**[FULL TRANSCRIPT BELOW]**

**EETimes announcer:**Welcome to Brains and Machines, a deep dive into neuromorphic engineering and biologically-inspired technology. In this episode, we hear from key technical people at the Southern California-based AI hardware company BrainChip. Your hosts are Dr. Sunny Bains of University College London, and Dr. Giulia D’Angelo of the Czech Technical University in Prague.

**Giulia D’Angelo (GDA):** Welcome to Brains and Machines. I am Giulia D’Angelo…

**Sunny Bains (SB):** … and I’m Sunny Bains.

**GDA:** In today’s episode, Sunny talks to five of the engineers and computer scientists working at BrainChip, including the company’s Chief Technology Officer, Dr Tony Lewis. There are links to their work and some of the specific papers we will be discussing on our website—you can check them out at BrainsandMachines.net. After the interview, we will be talking to Ralph Etienne-Cummings from Johns Hopkins University about the issues raised.

**SB:** Thanks, Giulia.

BrainChip is based in Laguna Hills in Orange County and is probably best known for its Akida chip. Today, we’ll look at the company’s business strategy, their new flavor of neural network that’s intended to run at the edge—including both how it works and what results they’ve had—as well as providing a little insight into their strategy with their next iteration of chip.

In AI terms, BrainChip is a relatively old company. It was founded by Peter Van Der Made in 2004 to commercialize neuromorphic devices and dynamic learning. He was joined by Anil Mankar in 2015 as Senior Vice President of Engineering. Both have now stepped down and, at the end of 2023, Tony Lewis was brought in as the Chief Technology Officer.

**Tony Lewis (TL):** The company was primarily doing research—so, trying to figure out the best way to take ideas from the brain and put them into devices. And then, at some point about two or three years ago, they began to think about commercialization and how to take these ideas to market in the form of IP or intellectual property. What does that mean? That means if you wanted to build a chip, you need to have the underlying design to build that chip. And so you might go to a number of vendors for parts of that design. One part that we would provide would be the neural processing engine—that neuromorphic engine that would allow you to execute AI workloads.

And so the model of the company is that we sell, or we license, this IP to people who are building chips. Occasionally, we will fabricate our own chip to convince ourselves that our IP is good and that it performs as we think it should. But primarily, the way we make money is by licensing IP.

**SB:** For the last ten years, the focus of the company has been consistently AI at the edge. BrainChip has particularly been working on processing signals with a temporal dimension on tiny, low-power devices. Applications can include anything from large language models and speech recognition to eye tracking, gesture recognition, and noise suppression for hearing aids.

Tony Lewis explains that standard neuromorphic approaches use networks as long short-term memories, or LSTMs. These can be sped up and made more efficient using recurrence, but that’s not always straightforward. BrainChip has developed something called Temporal Event-Based Neural Networks, or TENNs, that he says solve many of the problems that arise.

**TL:**So recurrent networks are wonderful because they can represent a problem in a very compact way with very few weights. Their difficulty is that they’re really difficult to train—they’re unstable, they’re unfriendly. People would rather be in the convolutional domain. But when you go to the convolutional domain, suddenly you’re doing a lot of computation each time you want to come up with the next inference.

The breakthrough in state-space methods is that you can now train recurrent networks just like an LSTM, in a very stable way. You can convert between a convolutional domain, where things are easy to train, and you can fold it into this recurrent domain, where you can run really fast. And so you get the best of both worlds: stable training plus a very compact size.

And so what’s inside this recurrent neural network? You have this thing which is essential: it’s a state. What is a state? It’s a summary of everything that has come before that the network has been exposed to. And, in terms of a TENNs network, this could be virtually all of the time that has come before. It can listen continuously—it doesn’t have any limits as to how long it can listen—and so this state is a complete and compact summary of everything that has gone before. And the network uses that state, plus incoming information, to make its next prediction. And, as a result, we end up with networks which are far smaller; a small fraction of the size of a convolutional neural network. And, remarkably, they also behave much better.

One of the reasons why they may behave much better is that the networks have a physical basis for representing the world. So why is that important? A traditional convolutional neural network can represent anything—even things which are not physically possible. And so, as a result, they take a lot more training data to train and they take longer to train. But if we confine ourselves to things which are physical, like audio signals or cars moving around on a freeway, we can come up with a solution which trains much faster with less data.

**SB:** Dr. Yan Ru Pei, known as Rudy Pei, is a research scientist at BrainChip and one of the inventors of TENNs. I asked him to start by explaining what was the difference in the way these networks deal with time.

