/mlst

LSTM: The Comeback Story? [Prof. Sepp Hochreiter]

March 23, 2026



Machine Learning Street Talk

Podcast Transcript

Revision: 848a0cef

Topic headings include retention indicators: ● above average, ● average, ● below average (vs. similar YouTube videos)

Interested in licensing ReScript for your podcast? Get in touch with Tim Scarfe.

Contents

Overview 2

LLM Capabilities and Limitations Debate 3

Program Generation and Reasoning in AI 4

Human vs AI Reasoning and AlphaGo's Move 37 6

LSTM Development History and Vanishing Gradient 8

LSTM vs RNN Architecture and Complexity 12

xLSTM Architecture and FlashAttention Comparison 14

Exponential Gating and Mamba Comparison 15

Industrial Applications and Fixed Memory 19

Neuro-Symbolic AI and the Pi Project 20

AI Reasoning, Kahneman, and Core Knowledge 23

NXAI and Commercial xLSTM Deployment 26

Future of AI and xLSTM's Role 29

References 30

Overview

Sepp Hochreiter, the inventor of LSTM networks, makes a forceful case that large language models are fundamentally limited – they store and retrieve human knowledge rather than reason about it. He walks through the technical history from the original 1991 vanishing gradient problem through to his latest work on xLSTM, explaining how exponential gating and matrix memory replace the old sigmoid bottleneck to create architectures that can compete with Transformers on sequence modeling while maintaining fixed memory footprints.

The conversation covers the practical advantages of xLSTM for industrial applications like robotics simulation and discrete element modeling, where constant memory consumption matters more than raw benchmark performance. Hochreiter also discusses the neuro-symbolic Pi AI project at JKU Linz, his view that genuine reasoning requires something beyond pattern matching on training data, and why he believes the current scaling paradigm will hit a wall. The technical deep-dive into FlashAttention comparisons, recurrent vs attention-based architectures, and the evolution from sigmoid to exponential gating mechanisms gives a clear picture of where LSTM-style architectures fit in the current landscape.

LLM Capabilities and Limitations Debate •

Sepp Hochreiter 00:00:00

We need new direction. Large language models are not our way to advance AI. Language models is for me a database technology. It's not artificial intelligence. You grab all the human knowledge in text, perhaps also in code or whatever, and store it. Currently, I say reasoning is not real reasoning. It's repeating reasoning things or code things which have been already seen. Our path ends with scaling up. You put more training data to make it larger but not smarter, the systems. They are not different. They are only larger. Something is missing. Yeah. Something is missing.[1]

Tim Scarfe 00:00:42

You and Juergen, you're pioneers of of connectionism in in a way, and you're neurosymbolic guys. You've you've always been. Why is that? Sepp, it's an honor to have you on MLST. Thank you so much for joining

Sepp Hochreiter 00:00:53

us today. It's an honor for me that you have me.

Tim Scarfe 00:00:57

Oh, don't don't don't be silly. The thing that is amazing about language models and and deep learning in general is that they capture a lot of the subtle intuitions, cultural information, creativity, and and so on. So, they're really good for generating programs. And the thing is, if we want to do abstraction, we need to have programs. But the thing is, where do the programs come from? If we if we build systems that can create and acquire abstractions, we need to build systems that can write their own programs. It doesn't seem possible just to do some discrete program search because it's too difficult.

Sepp Hochreiter 00:01:30

What my view on large language models is, a large language model is for me a database technology.

Tim Scarfe 00:01:37

Yes.

[1] Long Short-Term Memory (Original LSTM Paper) — Paper

Sepp Hochreiter 00:01:37

It is it's not artificial intelligent. Okay. Give me it's artificial, but sets more or less a database technology. You grab all the human knowledge and text, perhaps also in code or whatever, and store it. And you generalize it. You combine it. You know? If there's a Tuesday, I can replace it by Wednesday because it's days of the week or or names or or numbers. It's some kind of generalization, but it's things which already exist. The question is, do we need new code? Is every code already written somewhere? You only have to pull it together or to combine it. But if you really can can should come up with a new code, with a new idea, with a new concept, Large language models can only pull out existing code as they have trained on. It's just not possible. It's like they're not trained for it to produce something new, and servers are very limited. But they're very powerful because AI needs a knowledge representation. And right now, there's a problem of hallucination. Okay. How to pull out the knowledge? Also, this inference with strawberry. It's about the knowledge is perhaps already in the system. How do I get it out? It's a database where I don't know how can I access information? We need new direction. Large language models are not our way to to advance AI on the long term. It's a good database technology. It's a good knowledge representation technology. It's important for AI, but we have to find new ways.

Tim Scarfe 00:03:15

Could I could I challenge a tiny bit? So I completely agree. Vanilla LLMs are

Program Generation and Reasoning in AI

Tim Scarfe 00:03:20

approximate retrieval engines, and and even that, then they're not quite databases because they have this interpretive property. Things like o 1, they are approximate reasoning engines, so they're doing this test time compute, they're they're searching many combinations. Because the thing is, even though this thing is a a finite state automata as a fixed amount of compute does a single forward pass, but it can generate code. And the code contains all of these fundamental basis primitives that can be composed together. So you can do this test time search, you can compose together programs, you can in a sense search Turing space indirectly, right, by searching through the space of programs. So there seems to be something there. Are you saying that doing these kind of methods like o 1 is the road to nowhere and we need something completely different? Or could we just tweak it a little a little bit?

Sepp Hochreiter 00:04:05

You can tweak it and you you will go very far because you have let's strain the program space because that's very nice. There are so many programs, so many combination of programs which give you a new program. If you if you, like, if you think it's a common worker of complexity, it's a length of the programs and complexity of the programs. Programs are simple, common worker of complexity programs already stored or combined. But if you have to find a program which needs completely new concepts, cannot be combined out of existing programs, I think it cannot do it. It can only combine things it has already seen. So large language models learned on code, but it cannot come up with completely new code concepts. Perhaps they don't exist. Then then if you say everything which in code was already invented, we only have to combine and there's some nothing new, then I'm with you. But if there's something new to invent, I don't think large language models can invent this.[2]

Tim Scarfe 00:05:13

MLST is sponsored by CentML, which is the compute platform specifically optimized for AI workloads. They support all of the latest open source language models out of the box, like Lama, for example. You can just choose the pricing points, choose the model that you want, it spins up, it's elastic auto scale, you can pay on consumption essentially where you could have a model which is always working or it can be freeze dried when you're not using it. So what are you waiting for? Go to sentml.ai and sign up now. Well, yeah, let me push gently on that. So I think this is a discussion about creativity, and also epistemic foraging. So, you creating new knowledge to explain.

