah, this was a lot of fun and I'm happy to be back one day. Maybe we have the successor to LLMs and transformers then, we will see, will be fun to look back on

		the topics that we covered here. And I really enjoyed chatting. It was such a nice spontaneous get together and I really liked it and I had a lot of fun and I hope the listeners too, so I hope there was at least a useful nugget in there somewhere. And thanks for having me.
Jon:	01:45:22	I'm sure there are many, no doubt. All right, thanks Sebastian.
Sebastian:	01:45:25	Thanks and I'll speak to you soon.
Jon:	01:45:32	Holy crap, what an amazing guest and what an unbelievable episode. In today's episode, Sebastian filled us in on Lightning AI's libraries and tools including PyTorch Lightning, Fabric, Lit-GPT, Lit-LLama, and Lightning Studios. He also detailed DoRA for PEFT with half the parameters of the more popular LoRA. He talked about Gemma 2 billion or 7 billion parameter LLMs, multi-query attention, RLHF alternatives, namely RL-AIF, direct preference optimization, and Kahneman-Tversky Optimization. He also filled us in on how he's excited about mixture of experts models and alternatives to transformers like Mamba as emerging opportunities for LLM development.
	01:46:10	As always, you can get all the show notes including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Sebastian's social media profiles, as well as my own at SuperDataScience.com/767. And if you'd like to engage with me in person as opposed to just through social media, I'd love to meet you in real life at the Open Data Science Conference, ODSC East, which will be held in Boston from April 23rd to 25th. I'll be doing two half-day tutorials. One will introduce deep learning with hands-on demos in Pytorch and TensorFlow, and the other will be on fine-tuning, deploying, and commercializing with open-source LLMs featuring the Hugging Face transformers.

And perfect for today's episode, the Pytorch Lightning Library. In addition to these two formal events, I'll also just be hanging around and grabbing beers and chatting with folks. It'd be so fun to see you in Boston at ODSC East.

01:47:04 All right, thanks to my colleagues at Nebula for supporting me while I create content like this Super Data Science episode for you. And thanks of course to Ivana, Mario, Natalie, Serg, Sylvia, Zara, and Kirill on the Super Data Science team for producing another most excellent episode for us today. For enabling that super team to create this free podcast for you, we are deeply grateful to our sponsors. You can support this show by checking out our sponsors links, which are in the show notes. And if you yourself are interested in sponsoring an episode, you can get the details on how by making your way to jonkrohn.com/podcast.

01:47:38 Otherwise, please share with folks that you think might like it. Review the episode in your favorite podcasting platforms. I will read those reviews on air. Subscribe if you are not a subscriber already. But most importantly, just keep on tuning in. So grateful to have you listening and I hope I can continue to make episodes you love for years and years to come. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science Podcast with you very soon.