**Yan Ru Pei (YRP):**So Temporal Event Neural Networks, or TENNs, are basically neural networks that can leverage long-range temporal dependencies. And a wide range of tasks nowadays require a long context—one popular example being large language modeling. So, when a human reads a novel, they have—at the back of their mind—sentences or paragraphs from a long time ago, or even from the previous chapter. To establish this sort of long context, you need a model that can sort of memorize the past, or compress the history of the past in a very intelligent way.

So large language models are obviously the most popular, but even for audio processing—or any signal processing in general—you require some sort of historical context. So really the goal of the network is to leverage context to make intelligent predictions based on temporal correlations in the past.

**SB:** These applications are both sound-based, and so essentially one-dimensional. However, Rudy says the network is particularly efficient when it comes to the kind of 2D tasks required for vision. This is important because they tend to scale very badly in traditional networks.

**YRP:** Traditionally, we use something called a spatial-temporal convolution network. So in tradition, in order to perform temporal, let’s say, convolution, you have to store or buffer all the previous frames. So let’s say you want a context of 30 frames. Then you literally, at each layer, have to buffer 30 frames—which gets magnified by the spatial dimension. So obviously this is not a very edge-friendly way to approach it.

In our TENNs network, we have what I guess we can think of as a very intelligent online compression of the historical frame. So we don’t actually have to explicitly buffer previous frames; it’s more like a continuous compression of the historical context into a single spatiotemporal feature. So it’s much less memory-bound in our case.

**SB:** Key to the working of these networks is the way time is represented, using Legendre polynomials. These not only make the networks more efficient, but also more representative of physical reality.

**YRP:** So the Legendre polynomial is derived from a set of physical differential equations. Let’s say we want to approximate the past trajectory of the car—we can sort of project the trajectory into these orthogonal polynomial bases, and the coefficients of these polynomial bases basically form a very minimal compression of the previous continuous trajectory of the car. So we chose the Legendre polynomial precisely because there’s an underlying physical system basically describing it.

**SB:** We’re not going to get deep into the maths here—you can look at the technical papers for that. But if it helps, which it does for me, you can think of these polynomials as being somewhat analogous to the Fourier transforms that are used in signal and image processing.

**YRP**: Yeah, exactly. As you mentioned, spatial patterns can be compressed using Fourier transforms. So, in the temporal dimension, you can technically also use FFT or Fourier transform. The Legendre polynomial is related to Fourier transform, but it’s like a generalized version of it. So one way I like to think about it is that it’s a Fourier transform with a certain physical basis.

I guess one very popular application for this is object tracking. So one of the common problems with autonomous driving, let’s say, is—we call them ghost vehicles. So suddenly, you have a bounding box around a car, and then the car just disappears in one frame and reappears again. This is obviously not abiding by physical law. One of the important points of having this temporal continuity it’s that you want to consistently track the object in a very physical manner. The car has a certain limit of acceleration and jerk, based on some physical-inspired algorithm that is baked into TENNs, so we can make a spatio-temporal prediction that abides by continuity predictions.

So self-driving, object tracking, and many computer-vision tasks in general can benefit from this sort of temporal consistency.

**SB:** Tony Lewis points out that what’s cool about this is not just that it can be done, but that it can be done in a way that doesn’t involve fiddling with the results after they’ve been obtained.

**TL:** In the TENNs approach, this is based in the basic architecture—versus maybe other approaches people might use which would be to first detect an object and then put that into some sort of mechanism that maintains continuity in an ad hoc manner. So the TENNs approach is, in my opinion, much more elegant and complete.

**SB:** One of the key concepts in temporal processing is causality. Essentially, this has to do with how much data you have to gather and process before you can make predictions about the future, or categorize the past.

**YRP:** Causal basically means that the current prediction only depends on the current input and all the previous input. The reason why it’s important is because for a low latency, non-causal system, the current prediction has to depend, possibly, on future frames as well. So in terms of an edge system, this means that you have to wait until you gather all future frames to make a prediction in the past. So causal, I think, is a cornerstone of online inference systems. If any system has any hope of being deployed on the edge, or being efficient-online-inference compatible, it has to be causal.

I guess one example would be that most popular ASR systems, like Conformers and wave2vec2—they’re a popular ASR model, and ASR is a test of converting audio into speech—they basically have to listen to your whole speech first before they make any prediction. Even the first text has to depend on your entire speech. So that is a non-causal way to do things. Whereas for causal models, as soon as your speech token has finished, it can make the correct prediction based on whatever it has heard—so, as I’m speaking, it’s like a live transcription. And if the model is truly causal then the live transcription will have a theoretical latency of zero and so, as soon as I finish speaking, the live caption would pop up. So that is the end goal of a causal system. And I think ASR is a good example of the distinction between the two modalities, essentially.

