EPISODE 90

[INTRODUCTION]

[0:00:10.4] SC: Hello and welcome to another episode of TWiML Talk, the podcast where I interview interesting people doing interesting things in machine learning and artificial intelligence. I'm your host, Sam Charrington.

This week on the podcast, we're featuring a series of conversations from the NIPS Conference in Long Beach, California. This was my first time at NIPS and I had a great time there. I attended a bunch of talks and of course learned a ton. I organized an impromptu roundtable on building AI products and I met a bunch of wonderful people including some former TWiML Talk guests. I'll be sharing a bit more about my experiences at NIPS via my newsletter which you should take a second right now to subscribe to at twimlai.com/newsletter.

This week through the end of the year, we're running a special listener appreciation contest to celebrate hitting one million listens on the podcast and to thank you all for being so awesome. Tweet us using the #twiml1mil. Everyone who enters is a winner and we're giving away a bunch of cool TWiML swag and other mystery prizes. If you're not on Twitter or want more ways to enter, visit twimlai.com/twiml1mil for the full rundown.

Before we dive in, I'd like to thank our friends over at Intel Nervana for their sponsorship of this podcast and our NIPS Series. While Intel was very active at NIPS with a bunch of workshops, demonstrations and poster sessions, their big news this time was the first public viewing of the Intel Nervana Neural Network Processor, or NNP. The goal of the NNP architecture is to provide the flexibility needed to support deep learning primitives while making the core hardware components as efficient as possible, giving neural network designers powerful tools for solving larger and more difficult problems while minimizing data movement and maximizing data reuse. To learn more about Intel's AI products group and the Intel Nervana NNP, visit intelnervana.com.

[INTERVIEW]

[0:02:30.9] SC: All right everyone, I am here in Long Beach, California at the NIPS Conference and I am seated with Joan Bruna, assistant professor at the Courant Institute at NYU and also at the Center of Data Science at NYU, and Michael Bronstein, who is currently on sabbatical at Harvard. Joan and Michael, welcome to this week in machine learning in AI.

[0:02:52.7] JB: Thank you.

[0:02:54.3] SC: Why don't we get started by having each of you introduce yourselves and tell us a little bit about how you got involved in machine learning and artificial intelligence.

[0:03:03.8] JB: Yes. I did my masters and my PhD in France, and actually at the beginning my training was more on the signal processing and applied math and I was not so much into machine learning up until I started to mutate through my PhD and entered machine learning through the ideas and tools from signal processing. That was five years ago, and ever since I've been getting more and more interested in ways and problems in which we can maybe leverage some of the more techniques and ways of mathematical tools from applied math into some of the current modern problems in deep learning. Yeah, that's kind of how I arrived here.

[0:03:50.1] SC: Awesome. Michael?

[0:03:52.4] MB: I did my studied in Israel at the Technion and my main background is in geometry, different types of geometry; metric geometry, spectral geometry and mostly in applications, computer vision and computer graphics problems. In the recent year as well, probably with many other people, I wouldn't say everybody, they can be following a little bit the type of deep learning. We're trying to apply some of deep learning methods to geometric data, which comes with many of its challenges that there are many similarities and many dissimilarities from the more traditional Euclidean [inaudible 0:04:26.5] data.

[0:04:28.3] SC: I mentioned you are on sabbatical at Harvard. Where are you otherwise affiliated?

[0:04:33.6] MB: Otherwise, I spend most of my time between Switzerland and Israel, full professor at the University of Lugano in Switzerland, the Aviv University in Israel and I also have a position as a principal engineer at the Intel Perceptual Computing.

[0:04:48.7] SC: You sound like a very busy guy.

[0:04:52.5] MB: Tell it to my wife.

[0:04:56.6] SC: The two of you delivered a tutorial. Is it today, right?

[0:04:59.6] JB: Yes.

[0:05:00.0] SC: You did a tutorial. Tell us a little bit about the tutorial you did.