Sepp Hochreiter 00:05:52

But also reasoning, program is a lot of about reasoning. But you have some logic if you do this, this, this, this. But a complex logic, said your program is working.

Tim Scarfe 00:06:02

Absolutely. But if we say reasoning is knowledge acquisition, and we need systems to come up with new abstractions, and if we agree that those abstractions can be combinatorially deduced from abstractions that are already in the system, so we we have the combinatorial closure, so they they do exist. Then creating the abstractions is more a matter of understanding when a good you know, how do I find a good abstraction using

[2] Kolmogorov Complexity — Paper

Human vs AI Reasoning and AlphaGo's Move 37

Tim Scarfe 00:06:30

an algorithm?

Sepp Hochreiter 00:06:30

But I think what we humans have, it's not only that we draw our ideas from coding, but understanding sub world having all this world knowledge. I think from coding alone, you're limited. I think we have much reasoning capabilities outside of doing only programs, but I agree. In programs, you can go very far. I think if it's a program where you need a lot of reasoning, a lot of logic to go to the next step, next step, next step, and it was not in your training database, I don't think the current large language model can do it. But now I don't believe they really understand reasoning as a concept, say, imitate reasoning, say, reproduce they have already seen. But I don't know if this is a understand it. And there are many examples. You change a little bit and then as I go wrong.

Tim Scarfe 00:07:31

Can can you explain the difference between the kind of reasoning we do, you know, so strong reasoning, and the kind of reasoning that we can do in in current AI? I think in current AI, the reasoning is not real reasoning.

Sepp Hochreiter 00:07:44

It's repeating reasoning things or code things which have been already seen in the input data and combining it and also replacing some variables. The reason we do, we have we have this reasoning concept like contradiction, induction, all these things we learned, and also for us, it was hard to learn it as in school or in study, but now we have reasoning concepts how we can do, how we structure things, and how we show that something is true or not true. All these formal systems, you have to have some formal rules. In theory, LLMs might learn some formal rules, but then I can do a reasoning in a very specific thing and produce new things because I only apply the rules. If the rules are it's a training data, I can apply they can apply the rules also to newer things. But then in this reasoning system, I can reason. But if you go to another system, I lose the reasoning capabilities.

Tim Scarfe 00:08:59

I mean, 2 quick points on that. First of all, would you consider, you know, move 37 in AlphaZero? Would you consider that reasoning? So in AlphaZero, you know, the the Google, the Go player algorithm creatively discovered this amazing move that b

Sepp Hochreiter 00:09:12

I saw now it's a move. Okay.

Tim Scarfe 00:09:14

Yes. Yes.

Sepp Hochreiter 00:09:15

It created new knowledge. It it sets things it created new knowledge, but here, there was a sub symbolic part. It's a sub symbol. It was multi colored research. Yes. Search is is 1 AI, classical AI concept. At the end of the Monte Carlo Tree Search, have the value functions and so on. Mhmm. But it discovered it. But by checking things and then evaluating it, Yes. I think it's a combination between understanding the game and computing a lot of moves into the future with Monte Carlo Tree Search,

Benjamin Crouzier 00:09:59

Tufa Labs is a new AI research lab. I'm starting in Zurich. It is funded from past ventures involving AI as well. We are hiring both chief scientists and deep learning engineer researchers, And so we are Swiss version of deep sea. And so a small group of people, very, very motivated, very hard working. And we try to do some AI research starting with LLM and O 1 style models. We want to investigate, reverse engineer, and explore the techniques ourselves.

Tim Scarfe 00:10:29

Even with that, I guess you could say it was an it's still an approximate value function. There was no formal guarantees or anything.

Sepp Hochreiter 00:10:34

Exactly.

LSTM Development History and Vanishing Gradient

Sepp Hochreiter 00:10:35

That's true. But but

Tim Scarfe 00:10:36

I I completely agree that LLMs on their own, that they are they are approximate retrieval engines. But the thing is, we can build systems. We we can have formal verifiers. We can have neurosy you know, we can have, like, LEAN and COCK, for example. We can build these systems. So with these systems, can we do reasoning?

Sepp Hochreiter 00:10:53

I I think in principle, it should be possible. I'm not sure. But I think the reasoning is limited to the domain you see in the training data. There are different formal systems, formal logics. You can learn at 1 logics. If it sees enough of these rules, I think it can it knows what are variables, what they can change, and how it can produce something. I I can I think you can train an LLM for 1 logic system to produce new logic? But it's it's it's it's it's you learn the syntax. What you don't learn is the semantics. If you want to prove something, then you have to have different steps towards this proof. Yes. And here, I think I was struggling. So I would learn to do the formal things, the syntax of a a sentence, of correct formula, and I produce another correct formula by applying the rules. It has learned it, it has seen it, it can do it. But it's not goal directed. It's a step by step, they're still not perfect, they're still sometimes, or in most cases, not as good as humans.

Tim Scarfe 00:12:03

It it's interesting that I I kind of agree that knowledge is created in service of a goal, and there's a creative component to to reasoning. And we can build systems that can dream and and generate data, and we we can bootstrap Yeah. That, and some of it can come from the users of of a system. So it feels like we can build systems that that can reason. But perhaps it wouldn't have something that that we have. Maybe we have something extra.

Sepp Hochreiter 00:12:27

Yeah. But I I would bet on this, said, why should also for us, why should you learn to reason? Why not use a reasoning system? Why not call a subprogram? Can you prove this? A theory improper. Mathematica, stuff like this. You can learn it. Perhaps it's also okay to learn it, but I don't see the necessity because also we use tools. Why should not this future AI systems use use use tools for everything? For math, for looking up knowledge and stuff like this. For me, it's stupid to push everything in 1 system because also we don't do it. We know how to use our tools. Somehow, I feel that's a better solution.

Tim Scarfe 00:13:18

What has happened in the last 2 years since since we spoke?

Sepp Hochreiter 00:13:21

A lot of things happened. For example, I founded a company, NXAI. It's a company dedicated to industrial AI. And also, xLSTM happened. This is the revival of LSTM method, which now should compete with the transformer technology.