**SB:**Of course, as with everything, there are always tradeoffs to be made.

**YRP:** I see causal and non-causal as more like a spectrum. In theory, there is no pure non-causal system or a causal system. It’s more about how far you peek into the future. Of course, if you finish listening to the human speak first, you have more future context and so you’ll be able to make a better prediction. And that is a fact that is commonly being observed in the literature.

So it’s really like a balance of how low you want the latency to be and how accurate you want the prediction. And if you plot it on a graph, it’s like a Pareto frontier. So basically, the TENNs network is sitting at the frontier of the curve—meaning that for both causal and non-causal applications, we will be expected to outperform the counterparts. This is what we hope to achieve. We’ve already achieved it in the causal domain and we’re trying to push it to the non-causal domain as well.

**SB:** As we discussed back at the beginning, recurrent processing is also critically important to the BrainChip approach.

**YRP:** The difference between a recurrent network and a typical, let’s say, convolution network or feed-forward network is that a traditional neural network, or a feed-forward network, does not contain internal states. The internal state can be considered as a memory of the network. So what happens is that if you feed the data to the system, the network will intrinsically memorize what it has seen in the past in a very abstract, compressed state. So each layer of the network contains an internal state, and that is the traditional recurrent network.

I guess one thing I should mention is that, in traditional recurrent networks, the recurrent mechanism is non-linear—meaning that, during training, the backpropagation also has to be sequential. So the forward inference is sequential, based on the recurrent update structure itself of the system. So that basically induces a backpropagation that is also sequential. That is why it is poorly supported on modern GPU hardware.

And one advantage of TENNs is that we don’t actually suffer from this sequential sort of blockade because we have a—we admit like a parallel form for our network. So, in some sense, I guess our network is a recurrent network that can also overcome the limitations of traditional nonlinear recurrent networks.

**SB:** Dr Olivier Coenen, Senior Research Scientist at BrainChip, worked with Rudy Pei to create the Temporal Event-Based Neural Network. He gets deeper into its operation and where it saves on resources.

**Olivier Coenen (OC):** So in our case, if we’re talking about TENNs, we have a recurrent form and a buffer form. In the recurrent form, what that allows is to decrease the amount of memory that you need to store all those weights. Usually, all the weights—the kernel weights, convolution weights—need to be loaded and to be available in memory. While in recurrent mode, you only need to have in memory the current state and the actual local, little weight that is necessary for the next iteration—so that size is much smaller. And so it allows you to do the same type of computation but with smaller size, essentially, network in memory.

So recurrent networks came along and they were great, and they’re especially very good for inference. But then training was really slow. And then the transformer came along and said, “Here, we have a system that basically does the job of an RNN, but that we can train at least 10 times faster,” because they could use parallelization in training. And then we’re using transformers.

But the problem now is that you need a lot of memory. And with our system, you can actually train a transformer or CNN using all the parallelization, but then you can transform it into a recurrent neural net. And the advantage is that you can basically have your cake and eat it too—meaning that you can train very efficiently and then, at the end of the day, transform it into a recurrent neural network, which is a lot more efficient in terms of memory. And so that means that we can have a much smaller footprint.

**SB:** Mamba is another relatively new model that’s good at processing temporal data. I asked Rudy Pei how it compared to TENNs.

**YRP:** So Mamba is basically another line of research that happens concurrently with the TENNs research. And obviously, we take inspiration from Mamba as well. So, essentially, it’s like a parallel line of research. But I guess the core difference between Mamba and TENNs is that Mamba uses internal state as well, but their internal state is actually expanded—meaning that they have a very large internal state bank. So this is very good for GPU inference, obviously, as proven by their recent popularity—they can do large language modeling very well, especially for hybrid models, due to their large internal state bank. But one of the disadvantages is that this is very poorly supported for edge inference, because you have to maintain all these large states and they have to continuously evolve them, which is not amenable for lightweight compute.

So the major difference for TENNs is that we do have a very small, compact internal state bank that can be very easily maintained on-chip. So, to be a little bit technical, the internal state bank can live directly very close to the processor so that the data can be fetched and continuously updated very fast. I guess that’s the core difference.

**SB:**In effect, it’s not that TENNs is necessarily better or worse than MAMBA—it’s just different.

**YRP:** We don’t know for sure, but we believe that Mamba and TENNs both have their respective domain specialty. Maybe TENNs is better for certain tasks like audio, whereas maybe Mamba is better for certain tasks like large language modeling. We’re actually not sure. So obviously, these two models will continue to evolve and they will continuously push the Pareto frontier. We think, in the future, TENNs possibly can do large language modeling very well—as we’ve recently proven.

So, I guess I cannot make too rigorous a comment on the exact domain that they’re good at, I don’t think… there’s a very fluid boundary, actually, between the two. So it’s hard to say.