[0:05:04.6] JB: Okay. The tutorial is called Geometric Deep Learning on Graph and Manifolds. In one sentence, what the tutorial is about is how can we leverage the successful techniques that deep learning developed to process images, texts and speech into data and tasks where your input might be a bit more exotic. For example, [inaudible 0:05:32.1] or it could be data from a social network or it could data captured from, let's say, Kinect, where you have a bunch of point clouds in 3D. What the tutorial is trying to do is to say, basically give a picture of our current standing of how deep learning models can be used and developed in this regime and also try to describe some of the future and current open directions.

[0:05:59.6] SC: Michael, in your introduction, you drew one of the key distinctions here, which is Euclidean space versus other geometrics bases. Can you elaborate on that distinction?

[0:06:11.2] MB: Sure. Basically, the difference between Euclidean and non-Euclidean data, one of the main differences is that you don't have the possibility of vector space operation. So in the Euclidean space, you can take your sample, your point and you can add or subtract to points. You cannot do the same thing on a graph for example. You cannot add or subtract to vertices on the graph. Basically, you have more general structures, but also less operations that you can do with these structures. So you need to reinvent many of the building blocks that basically

[inaudible 0:06:42.3] such as convolutions or pulling or that are commonly used in deep learning architectures when you deal with these data.

[0:06:51.4] SC: Joan, why don't you walk us through the general structure of the tutorial? How did you set the context? How did you get started?

[0:06:57.8] JB: Historically, yeah, there's a little clique of researchers that we started to look into this problem a few years ago maybe, like three, four years ago. That was at the time where I was a postdoc in [inaudible 0:07:09.5] Lab in New York. Back then when I was a postdoc. There we, and together with Arthur Szlam who is another organizer of the tutorial, we came up with a simple, let's say, first model that was trying to set up the techniques that later on we realized, and Michael and his group also helped us understand that they were far too naïve, that a lot of things that could be improved up on this first idea.

I guess that since the very beginning we kind of saw that there was a very interesting exchange of ideas and also leveraging the fact that we came from slightly different backgrounds. Michael's group, they have a lot of expertise in geometry and understanding manifolds and we came up with an expertise maybe where we had been working hands-on with convolutional neural networks and some of the ideas that I also worked under in my PhD that involve more the harmonic analysis of this convolutional neural networks. The tutorial came quite naturally. We have been also collaborating in a variety of different projects. Yeah, that's how it came up.

[0:08:18.1] MB: For us, basically the paper that Joan mentioned, also there's a postdoc, was actually an inspiration and that's where we started looking into these kind of problems. We've been working of course on spectral geometry for a long time and did computer graphics geometry processing community, these or similar ideas have been around for a while. Basically, combining it with learning and redefining operations probably in the spectral domain as they did in their influential paper was to some extent eye-opening and basically we took it from there. We rolled together a review paper that appeared in IEEE Signal Processing magazine just this summer, that for us it was also doing some homework. We discovered many works in other communities that existed for quite some time or almost forgotten or not given sufficient attention and maybe even in different communities people calling the same things with different names or

to some extent even I wouldn't dare to say reinventing approaches just without probably being aware of what is done in other communities.

[0:09:23.3] SC: That never happens. What do you mean?

[0:09:24.4] MB: Of course, it always happens, but in these domain — Because it's a new field. We can probably already call it a new field or a new trend or a new niche in machine learning, and because it's new, then basically it's an effort of — There are several seeds that exists in other domains, and I think now they've all started getting and maybe somehow a uniformed picture of what exists and what's the relation between different methods. I think the tutorial is very timely.

[0:09:53.9] SC: You mentioned spectral analysis. When I think of that, I think of things like fast Fourier transforms and alike. How does that fit in to graphs and manifolds and non-Euclidean spaces and all these things?

[0:10:06.5] JB: One way you could think about it is that it's like your dictionary.

[0:10:10.9] SC: We should probably even take a step back and explain what an FFT is for folks who might not be familiar with that.