Tim Scarfe 00:13:38

Yes. And we're gonna get on to that. And before we do that, it would be good to go on an intellectual journey of LSTM's. But just before we get there, we've not really spoken about some more, you know, broad stuff. What was it like working with Jurgen? Jurgen is

Sepp Hochreiter 00:13:54

a very special person. He's very inspiring. He has charisma. I can tell you 1 story at University of Munich, where we both were. There was a seminar, and there were 3 persons. 1 tried to get all the students into multi agent systems, the other in space cognition, and Jung in neural networks. Jung came and said, oh, I'm not prepared. I don't know what to do. Then Jung gave his introduction to his topic. Out of 50 students selected his topic. So you see, he can convince persons, and it was fun. I was sitting there. I did programming, and Jung did his art things where he made circles, and out of the circles women appeared, and he did a lot of things. And he once told me it was not clear for him whether he will go into arts or whether he goes into science. But it was always fun with him, always inspiring. Yeah. Yes.

Tim Scarfe 00:15:08

Just in case the audience don't know, of course, you've worked under Jurgen, and you are both 2 pioneers in the realm of artificial intelligence. It's insane. But what gave you the intuition all of those years ago to be working on the right things?

Sepp Hochreiter 00:15:22

Probably, mean LSDMs, which stands for long short term memory. Jung introduced me to neural networks, and particularly into recurrent neural networks, but that did not work. And in my diploma thesis, he was my supervisor. He gave me some task. It's called the trunk system where you have a sequence, and everything which you can predict, you can remove from the sequence because it's predictable anyway. So if you shorten the sequence and you can run it, that was the idea of the Chanka system. But this was a sort of a solution because recurrent neural networks are not working. And then 2 things happened. First, I built a neural network. Yeah. Only 1 weight have to be adjusted as a weight to store a piece of information which you need at the sequence end. And the network could not do this. I did all my print f's, all my my coding on the screen, numbers flow over the screen, and then saw, hey. These are super small numbers, and this was the gradients. There was no wait update. The gradients were not there. And this was since a discovery of the vanishing gradient, said if you you want to have some target, you want to know what is needed to predict the target, you would do credit assignment in its sequence end, you got no signal and finishing gradient. Now a new by a recurrent networks do not work. And the solution was LSTM, so long short term memory. It's a memory cell where I built something which make sure that the gradients, as I get propagated back, as I transferred back, as I don't scale, as I remain the same. Therefore, at the sequence beginning, there was exactly the same gradient as at the sequence end. There's no vanishing gradient anymore. This was the architecture memory cell architecture, which is the core of LSDMs. And so I discovered LSDMs. I wrote it up in my diploma thesis. And later, as Jurgen came back, he asked me, hey. You did something in your diploma thesis. Should we publish it? And then we published it. Yeah.

Tim Scarfe 00:17:38

Yeah. And it's been 1 of the most cited papers in the history of deep learning. Very, very impactful paper. I mean, reflecting back on it, what do you think the long term impact has been of LSTM?

Sepp Hochreiter 00:17:49

I think it's still used in my keynote. I gave 1 example. The example is from this year for predicting floodings or droughts, but especially floodings. LSTMs, there's a major model in the Google app. And also for predicting floodings, it's used by the US government, by the Canadian government. And here, LSTM works better than everything else, better than transformers and so on. OpenAI built a big LSTM network as an agent or DeepMind with StarCraft. AlphaStar was a big LSTM network. LSTM became the major thing in language up to 2017. Everybody used LSTM together also with attention. Attention was together with LSTM. And since the 1 paper came out, attention is all you need, meaning you only need attention, and you don't need the LSTM anymore. And this where so transformer was born, and the new technology took over language. But LSTM still were good performing in time series prediction as reinforcement learning agents and so forth. But transformer were stronger in especially in language. This changed now again, hopefully. But at this time, transformer took everything over. So we are better parallelizable. You could throw more data to this model, learn on more data. So we are faster, and LSM could not compete at this time.

Tim Scarfe 00:19:27

How did the LSTM solve the trade off between storing new data and protecting data that was already stored? That's

Sepp Hochreiter 00:19:36

a very interesting questions, and that's also the strength of the new xLSTM. This idea is gating. I have we have different gates. Some perhaps most important thing is the input gate, and it scales up or down new information which coming in. It can be scaled down to 0. It's not stored or 1. Everything is stored. And the input gate is something like an early attention mechanism because input gate is an attention. You have a time series and to put what sequence elements you want to pay attention to, the input gate would do that. And then as the forget gate forget gate is saying, is already stored memory important or should I downscale it? But more important is the input gate. The input gate really picks out specific sequence elements to be stored, and so non relevant stuff is not stored. It was 1 of the first attention mechanism, but we called it gating.[3]

[3] xLSTM: Extended Long Short-Term Memory — Paper

LSTM vs RNN Architecture and Complexity

Tim Scarfe 00:20:38

Yes. Just before we get to xLSTM, because of course you've this amazing new invention, which solves many of the problems that the original LSTM had. Can you tell me a little bit, maybe like the computational complexity of an LSTM compared to an RNN? How did it compare? LSTM is an RNN. Yep. Vanilla, without all of the gating.

Sepp Hochreiter 00:20:59

Yeah, without the gating. And the complexity is only increased by this gating mechanism, but it's still linear in time. Because perhaps it's better to compare it to to attention. Attention, if you have a a new query, a new piece of information, you have to look back to all previous items, and LSTM only interacts with the memory with all stored already stored memory. Therefore, it's always a constant for 1 query. You have a constant interaction with some memory. Attention have to go through all keys and have to do this pairwise interactions. And there are 2 disadvantages. First disadvantage is competition is very complex. It's quadratic in the context length. And the second disadvantage is you have only pairwise action. You do a dot product, then exponential of the dot product, which is sensor softmax, but you have only pairwise comparison. It could be better set more talk with more sequence elements pulled together, and a new element interacts with, I would say, an abstraction of of this different tokens. 2 disadvantages as a transformer, a computation complexity plus very simple interactions. But LSTM is like a recurrent neural network. All recurrent neural networks are linear in the sequence lengths, linear in the context lengths. LSTM is a little bit more complex because it has a gating mechanism, which makes it a little bit more complex. But it's by far not as complex as a transformer with this quadratic complexity.

Tim Scarfe 00:22:36

Could you explain to the audience why something which is quadratic? So, you know, it's something that should be worse, but it actually ran faster. But why is that?