**SB:** Tony Lewis says, though both are still new, the likely sweet spots for each are starting to emerge.

**TL:** As Rudy mentioned, these are two parallel lines of research. Rudy and Olivier began this TENNs research about two years ago and they were getting good results. And so we were able to create underlying hardware which could run these networks very efficiently. So there’s been a continuous evolution, or co-evolution, of both hardware and the algorithm so that we get good performance at the edge.

Whereas I believe we can characterize Mamba as something which is recently targeted at GPU hardware and which has some optimizations for GPU hardware, et cetera, which don’t really translate very well to edge use cases.

So one view is that we’re more tuned to the edge use case.

**SB:**One important area that they’re looking to commercialize is large language models on the edge. This is where the comparison with Mamba becomes interesting.

**YRP:**So we basically benchmarked TENNs against popular state-space model networks. There’s two flavors of Mamba now: Mamba-1 and Mamba-2. And we outperformed Mamba-1 and also the transformer models, like Tiny Llama or Llama 3.2, basically up to the one-billion parameter regime—so with much fewer parameters. So I think we’re very confident that this sort of approach can push the boundary of lightweight, small language modeling.

We originally thought that TENNs could not perform language modeling super well due to its small internal state size. But it turns out that, once you have a very good enough physical inductive biases for compression, you can get away with large language modeling with small states as well. So that was one of the results that kind of surprised us; that we were able to outperform Mamba.

**SB:**Among the underlying concepts behind the efficiency of the BrainChip approach is the use of events and the sparsity they provide. Tony Lewis points out how important it is for us to remember what events and spikes are, and that they’re not the same.

**TL:** What’s the difference between an event and a spike? A spike is usually defined as a single value—either the appearance of something (a one) or non-appearance (a zero). An event could also carry information additional to that zero or one; it can carry the activation level, say, of a previous neuron.

So we look at our hardware as event-based hardware. And where does it excel? Whenever we have both weight sparsity and/or activation sparsity—so where perhaps the output of a neuron, after some of this computation, it computes zero. And so this zero is never forwarded; it’s never sent to the compute engine for processing. And so we naturally save some compute power when we have sparse networks.

When we look at Rudy’s networks—when we look at any sort of TENNs network—we do see a great deal of sparsity. And we’re trying to figure out ways of increasing the sparsity so that they run very efficiently on our underlying event-based hardware. So there’s a distinction between pure spike-based and events. And then it just so happens that the TENNs stuff is sparse, which is fortuitous for us.

There is also a formulation of TENNs which can digest just pure spikes—just zeros and ones—and so there’s a lot of theoretical foundation behind it. We’ve been able to show that we can do some interesting things with event-based cameras. For instance, digesting the outputs of event-based cameras for eye tracking, for gesture recognition, et cetera, where we achieved incredibly good results; better than other researchers who have experimented.

Without getting too far into the weeds, we can digest either analog signals, we can digest words, we can digest events—all using the same framework, which is TENNs.

**SB:**Dr. Kris Carlson, Manager of Applied Research at BrainChip, explains how the sparsity that we’ve been discussing works within the system.

**Kris Carlson (KC):** We have these things called neural processors and then we have a mesh that communicates events to these neural processors. Each one of these neural processors has eight engines. And these engines, each one of them has a different set of weights to process this event with. So, again, picture a single event coming into my neural processor, and then I’ve got some set of neural processor engines. They each look at that event value, but then they each apply a different set of weights to that event. And so that’s how you’re parallelizing things at the event level. But the real benefit here is the fact that the events that they see, those are already only non-zero activations.

**SB:** This is a big advantage; there’s less information to process. But the way the parallel computing is done has to take account of that.

**KC:**The reason this is the case is because, a lot of times—when you’re trying to parallelize things—what you want to do is you want to group everything in a very organized way and then you want to just chunk through it as efficiently as possible. However, when you’re trying to take advantage of sparsity, you have a little bit of an overhead for parallelism in some ways, right? Because the idea here is I want to allow myself to compute basically the minimal amount possible. And that necessitates that you look through, or basically communicate, only when you need to. For most of the designs that are massively parallel, there’s no way to be able to say, “Oh, just do this a little bit.” Right? Because that means you have to have some sort of logic there, to choose that, and there’s an overhead for that. So, of course, you don’t want to be able to do that. So you want to order your data in a very specific way, so that you could just crunch through it all in one go. Whereas with sparsity, you want to be able to say, “Oh, actually, this whole area? I don’t even have to look at it. I’m just going to skip it,” or something like this. So that’s why, at a design level, they can be at odds.

**SB:** I wondered whether this suggested that it wasn’t the parallelism that was the problem, but rather the fact that the processing was fixed and couldn’t adapt to the changing input.