[0:10:17.2] JB: Yes. Maybe if we take a step sufficiently back we can maybe start with physics. There's a very fundamental equation in physics that governs how things oscillate, right? Like in [inaudible 0:10:30.3] or in a domain. If you take a drum and you hit it, there are some equation that is going to determine how things are going to oscillate. The modes of oscillation, as you might suspect, they are related to the Fourier analysis, right? The equation and the operation and the operator that governs these behavior is called Laplacian. The Laplacian is the — It's a differential operator that in Euclidean domain might look very inoffensive. It's just you take the partial [inaudible 0:11:01.2] with the spectral directions and you just sum everything.

It turns out that from this operator, you can use this as a vehicle to generalize things, because now if you want to generalize convolutions from the Euclidean domain to a non-Euclidean domain, you will have a hard time, because they are defined through something that doesn't

exist in another domain. That if you take one step back and they said, “Okay. I cannot directly use the convolution, but maybe I can step back and use these Laplacian operator as a tool to maybe go from one world to the other.” As it turns out, the Laplacian is an operator that is intrinsically defined with very weak assumptions. What it means is that even on a graph or on a manifold, it’s an operator that exists and it has a very complete and rich structure and status.

[0:11:49.8] SC: The application of the Laplacian to these different domains, this is all work that is done with regard to that domain independent of what we’re trying to do here with machine learning.

[0:12:00.4] JB: Yeah, absolutely. I will not claim that I know exactly one of these things, but I’m sure that many famous physicists from even like the 1900th century were aware of distance and the generality of this operator, right? This is why I think I believe that when people refer to spectral graph theory, most of the data or a big part of it is in developing the properties of Laplacian and related operators in graphs.

[0:12:26.5] SC: As applied to graphs.

[0:12:27.3] JB: Yes.

[0:12:28.4] SC: Okay.

[0:12:29.0] JB: So in that respect, the Fourier transform is not a concept that you can — Of course, if you analyze it in the Euclidean domain, it has a very — Again, a very rich structure and it’s useful beyond our imagination. It can be used to mostly things to the fast Fourier transform. But it’s also an object that exists in general graphs. Perhaps with not the only detail that is — Like a single detail, but might be very important, is that it doesn’t come with a fast algorithm. I would say that this is the main or one of the main technical differences.

[0:13:04.8] SC: When I think of the Fourier transform or the FFT, you have some input signal and you pass it through this Fourier transformation and it basically decomposes it into its frequency parts. How does that apply to a graph? What does that even mean?

[0:13:22.0] MB: On a graph, basically in the Euclidean case, you can — As you said, you can represent a function or a signal as a superposition of sines or cosines basically, harmonic functions. These functions turn to be Eigenfunctions or Eigenvectors of the [inaudible 0:13:35.2] operator. In the Euclidean case, in the one dimensional case, it's just the second or the derivative.

Basically if you replace these Euclidean operation with a non-Euclidean [inaudible 0:13:45.5] with a graph [inaudible 0:13:45.5], or in many fold, that would be what is called differential geometry [inaudible 0:13:50.5] operator, basically you get exactly the same thing. You get Eigenfunctions of these operator that turn to be an orthogonal basis. Basically it's a self-joint operator, so it has an orthogonal Eigen decomposition. The Eigen values play a role of frequencies that are exactly, as Joan mentioned, vibration modes of basically the Eigenvalues that you opted in [inaudible 0:14:13.2] or the spatial parts of the wave equation that governs vibrations of a domain.

[0:14:18.8] SC: Joan, you were saying?

[0:14:19.8] JB: It's just that one way where you can maybe picture this thing in your head is, yeah, if you take like a spherical play that would correspond to a drum as we understand it, it's a drum. Then you could imagine like deforming somehow in the drum and just playing it, intrinsically, you'd still be playing things that can be seen as a superposition of like fundamental waves that would look a bit more funny, right? That would probably adopt and it would be like the formations of the original sines and cosines depending on how you have deformed the original drum.

[0:14:56.1] SC: Interesting. This is interesting stuff. It's bringing me back to my DSP class in grad school. I don't think Eigenvalues and Eigenvectors has really come up much on the podcast either, but we're not going to go into the linear algebra. We'll assume a bunch of that.