Sepp Hochreiter 00:22:44

It ran faster because of its implementation on the GPU graphical processing units, so chips where everything is implemented on. We had something like it's called FlashAttention. This is very fast attention mechanism, and you use hardware optimization. This is 1 thing. And the other thing is you could do it in parallel. I say 1 critic is looking back to all keys, but it can look back to all keys at the same time. You can do everything in parallel. You can push up all keys or or or or the whole like, assuming you have a sequence, have you have a sentence, and also words are pushed up 1 level simultaneously. While recurrent neural networks or LSM has to go over it sequentially. So first thing build up a new memory, it's the next thing build up a new memory and attention could push up everything in parallel. And therefore, attention was at this time much faster than LSTM because of this parallelism. And the second thing was you can could optimize it for the GPU for the hardware. These 2 things, parallelism parallelizing and hardware optimization. This gave attention a big advantage. You can train on much more data at the same time, and could not compete with this technique.[4]

Tim Scarfe 00:24:12

You mentioned FlashAttention as well. So can you just quickly explain that to the audience? So does that mean in certain circumstances, you don't actually need to do the full quadratic attention?

Sepp Hochreiter 00:24:21

It's still quadratic, but super highly optimized. Right. Where you use this fast memory caches in in the thing. You you use registers of of of the GPU, very, very fast entities, memories in the register. It still has the same complexity because it's mathematically quadratic, and you cannot cheat math. But you can do it super, super fast. FlashAttention was super super fast because it is hard hardware optimized.

Tim Scarfe 00:24:54

Wonderful. So can you bring in xLSTM, which is this this new invention and how does it overcome some of these problems with the original LSTM? Yes. I start with a

Sepp Hochreiter 00:25:03

spoiler because I I talked about FlashAttention. Be with with XLS10,

[4] FlashAttention: Fast and Memory-Efficient Attention — Paper

xLSTM Architecture and FlashAttention Comparison

Sepp Hochreiter 00:25:10

faster than FlashAttention, both in training as also in inference, especially inference is important. Now I go back. xLSTM. We after seeing this this this rise of the transformer, and suddenly we thought, okay. First of all, could it could it not be LSTM? Because the ResNet backbone architecture to build very, very large models, was this the key to have this feed forward connection, this many parameters where you store all the information? Meaning, is it important to build big models? Was it important to have a specific technology looking back to compress the history. We thought LSTM should do it. And we asked the question, can we scale up the LSTM like transformers and get the performance of transformers. But we know some limitations, some drawbacks of LSTMs. We already mentioned once 1. This was a a parallelization. We now made LSTM also parallel. We used the same ideas like attention to paralyze LSTM. But there were 2 other limitations. 1 limitation was that LSTM could not revise decisions. Yeah. If you store something and then you see something different comes where say, oh, this should have been stored. You cannot revise it. I'll give you an example. It's like you want to find new clothes, for example. Then you say, and now I found this clothes with this price. And if you look further in in the Internet, you find new clothes, which is even better plus surprise. Perhaps the clothes should fit your your your shoes or whatever. And if you find something better, you should stray away throw away what you already have stored. Both how a large is similarity to your shoes and also surprise. So old LSTM could not do it. If I found a better matching thing and have to memorize surprise, I have to delete everything. As the xLSTM can do this, it's the idea is exponential gating. Yes. To revise a storage decision, we do this exponential gating. So the idea is if I find something better, I upgrade very heavily, and then I normalize it. Therefore, the old best solution is down weighted. And by this, I can find something better and can throw away my my old stuff. In theory, so for get get could do it, but in practice, you cannot learn to forget because you cannot learn at the same time to store very precisely and then forget some the 1 time step. But exponential gating, exponential input gating was the key, so to say, I have something better, forget everything what was before, and this gave us an advantage. So second thing was a matrix memory, a big memory, and the original LSTM has a memory, a scalar, 1 number. You have only 1 number what which you can store. That's not much you can store. And now the new LSTM, the x LSTM, has a whole Hopfield network. We use a classical Hopfield network, and it became popular again because there was something called Nobel Prize, and John Hopfield got the Nobel Prize for this classically Hopfield network. And now instead of this single Scala, we use

a whole Hopfield network. It's like a a classical Hopfield network plus gating. We with the input, can say what we should store in the Hopfield network, and with the forget gate, how much should be the old storage items be down weighted? It's a Hopfield network equipped with gating. So if we merge the Hopfield network idea with the LSTM idea, and this gave us an LSTM with a much stronger memory, with a much bigger memory. So for exponential gating was important, increasing some memory, and so third I already mentioned, it was to paralyze it. And these 3 ingredients we used to build this a new xLSTM. It was fantastic what results it gave. We didn't expect so good results, to be honest.

Tim Scarfe 00:29:55

And I suppose that the memory mechanism is also reminiscent of the fast wave programmers of the of the 19 nineties.

Sepp Hochreiter 00:30:00

Exactly. Yeah. You can already this is so we are also, like, hopeful networks. It's always an outer product memory. You have a memory, and you have a new you have 2 vectors. 1 we like in attention we call key, the other we call value, and you have an auto product of key and value and access to some memory. This is this idea, but what we did add on is an input gate to this new item which is added and a forget gate to the old memory. But it's a known technique, this auto product storage. It's also even older for this Ising models in the seventies said already these ideas. And also, Hopfield Networks used the same idea. But also, the fast rates used

Exponential Gating and Mamba Comparison

Sepp Hochreiter 00:30:51

this idea.

Tim Scarfe 00:30:51

Could you also just give a little bit more intuition on, you know, the gating? So you said you moved to exponential from from a sigmoid. Certainly in the 1990s, people were using like sigmoid and hyperbolic tangent even for an activation function. What was the intuition at the time to go sigmoid? Just in a bit more detail, how does the exponential version fix the problem?