**KC:** Exactly. That’s what I mean. So it’s perhaps the data ordering, or the way that you’re crunching through things. That’s why I said the data pipeline is perhaps too fixed, so you can’t choose subsets of it. So it’s not exactly parallelism it’s more, you’re right, these things that don’t allow you to choose things. You just have to do them all at once. So you’re right, I think that’s true.

**SB:** Olivier Coenen expands on this.

**OC:** In many systems currently, you assume that you have a certain size of data and then you try to process it as efficiently—as quickly—as possible. And basically the only thing one can do is chunk it in parallel. While, on the other extreme, you assume that you don’t always have the same size of data. And then you are trying to minimize the number of operations and so, therefore, only processing the ones that are active—and so, therefore, using sparse activation to determine what you’re going to be processing.

And so it’s true that, even in the sparse system, you can use some level of parallelization in order to process the ones that are active. But I think the main point, or idea at the first, was that as these systems were developing, parallelism was the only thing that could help and make them advance and process faster.

Whereas we have two things that we can do, right? In our case, with an Akida system, we have, at some point, a level of parallelisation we can use—it’s always there. But it will also have sparsity or, in other words, only focus on non-zero activation in order to be more efficient.

**SB:** Tony Lewis says he’s excited about their recent results.

**TL:**I’ve been here for a year and so I’ve been observing Rudy’s progress and the progress of others. And what’s surprising is that we pick disparate fields and we get excellent results in a very short period of time. We aren’t experts in eye tracking, but we competed with an open competition and, in a very short period of time, we beat everyone in four out of five categories. So that was surprising. And we don’t know anything about eye tracking.

We don’t know much about audio denoising. We put together a network, a pretty innovative network, which is built upon the TENNs building blocks. And we got incredibly good intelligibility scores, but at a small fraction of the size and compute as existing state-of-the-art networks. We then looked at ASR and we put together something that, within a very short period of time, we were getting really competitive automatic-speech-recognition results with excellent word-error rate. And we had a perfect score on our gesture recognition.

And so what’s interesting is we have this tool which allows us to get incredibly competitive results without being domain experts. And so, for me, that’s surprising and shocking at the same time. But we seem to do it again and again, on a very regular basis.

**SB:** Kris Carlson says their experience with eye tracking has been especially illuminating.

**KC:** So the idea is that you have an event-based camera and you’re trying to track someone’s eye. What we found with our TENNs model is that it did spatiotemporal convolutions—basically a 3D convolution over space and time—and it had a bunch of these spatiotemporal blocks. And, remember, this thing is event-based, so it’s sparsity-aware. We entered into this… I think it was a CVPR 2024 competition. And I think we got third place, from an accuracy perspective. I think the threshold was that if you allow for 10 pixels of error, then I think we got third place. But what we found was that if you allowed for five pixels, or four pixels, or three, or two, then we did better than everyone. So this thing was really accurate.

And then the really interesting thing about this model was that when we added activity regularization, which basically helps you induce activation sparsity, we found that we basically lose very little accuracy—so the ability to track, accurately, one’s eyes—even when we get up to sparsity that’s around 90%. And so what’s really nice is that if we had no sparsity in the model then you would do, I think, 55 million max per inference—or something like this. But if you have sparsity then you’re doing 5 million-ish—5-6 million max—per inference.

**SB:** There are also many other markets where Carlson sees potential.

**KC:** So a lot of the things from an application standpoint that I’m interested in and excited about are basically streaming data at the edge. A few examples of this are something like audio denoising—where you have noisy audio coming in and you want to have clean audio coming out of your system—or something like keyword spotting or wake word, and you’re listening to streaming data very efficiently at low power. Some of these other things would be vision applications that, again, are streaming. So, picture some sort of security camera or something like this where now you’re not processing individual frames. Or another quick one, just off the top of my head, is wearables for health monitoring—things like this. So maybe you’re monitoring heart rate, or any type of biological signal, but you can picture that as a system of 1D time series. And you might want to be able to find some correlations or do some regression or whatever. You could do this really efficiently with our Temporal Event Neural Networks.

**SB:** I asked Tony Lewis whether he agreed that biomedical applications would likely be important.

**TL:** If you have anything that’s wearable, anything that’s battery powered, then power or energy is at a premium. And so when we see a reduction by one-tenth in the power usage for equivalent or better performance, it suddenly makes this new flavor of neuromorphic engineering incredibly attractive to the biomedical field.

And I can’t guarantee it but, based upon our previous experience, we think that we would do extremely well in this field once we start developing algorithms—once we’ve identified partners. Because as you know, Sunny, if we’re going to be good at this field, we require someone who has the data and the vision for how this would be used. We’re merely IP providers and purveyors of incredible algorithms. We need those partners to affect a change on society.

**SB:**Although they are mainly providers of IP they do also, as the company name suggests, have a chip—called Akida. I asked Dr. Jon Tapson, Chief Development Officer and Vice President of Engineering, what their plans were for future iterations.

**Jon Tapson (JT):**At BrainChip, we are currently finishing off our Akida 2.0 architecture. And this is an expansion on all of BrainChip’s prior work. It’s essentially a neuromorphic chip, it’s event-based, and it makes use of the intrinsic sparsity in signals to achieve high levels of efficiency.

But we have the same problem that any company has that does actual… there’s an enormous difference between research silicon, if I can put it that way, and commercial silicon. And the problem with commercial silicon is you’ve got to fit into the ecosystem of what everybody else is doing. The history of startup companies is littered with companies that had fantastic technology, but couldn’t progress across the interface to regular technology. The real problem is if you build a chip, the chip by itself is not a solution to anything. It’s got to go on a board, it’s got to have a software ecosystem around it. And the easier that is for customers and engineers to access, the better everything’s going to go.

So I think that a big part of Akida 2.0 is that it’s got some fun things. It can operate at multiple different scales of resolution—so it can operate at eight bits, it can operate at four bits, and it can operate at a single-bit resolution. And that’s actually a very interesting thing because, if you look at the trend in edge and neuromorphic processes, spikes are intrinsically one bit. And then a lot of people realized that to actually move a spike around a circuit takes a lot of signals. So you think of a spike as a one-bit signal but by the time you put an address and a bunch of timing information, there’s 19 or 20 bits of information actually being transmitted. So if you’re going to take 20 bits to send a single spike, you might as well add four more bits and send something which has amplitude rather than just being a binary (yes/no) spike presence. And then you might as well send eight bits, because that’s only another four bits kind of thing.

And I think in the field there was a real reluctance to move away from the binary spike idea, because that’s actually the core dogma of neuroscience. And, just as an aside, I thought a lot about something that was said by Rodney Brooks, the great roboticist—I think it was his advice to young academics. He said, “Choose the central dogma of your field and ask yourself: What if that dogma is not true?” And you think, okay, what’s the central dogma of neuroscience? And one of the central dogmas is that a spike is a spike—that the shape of the spike carries no information, and the amplitude of the spike carries no information; the only thing that carries information is that it happened at a certain time. And so you say, “Okay, what if that’s not true?” And there’s been a huge reluctance to move away from that dogma. Neuromorphic engineering as a field was tremendously committed to the idea that spikes were spikes. And then you find out that actually, you know what, I can send a value and I have a much richer range of algorithms and it doesn’t cost me much at all in the circuitry. So you get this kind of reluctant move from one to four bits, and then you say, “What the hell, I might as well go to eight bits.”

And, just foreshadowing, our next architecture is going to be 16-bit without a question because it just opens up a tremendous number of algorithms. And then the funny thing is you actually see the big guys—NVIDIA, for example—moving the other way. They had a massive commitment to 32-bit floating point, 16-bit floating point and now suddenly they’re offering eight- and four- bit operation as well. So they’re converging in the opposite direction. We’re all converging to four- and eight-bit values. So I find that very interesting. And that’s one of the nice things about our new architecture.

And then just one of the big optimizations that you make when you make a neuromorphic chip is the extent to which it’s just a state machine—just a bunch of electric connections that are hardwired (I think hardwired is the key phrase there) to do a task. And then the extent to which it’s actually programmable. And at the neuron level, you have a little bit of programmability—you can maybe change output functions or change flow control or something, make simple sort of if–then–else type decisions at the level of a neuron, at the level of data passing through a neuron.

And again, I think there’s a well-established trend where, as a neuroscientist, you think “Well, this stuff is hardwired. The brain doesn’t run microcode. It learns, but it learns by changing its circuit. And therefore we electronically should be learning by changing the circuit.” And then you realize that we’re neuro-inspired, but not neuro one-to-one copying. And it’s not a bad thing to have some programmability to give you just the versatility to run a bunch of different workloads.

**SB:** I wanted to end with a vision of the future of neuromorphic engineering and of BrainChip. Tony Lewis gives his perspective.

**TL:** So right now, what is the overall point of neuromorphic engineering? And what is the right level of abstraction to take ideas from the brain? I think that’s something that needs to be teased apart.

I think Carver Mead’s original idea was that we can look at the underlying physics of silicon and we can see some parallels with the human brain. So this is a tremendous insight. What we have is a digital framework which can be fabricated with standard technology and which doesn’t require anything exotic. And it may require a different level of abstraction when we borrow ideas from the brain. So when I look at a neural network like TENNs, for example, I think that it doesn’t look anything like a conventional neural network that might be inspired by Hubel and Wiesel. It’s something different. What it does mimic in the brain is this idea of an internal state.