Sepp Hochreiter 00:31:12

You go sigmoid, it's gating. Sigmoid is between 0 and 1. Yes. And that's the natural thing to do. 1 is as the gate is open, everything goes through. 0 is nothing goes through, the gate is closed. In between, you do scaling. So for the sigmoid was a natural thing using for gating. But it has a problem because if you encounter 1 sequence element and you say, I multiplied by open 5, let's say, is a common last element, say, oh, if this is 0.5, I should multiply this by 4. But this doesn't work because a sigmoid only goes to 1. I cannot go higher. I cannot overrule this. So sigmoid is is limited, and you have to make a decision. But late, it's limited to higher values. So exponential gating is not limited. You always can do larger values. But the problem is we never used in earlier days exponential activation function because learning would break down. But we have to have a second ingredient. 1 is exponential gating, but it's a normalization. You have an exponential thing, but then you normalize by this exponential input gates. But it's like a softmax. If you remember how a softmax works, you have this e to the power of something. You have these exponentials, and then you divide by the sum of these exponentials. It's like a rolling softmax, and therefore, went into direction of attention with LSTM, but it's recursive, but it's very similar. You have exponential input gate, but then you divide by the sum of all input gates. It's a little bit like a softmax, but it's different also. And there's another thing, it changes salone dynamics. But we we don't have a clear understanding what's happening. We only saw if we have different architectures. So softmax with this exponential function has gives advantage of learning dynamics, meaning if other systems got stuck, are stalled, do not learn anymore, there are some gradient peaks which led the transformer learn. And we now observe the same with xLSTM. To revise solids decision was our reason, but learning dynamics also were modified in a positive way. But it's not understood completely what's happening here exactly. I think there's some random directions where you if if nothing goes anymore, that you have some random weight updates, help you to progress the learning. But it's speculation. Exponential gating again, matrix memory, and to paralyze it.

Tim Scarfe 00:34:15

I'm just interested in in what triggered the the flash of inspiration. I mean, if you could go back in time and tell your younger self about this, would your younger self just say, yes, absolutely, and and would you have done

Sepp Hochreiter 00:34:25

it then? Yes. But my younger self had have to see a couple of examples because at this time, we don't have this big language models. We don't have these problems where we see exponential gating helps. This big memory helps because we did we did not have these datasets.

Tim Scarfe 00:34:48

Yes.

Sepp Hochreiter 00:34:48

I know only had to say what to do, but also what data will come. Then I would say, yes. Of course. I have such a small storage if you want to store much more. Of course, you have to do this. I would have seen it, but I also needed a glimpse into future what data will come.

Tim Scarfe 00:35:04

In your paper, you you studied how these things scale with with data and model size and so on. Can you tell me about some of the theoretical underpinnings?

Sepp Hochreiter 00:35:12

This is a standard scaling loss. It's we are not developed by us where either you do model parameters, you increase model parameters, and they follow a a certain, like, exponential law, a certain curve, and you compare it with, like, transformers or state space models, which also follow a certain law. And then you can extrapolate and say, if you build larger models, we will be also better. But this is a scaling loss which we are used not invented by us by others, which is nice because you can now predict if it makes a model larger or if I use more data, how will this model this larger models behave.

Tim Scarfe 00:36:01

You mentioned state space models, you know, things like Mambo. Could you just contrast to that?

Sepp Hochreiter 00:36:06

Mambo was most competitive method for LSTM, And then after our publication of xLSTM, Mamba 2 came out. And the nice thing is MAMA 2 is xLSTM without gate input gate. It's exactly the same because it has e e to the soft plus. E to soft soft plus is sigmoid. You can do some math. Then you'll see they have also this for gate gates, they have also an output gate. Mammer 2 is like an x LSTM, but no input gate. Input gate is left out. Therefore, it's nice to see the different method converge to the same architecture a little bit. So I'm not service mama because I don't have impugate. I think the impugate is important. So remaining architecture is very, very similar now. So I started with state space models. We started with LSTM, with Hopfel and blah blah, and now we more and more converge to very similar architectures.[5]

Tim Scarfe 00:37:11

Are you seeing any hints of industry adoption of xLSTM? Yes.

Sepp Hochreiter 00:37:15

First of all, xLSTM is now faster than FlashAttention in inference also in training. I can tell you why. With FlashAttention, you have to put also stuff along context into the GPU. What we now do, we did chunks of FlashAttention, and between the chunks, we do the recurrent stuff. And we designed the chunks of FlashAttention so that we can more be more efficient on the GPU. If you have smaller chunks, you don't have to squeeze it in and do inefficient stuff. You can exactly make it so large like the cache is. So many users of FlashAttention technology. We stole it from these guys. But to do the right size of FlashAttention makes it fast. We do FlashAttention, recurrence. Flash attention, recurrence. And now we are faster than if we do a whole FlashAttention over the whole context. And that's both in training, but also in inference. This is chunk wise flash attention, it's called, or we call it like this, and this gives us a speed. I didn't expect that we could be faster than FlashAttention in training. I thought, no. No way. But, hey, it's unbelievable. Fantastic. But said we are faster in inference, we know because here attention has a has also go autoregressive because you have to produce a new word in generation and then push everything into your system again. Mhmm. And you produce a new word, you have to push everything in the system. You can cache some of the processing. Yeah. You can do it fast. But attention is not for autoregressive mechanism. In training, you have the whole sequence. So be fast. We are now faster. Unbelievable. But in inference, I was sure that we are faster, this gives us advantage in different ways. First of all, I will mention something in language. You're aware of this strawberry or 1 thing. Oh, yes. It's

[5] Mamba: Linear-Time Sequence Modeling — Paper

doing more on the inference side. It's more thinking. Yes. And on the inference side, we would be much faster. If we like, 100 times faster in the inference, we can do do 100 times more thinking.

Tim Scarfe 00:39:39

Mhmm.

Sepp Hochreiter 00:39:39

And this is a big opportunity. It it it plays in our hands. It was so nice since it came out because we are exactly there. We are fast at in inference. That would be better. But this fast inference speed also helps us to grow in industrial applications away from language. Language is not at the core of many industries, it's not quite many business, many companies. But I now can go into robotics. Transformer were used for robotics. A DeepMind had a paper. Tesla had a paper, but they all struggled. Such a transformer is too slow.

Tim Scarfe 00:40:23

Yes.

Sepp Hochreiter 00:40:23

That's sometimes you have to wait a couple of seconds before the agent is reacting. No. We have something which is much faster. And we have a second advantage. We have a fixed memory.