When you look at a conventional neural network, it’s usually a feed-forward process: you get an image, you chunk through and you get your result. But we know for a fact that the human brain doesn’t process images that way. Along the way there’s many states which are maintained, which are influenced by the previous history of what you saw. And, in a similar way, this TENNs framework captures that internal state—in a very nice way, but also in a way that’s amenable to using engineering tools and things that we understand.

So I think that, at the level of abstraction, it captures an essential property of the human brain—of animal brains—which feed-forward networks don’t. If we take a feed-forward network and we try to map it into neuromorphic hardware, we’re taking away some of the biological inspiration, as opposed to adding it. We’re taking away that idea that we have internal state, where we understand what has gone before and we use that to interpret new data that’s coming in. And so by just simply mapping a ResNet50 to neuromorphic hardware, we’re going back in time.

This TENNs stuff allows us to go to the next step, which is to maintain internal state—much like the human brain—maintain some aspect of the human brain which is at a different level of abstraction than the biophysics, which was the subject that I think Carver Mead was recently interested in. So let’s call it a new flavor of neuromorphic engineering, is how I would look at it.

And so what’s the future? It’s surprising that this technique works so well, that we can apply engineering tools to it to control the learning and training so incredibly well, and that we beat out conventional convolutional neural networks, especially at the edge. So right now, I think BrainChip is in an enviable position, being able to run state-space models at the edge. There’s maybe one other competitor—Applied Brain Research with Chris Eliasmith, who’s done foundational work in this area as well—but right now, I think we’re at the forefront of commercialization.

So we have a lead. We have to keep the lead, hopefully. But we think that the world is going to move in our direction—it’s going to be very fast, where people are going to start implementing state-space models on the edge as well. And so we just hope that we can maintain a lead. And I think we’re very proud that we’re pioneering this area. Yeah, I think that’s really cool.

**GDA:**Thanks, Sunny. I could clearly feel the cohesion and maturity of this company, and I really liked all of the details and the multiple applications they already have. For more about BrainChip’s work, please go to BrainsandMachines.net.

And now, we welcome back our regular commentator, Professor Ralph Etienne-Cummings from Johns Hopkins University.

**Ralph Etienne-Cummings (REC):**Hello guys. How are you doing? Happy to be back.

**SB:** Good to see you, Ralph.

**GDA:** So I really would like to start with something that Tony Lewis said. He said that with a recurrent neural network, we could do a smaller size than a classical convolutional neural network. And what I understand is that, usually, we use convolutional neural networks for spatial data and then we use recurrent neural networks much more for sequential data—for example, language and temporal understanding. So I was wondering if we can explain how you have this reduction in size with recurrent neural networks.

**REC:** I guess one way to think about it is that, if you have some kind of a recurrent system, you can actually—what’s called opening the loop and making a feed-forward version. But when you do so, you typically have to have multiple layers to represent just one loop back. Because the time constants are different, right? So you need to basically store the partial information that would’ve happened at different time constants at different points, and then, when all those things get summed together at the end, you get “Aha! That is what that one single loop of the recurrence would’ve done.” So, in that sense, the network ends up being a much tighter, much smaller network in the recurrent case than in the feed-forward case.

Why people like the feed-forward case, on the other hand, is that the data tends to flow a lot faster through it. You just get from input to output in a much quicker way. Whereas, in the recurrent case, there is settling time—there’s time that it takes for the network to physically settle onto the final value before the results are ready. So, temporally, there might be a bit of a difference in the execution.

**SB:** There’s a big question in my mind as to what flavors of recurrent networks there are. We were talking to Katie Schuman about her hairball networks, which are a kind of recurrent network, but these BrainChip networks are not recurrent in the same way—in that they have to be much more disciplined in the way that they’re recurrent, right? There will be a very clear algorithm as to how that recurrence takes place. And I still don’t fully understand the difference between the different kinds. But what I would say is it seems obvious that whatever way you’re doing recurrence, you are able to cut down on the resources that you’re using by essentially using the same resource again and again.

Another thing that’s confusing is time multiplexing. To what extent is the recurrence that we’re talking about a kind of time-multiplexed recurrent? And to what extent is it just, we’ve got all of these things wired up on the chip?

**REC:**I would argue the following. I would say that there’s two different types of time multiplexing, if you will. You could create an entire feed-forward network with time multiplexing; you’re just basically reusing that same neuron to compute partial products along the way. It doesn’t imply anything about feedback, necessarily.

However, in the same sense, I could have mixed an output now with something that happened in the past, and then I’ve created a loop that then allowed me to essentially create a recurrent net without necessarily physically wiring—but you’re basically temporarily wiring.

In this case, my understanding was that there was some form of physical interaction—physical loopback. However, because it’s digital, it still has to have this temporal slicing as well, right? Just by its nature, it’s not an analog feedback that we can think about.

**SB:** Exactly. And so if anyone’s listening out there, I would love to see a review paper about the different kinds of recurrence that currently exist in different neuromorphic networks. If you know of one, by all means send it to me.

**REC:**One quick other thought: I liked the acausal versus causal learning and projection, if you will, and response. I thought that was pretty cool. And the examples that they gave were nice. Think about the speech-to-text: you can generate a transcript online, or you can wait till I finish talking and then generate the transcript and get the best, cleanest transcript possible. So if it’s not necessary for it to be online then I would wait, right? But if you’re trying to create something where you know you’re making it a presentation that becomes accessible, and folks need the closed caption immediately, then that’s an ideal example where you are going to make it as causal as possible and only depend on recent past information. And that is something that their system can do, right? Because they’re taking advantage of local edge computation to do that speech-to-text conversion in a causal way.

But I do think that the other part about causality and causal learning is that you can also get creativity to come out of it. Because you can guess, if you will, on what the next step should be based on only what has happened in the past. I was just talking to somebody about this recently, about the fact that I heard this ad that says, “Hallucination-free AI!” or something like that, right? And I’m thinking, I want that hallucination! Because I want it to be creative. I want it to try to give me something that I haven’t seen before, based on what it’s seen in the past.

**SB:** I hallucinate quite enough already without needing my AI to do it for me!

**REC:** Oh, come on now, you can come up with really cool ideas that you haven’t even thought about!

So the other one was, I like the John Tapson piece at the end because I thought it was a really clear description of how event-based processing can be extended from being single-spike, binary values to spikes with four bits, with eight bits, with ten bits, whatever the case may be, right? The one thing that I will say is that, typically—as these cases tend to be—there are obviously other examples of that payload-carrying event, right? Spike-based processing, event-based processing is not limited to a single bit by any means.

**GDA:** Yeah. I also love the dogma that John was referring to. And I think it’s up to us, now, to play around with the inter-spike interval and try to understand the spikes in a different way.

But there is one question I would like to brainstorm with you. So, it seems that this Legendre polynomial has been widely used for RNNs, for language models, because you have the history and then because you have the memory of the conversation to get the context, right? What if you have an abrupt change, continual change of the context?

**SB:** My understanding is that what’s interesting about the Legendre polynomials is not that they provide context in conventional language models, but rather that they encode the way real physical objects behave, right? And he gives the example about things disappearing and reappearing—and of course, physical objects don’t do that. And so the whole point is that the Legendre polynomials encode a kind of smoothness that physical objects have that language objects would not necessarily have.

**REC:**While it is continuous in the domain (from minus one to plus one), the question, I suppose, is how does one stretch that domain to match, for example, a time duration or something to that effect? They are right in saying that the Legendre polynomials tend to represent real matter and how matter behaves—even to the electronic level, right? The shells of electrons and so on are usually solved in terms of the Legendre polynomials and so on. So that’s a cool thing. I always appreciate that, when you see physics coming in and that becomes part of the computation. I look at them as basically wavelet transforms—you’re going to combine them in some ways to create the actual function. Why they are better, necessarily, than discrete cosine transform or some other wavelet—Dirichlet or whatever—that I don’t know that I understand.

**SB:** I know that they’ve looked at lots of different kinds of transforms, lots of different polynomials in this particular class, and different ones are useful for different things. So I think they use something different for other kinds of problems. But this whole class has some nice properties.

**GDA:**It seems that they got crazy insane good results on all of the applications that they talked about, so it would be super interesting for me to look at it. Also, they worked on gesture recognition, which is super interesting for me. But the EBM DVS dataset has the camera stationary, so I would challenge them to use a dataset that actually moves the camera, and then do gesture recognition whilst the camera is also moving. Because I’m doing the same and it would be super cool to see how they do it.

**REC:**On the same front, I’d like to understand a little bit better what exactly was implemented. Was this implemented in purely an algorithmic form, or was it implemented on their hardware? And were they getting the benefits of the hardware in any way, or was this just purely algorithmic?  Listening to Tony, I thought he was saying that it was on the algorithmic side, but maybe I misunderstood.

**SB:** And I never was clear about that myself. But altogether it seems like a very interesting project, and I’ll make sure that you can see all of the papers that I got a chance to read in the show notes.

**GDA:** So, guys, I think this is a great place to stop. Thank you Sunny, again, for a great interview, and Ralph for your insightful comments as usual.

In the next episode, Sunny will be talking to Jennifer Hasler from Georgia Institute of Technology in Atlanta about programmable analog arrays. We hope you will join us then.