Industrial Applications and Fixed Memory

Sepp Hochreiter 00:40:35

We know in advance how large our memory is. If we know or embedded on embedded device, we know how large the memory is. We will design the LSTM with this fixed memory, and no matter how long the sequence was, you use the same fixed memory. If the sequence is 100 sequence elements or 100 millions, you have the same memory. And we can fix the memory, and we are very fast. And these 2 things give us advantage to go embedded, to go into robotics. Says even 1 already tried it out for drones. They have these devices, these GPUs on drones, and they emailed us. They don't want to reveal it. They said it's it's unbelievable. They had much better results, and also drones are flying autonomous. And you have to have real time control. You you cannot wait until you think. And with this exercise, it's working. We are so it's fan-

tastic. And this guy also talked to me at NeurIPS there. They said they don't know whether I want to reveal it because it's so good for them as a company. But to going robotics, going to drones, also to self driving. In a car, you want to be energy efficient. You want to have also batteries with you. You want to be fast. You want to be concise. You want to have a small, a powerful system. And here I see big advantages with this xLSTM, perhaps even going to the cell phone. I I'm not sure. I don't know the constraints of the cell phone. Perhaps this is too far fetched, but we have a thing which is energy efficient, it's fast, and we can control some memory. We can design some memory for the device, for the embedded device,

Neuro-Symbolic AI and the Pi Project

Sepp Hochreiter 00:42:31

for example.

Tim Scarfe 00:42:31

Do you think the xLSTM sort of move us closer to something that resembles symbol manipulation?

Sepp Hochreiter 00:42:37

Symbol manipulation?

Tim Scarfe 00:42:39

Yes.

Sepp Hochreiter 00:42:41

I don't know. We have a project about neurosymbolic AI. Yes. Symbol manipulation, I would say, in 1 sense, I think xLSTM is better in building abstractions. Yes. What I'm missing for us is AI system we have out there. I never saw an AI system build proper obstructions. It's always human made. Says language is human made servers. If you look at ImageNet, a human put the object in the middle. I want to see an artificial system which comes up with a new concept, which is not human made. And xLSTM, I don't know whether it can do it, but in some memory by combining more tokens, by combining more from the past, perhaps you can build a concept because it's more efficient to store a concept, abstract concept, than stores a single items, like a like an attention would do certain items. If you can compress it to something, if you send see, you have sun, beach, cocktail,

and so on. So you say, perhaps somebody sends a beach holidays, and it's perhaps 1 abstract concept, and to store this is perhaps more efficient than storing single items. And the same should happen in industrial applications that you see concepts, you see structure, and you store the structure, not the single things. And if you have the right abstraction, you bet in generalization because if you have abstract concepts, in the future we will encounter these abstract concepts again, hopefully.

Tim Scarfe 00:44:31

Yes. The reason I asked is, of course, you've your Symbolic AI paper as well. And I'm really interested in Eurosymbolic architectures, and there are many approaches to doing that. So in Europe, we've seen many people using transformers to generate programs. Some folks are just skipping the explicit program generation and just getting transformers to perform symbolic like tasks. And transformers are incredibly limited. They can't copy. They can't count. There's lots of things they can't do. Yeah. But do you think the xLSTM could overcome some of these obvious computational sort of like limitations of of the transformer?

Sepp Hochreiter 00:45:04

Probably some of them they can overcome, but I think the solution is to combine both. I think what we have right now is not the final solution. We have to go symbolic. And there are already things out where transformer is perhaps using MATLAB to to solve a equation or whatever or is inquiring into Internet or whatever. I think we need both because there are so many symbolic techniques out there. We are for 50 years we have developed, and we should somehow integrate them, use them. I don't know if everything is learnable perhaps in principle, but now shortcut would use what's already there, combine in the right way. And in Austria, the biggest AI project, it's about €40,000,000, I'm leading it, is bilateral AI. Bilateral because bringing symbolic and subsymbolic AI together. Because we need, said in my talk, scaling is over. Now we have to go into industrialization of AI, and here we need new techniques. And perhaps not new techniques only from the subsymbolic side, from the neural network side. Perhaps we need things from the symbolic side to make things more robust. Because if you if the production process stands or is stalled, that such that should not happen. And therefore, you need perhaps a sub symbolic methods integrated or surrounding the sub symbolic methods like large language models or or or others.

Tim Scarfe 00:46:49

I completely agree. No. I I I think we need to build hybrid systems.

Sepp Hochreiter 00:46:53

Yeah. Yeah. That's a neurosymbolic approach, and that's what we are doing in Austria. It's this big thing. It's hard. I still it's hard to bring these 2 communities together. Sometimes they don't like each other. The other side, we have big success stories. The other side have other success stories. But I think sets a way for Tunx to to to advance AI, but also to make industrial AI as I said in my talk. Because for industrial AI, we need symbolic systems to make it robust to to guarantee stuff. We now have to team up with the symbolic guys to advance AI.

Tim Scarfe 00:47:29

I completely agree. So we need to have formal verification. The thing is, though, can we have our cake and eat it? Because the only problem with these hybrid neurosymbolic systems is the degree of human engineering. Can we automate the the creation with some kind of architecture search? Because we're building the the these big these big systems that have many, many components, many verifiers, and so on. How much of that can

Sepp Hochreiter 00:47:52

we automate? It's a group from where we do the neurosymbolic, the symbolic guys said, hey. We need machine running, perhaps to adjust our parameters for our symbolic stuff. So subsymbolic guys say, we can use the symbolic perhaps as a shield surrounding it. But they don't merge it. They don't integrate it, like learning rules, learning new symbolic rules with rule things. I've I know how symbolic goes, but perhaps some rules are better. You have to better integrate those things. But now every cis 2 groups are thinking in their own domains, and I'm missing cis. And if somebody is doing this, a Texas from cis community, cis from cis community, it together, but it's a clumsy, not nice, not elegant things. Elegant things also we as we did is we learn some formal systems, but perhaps the learning should go into the formal systems, and the formal system should be a subcomponent, an integrated subcomponent of, I don't know, a large language model or whatever. Right now, it's not there. These 2 groups are too separated.

Tim Scarfe 00:49:12

Yeah. So on the connectionist camp, you know, Hinton and Benjo and Lacun. You and Juergen, you're pioneers of connectionism in a way, and you're a neurosymbolic guys, you've always been. Why is that?

Sepp Hochreiter 00:49:25

Perhaps going back in history, you have to know that Germany and Austria were very strong in the symbolic case. There was this DEFKE, the and and this formal systems of many professors doing this. And in The US and so on, there were these things where Europe started and Snowbird and so on. But Jurgen was a very you know, he's still a guy who is thinking along different lines. Yeah. And there was a big group on AI, but it was formal. But he said, no. I think this neural networks. And as I went to the university, I was a student. Everything was boring. These are some theorems, 50 years old, 100 years old, all computer science, quicksort, everything, this old stuff. But since there was this neural net network stuff, Jung did, nobody knew what's coming out. You learn something. This was super, super interesting. And this was also Jung's thing. It it it was something new, not something which was very old and traditional. And also in the group we were, we read this science fiction books. Yeah. And I said, hey. I have a new science fiction book. And so many ideas came also, like, how you can transverse the universe with generationships, what's possible, what's not possible, good ideas. So it was also the time neural networks as a new technology and lot of innovation ideas and so on. This was and here, going away from this traditional symbolic thing, this was super fascinating, this new neural network stuff. You did not know what's coming out, you change something here and here. This was exciting.

Tim Scarfe 00:51:29

Yeah. And and in a way, that that's very polymathic. It's it's not it's knowledge of so many different fields at once. I mean, certainly, Jurgen was talking about things like Godel machines and recursive self improvement, artificial creativity. All of these

AI Reasoning, Kahneman, and Core Knowledge

Tim Scarfe 00:51:40

amazing ideas, some sense, they were before their time. But do you think things are starting to swing back the other way? Mean, I'm certainly seeing look at DeepMind, for example, loads of newer symbolic architectures coming out. I mean, do do

Sepp Hochreiter 00:51:51

you think that the the consciousness is changing a little bit? I think so. Perhaps it have to, because I think our past ends with scaling up, with making things larger, building, and and I don't know whether it was the right way because it's more about storing more information in these systems. You put more training data, make it larger, but not smarter. Not the systems, they are not different. They are only larger.

Tim Scarfe 00:52:20

Mhmm.

Sepp Hochreiter 00:52:20

And if this has an end, we have to be smarter. And I think the symbolic way or the neurosymbolic thing have to come because it gives us a way because I don't know where it's a subsymbolic way in which neural networks where should we go, what what should we do. We scale it up. We have this almost brain like models now, but something is missing. It's not what humans are doing. Humans learn different. They learn with a few examples, self as a abstraction capabilities. They are much more adaptive. Second plan. And something is missing. Yeah. Something is missing. And perhaps neurosymbolic gives us this much is missing. We miss something.

Tim Scarfe 00:53:09

How do we blend these ideas together? So people think of system 1 and system 2 as being completely different, and they might be very interlinked, know, like a lot of reasoning is perception guided. How how do we really integrate these ideas together?[6]

Sepp Hochreiter 00:53:20

Yeah. It's very popular after Kahneman, and and also it's a Turing Award speeches, so I used always system 1 and system 2. It's very compelling and but I'm also not sure if there's a there's a clear separation. Okay. There's perhaps a clear separation if you play a game of chess, and now you start to plan that system too. But there's this intermediate thing, something you do a gut feeling, like open a door. Yeah. You grab a thing, and you don't think about it. But sometimes you think a little bit. I think says gradually, I think. And sometimes you plan 2 steps. Should I go here or here? What's faster? Here, some guys coming. I think it's it's a very short 2 2 thoughts where you're think. And for me, it's it's not so separated. I do

[6]Thinking, Fast and Slow — Book

it immediately, and I do a long, long thinking. I think there's everything gradually many things you do do intuitively. It's and sometimes you have something where you're really thinking about. I was like, but there are so many things between when I said, if I leave here, will I go home? I can go straight ahead and see perhaps I go down there, and so I make a couple of decisions. It's a little bit planning, and it's intermediate. I don't think there's a clear difference between system 1 and system 2.

Tim Scarfe 00:54:55

Yes. I I I agree. I agree. The abstractions in these systems, should they always be human intelligible? And then what I mean by that is, you know, Elizabeth Spelke had these core knowledge prizes and, you know, things like agentness and spatial reasoning and objects and and stuff like that. And it's almost as if there's a core set of, basis functions that we've acquired or learned about how the world works. And that suggests that any reasoning system would just compose those simple priors together. Is is that all there is with reasoning, or do you think that AI systems could discover weird alien forms of reasoning that we wouldn't understand?[7]

Sepp Hochreiter 00:55:34

That's what I believe also are different concepts. The concepts we developed, words and so is what helps us. And for example, I would say neural network, perhaps you have speed, acceleration, and so on. You have these concepts. But if you now do a linear transformation of this, you have the same information but a little bit mixed up. For neural networks, that's no problem because using a linear transformation can do the inverse. It's the same information, a little bit different distributed. And perhaps sometimes it helps that the information is differently distributed. And I think for us, we develop concepts and abstraction which helps us as humans, helps us to, from 1 generation to the next generation, convey experience what we have learned, also inform others, pass food and so on. And that's the most important thing we do because if our kids have to learn what mushrooms are poisonous and whatnot, you the most information in our children is from the previous generation. They go to school and so on. And I think our language, our abstraction is designed, is tailored to from 1 generation to the next generation to to to to transmit this information because it's so most information, what you acquire as a human, as a singing human, is much less what you acquire by the culture, the I think. And, therefore, I think as our abstraction, our language, our kind of thinking is tailored to our society to and I think AI systems should come up with complete different reasoning things, but

[7] Core Knowledge Theory of Human Cognition — Paper

also different abstractions. For some other concepts might be much more useful because they live in the same world in a different way, say manipulates a world in a different way.

Tim Scarfe 00:57:52

Yeah. It's something I think about a lot because there's this constructive component to abstractions as you're talking about. So there's the language game, and we have this memetic cultural transfer, and it seems to be in service of some utility for us to understand each other. But they are still grounded in the physical physical world.

Sepp Hochreiter 00:58:12

Is it? I would challenge you. For us, it is perhaps acceleration plus something else

NXAI and Commercial xLSTM Deployment ●

Sepp Hochreiter 00:58:20

combined is the real thing.

Tim Scarfe 00:58:22

Yes.

Sepp Hochreiter 00:58:23

I don't know. It's it's only is it only convenient for us because it's helps our kind of thinking? Or might it be acceleration plus the location? Don't know.

Tim Scarfe 00:58:37

Oops. But but we we also have this weird ability as humans to think of things that are not directly coming from our sensory experience. Like, you know, abstract mathematical platonic ideals and and so on. Yes. And where do those come from? I think

Sepp Hochreiter 00:58:56

many of these things first of all, there could be only symbols. There are only placeholders for something. More interesting is also in physics because you have a concept of an atom. Probably you never saw an atom. I at least did not see an atom, but you have a concept of an atom. If you now say, what what shape has an atom, you say it's it's it's it's perhaps a ball. It's it's it's a circle. Why? Perhaps it's a triangle or whatever. You do this kind of abstractions, and you have some image in your head for for things, but often it's a placeholder. If this and this together, let's call it blah blah, and we invent a nice word for it. And you have an intuition. Perhaps you have even an image in your head versus, but sometimes it's it's it's it's abstract. It's it's has no con counterpound in reality.

Tim Scarfe 00:59:58

Yes. It's exactly. So so there's this huge difference between semantics and and the actual thing. I often think if we gave a 20 first century physics book, and we went back in time and gave it to Newton, I don't think he would understand very much of it at all.

Sepp Hochreiter 01:00:12

Yeah. I completely agree. Yeah. Yeah. Yeah. We trained in a specific way of thinking. It's perhaps different from the thinking many generations ago. Indeed. Indeed. Seth, this has been amazing. Can you[8]

Tim Scarfe 01:00:27

can you tell the audience a bit more about NXAI? NXAI,

Sepp Hochreiter 01:00:30

it's it's new company. First idea or first founding, already told you about this xLSTM. I was super excited. And I'm at the university here. And then I got to the media

Tim Scarfe 01:00:45

and say, hey. I have

[8]NXAI: Industrial AI with xLSTM — Company

Sepp Hochreiter 01:00:45

a new idea, but I don't have some money to show us that's a cool idea. And then there came this VC capital thing. Do you have a business plan? I said, no. I'm not interested in business plan. I need some money to show us a cool idea. I want to keep this cool idea in Europe. I want to keep it local. And that was a concept nobody understood until somebody local said, yes. I give you some money. Let's first fix the technology and then build on top of it perhaps verticals some companies. Since how it started, NX was for since xLSTM, so first €10,000,000 went into compute and for the first paper. And now NX for more, it's a company dedicated to industrial AI. 1 pillar is this xLSTM is a new technology we want to develop. We now showed for the with the 7 b model, we can compete with the transformer technology. It's powerful enough, but it has other advantages like this energy efficiency and speed that we can go in other directions, not only in in industry, not language. Because language, there are so many companies doing language and competing, and it's it's yeah. It's don't know whether we can make money, and it's not. That's a core business. The second pillar is AI for simulation.

Tim Scarfe 01:02:12

Yes.

Sepp Hochreiter 01:02:13

And AI for simulation, also here we have big success stories because now we can do simulations via numerical simulation struggle, which which they cannot do. There could be discrete element methods. It's like particles. You have many particles. But if it's 1000000 particles, 10,000,000 particles, 100,000,000 particles, So numerical method cannot cope with it, cannot do it anymore. It's the same with mesh points. Perhaps you have computational fluid dynamics. You have these mesh points, like if air goes over the car, you have all these points over air air airplane, and you have these mesh points. And sometimes the mesh points are so many that the numerical methods do not work anymore. Now we have things where, for example, for a car, change something. So, numerically, a simulation takes 3 weeks. The guy does something, goes home. After 3 weeks, he looks up what came out, and we can do it in 3 minutes. Wow. And and what is the idea behind it? Why is this this neural simulation so good? I always make as an example with the moon. The moon can be described by a location, by an impulse, perhaps by mass, but we don't describe each particle, each atom or each sun con. And it's a very good prediction where someone is in an hour or next day or whatever. And also in many numerical simulations, you can group particles because they are structures. And if you can group like, if you throw a snowball, you would not simulate every snowflake or whatever, but the whole snowball,

and it's it's quite good. And if, say, AI system can identify these structures, yeah, 10,000 particles stick together or do the same or in parallel or whatever, you can speed up the simulation. And this is happening. You have the things where particles are somehow synchronized or glued together. And 1 example is if you have something like a corn, you have corn in a machine. There's no physics between corn. If you have this so so a numerical simulation, you have to this quan go down to atomic level or something like this and how they interact. But if you have this coin and another

Future of AI and xLSTM's Role

Sepp Hochreiter 01:05:00

coin, and you can learn physics, how they interact, which is as a sys pushing sys, and perhaps how does it interact when if they're a little bit wet, if they're a little bit larger and so on. Then you have, like, thousands of points, which the numerical simulation needs. You have 1 thing, or a sun cone or or whatever. You learn physics of sun cones. So numeric simulations says no physics of sunkorns have to go down of atomic level or or this helps a lot to speed up the simulations. And it's super powerful because now we can do simulations where numerics struggles. And this goes so far that as there are we have from the local industry, like steel industry, you have this big oven with steel. They cannot simulate it because there are too many normally, too many particles. And often, they have to build a prototype, a larger prototype, because the simulation cannot cope with it. Now we can can jump over the prototypes. The prototype is a €100,000,000 worth because of this thing. And now we can simulate it, and we can build the real thing. And this gives industry a big, big push if this works. That would be the simulation idea. Johannes Braunschleiter is a guy he will come, hopefully. Yeah? Yes. He will tell you much more about it, But I think it's super fruitful. I think it's it's a super cool direction. But ask Johannes, he can convince you much better than I can do.

Tim Scarfe 01:06:43

Well, he's he's coming here in 30 minutes. So Yeah. Yeah. Yeah. All about it. It's been an honor and a pleasure to have you on. Thank you so much for joining us today.

Sepp Hochreiter 01:06:50

It was a pleasure to be here. It was fun. I enjoyed it. Thank you. Wonderful. Amazing.

References

[1] **Long Short-Term Memory (Original LSTM Paper)**
https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory
Paper

[2] **Kolmogorov Complexity**
https://link.springer.com/article/10.1007/BF02478259
Paper

[3] **xLSTM: Extended Long Short-Term Memory**
https://arxiv.org/abs/2405.04517
Paper

[4] **FlashAttention: Fast and Memory-Efficient Attention**
https://arxiv.org/abs/2205.14135
Paper

[5] **Mamba: Linear-Time Sequence Modeling**
https://arxiv.org/abs/2312.00752
Paper

[6] **Thinking, Fast and Slow**
https://www.amazon.com/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374533555
Book

[7] **Core Knowledge Theory of Human Cognition**
https://www.harvardlds.org/wp-content/uploads/2017/01/SpelkeKinzler07-1.pdf
Paper

[8] **NXAI: Industrial AI with xLSTM**
https://investinaustria.at/en/blog/nxai-top-researchers-in-austria-develop-soluti
Company