Basically, just to catch us up, this is all kind of background to the tutorial. Maybe let's take a moment to talk about applications of this to make it a little bit more concrete for everyone. How does some of these stuff that you're doing applied?

[0:15:31.3] MB: Maybe yeah, we can illustrate a couple of very different applications. One that I'm personally involved in is particle physics. The sort of experiments that people do in a large collider and like in particle accelerators, where there the goal of the game is to say precise things about how the standard model, like some very specific properties of the standard model. The way it works is that you do like a very — An experiment where you clash two particles with very, very large speed, and then they produce what they call a jet. A jet is like a shower of little events that can be detected through a very expensive and very sophisticated detector that looks like a cylinder.

The goal of the game is to say, "Well, given these observation, that it looks like this point cloud in my detector, how can I infer what was the underlying physical theory?" It's really a machine — There, machine learning is a very, very fundamental step for this experimental physicist. There, you use the techniques that we described in the tutorial to essentially learn a data-driven model that looks as input a set of, like a point cloud as modeled in the [inaudible 0:16:42.4] limiter and tries to predict if it was due to theory A or theory B.

[0:16:47.9] SC: Before we go to the other application. When I think about — When I visualize this, I'm visualizing something that is very well within the bounds of Euclidean stuff. It's a three dimensional point cloud. Why do we need to apply the stuff that you've done as supposed to the usual stuff?

[0:17:07.9] MB: Yes. Very good question. The answer is that you're not forced to apply what we do. If you decide to stay in the Euclidean route, you would have to somehow quantify your measurements such that they look like a regular grid. With that, there are two potential risks. The first one is that if the precision that you need, if the little blobs that the particles create are very small, you might need a sampling rate that will make inputs incredibly large, that you will basically be paying a huge price in order to transform your input into something that isn't a regular grid.

[0:17:46.6] SC: Maybe another way to put that is in order to solve this in Euclidean space, you have like a quantization noise that you have to sample more than otherwise if you were to use a different —

[0:17:57.2] MB: Right. There's a tradeoff between — Once you choose a quantization step, there's a tradeoff between how large and how sparse your input is going to look like versus how much resolution you are going to lose. In that respect, the models that we proposed here, they are an alternative that don't have this limitation, that can stay at the native. Let's say, they don't lose any information, because we don't need to quantize within the two going to this regular grid. I would say that there's another potential advantage, is that when you look at your — Or the experiment [inaudible 0:18:30.9] detector as if it was an image in a regular grid, the underlying assumption is that the statistics, the statistics of the input, they are following the same assumption to the statistics of natural images, namely that everything is stationary and that there's a, I would say, a canonical way to compare or like to measure distance between points.

Whereas the physicists, they have a very good and sophisticated notions of how one should be comparing particles. In other words, there's a more physics-aware version of a neural network that — Another way to say it is that for a physicist, it's very important to have a model where you can infuse and you can incorporate prior information about how things should be compared into the model. The graph formulation that we are working on is allowing you to incorporate this thing into the model. Whereas if you used just a quantization and a standard convolutional neural network, it doesn't seem so easy to incorporate and to accommodate for these.

[0:19:33.9] SC: Okay. When I go from my picture of the three dimensional Euclidean for this application to maybe something that's a little bit less traditional, I think of like maybe looking at your points in some kind of spherical coordinate system. Is that what you're doing or is it — Because we keep coming back to graph and I'm trying to wrap my head around where graph —

[0:19:56.1] JB: Yes. Here, the graph enters the game as just like a data structure that you use to relate particles. The thing that your input is discrete, it contains a number of particles that can be variable depending on [inaudible 0:20:09.7]. What you'll learn is a network that learns how to relate and propagate information across the set of particles. It doesn't see it into these extrinsic Euclidean space. Of course, you could say, "Yeah, I can see my input as being N-particles. I can also it does," as we say, "just put everything in a sphere and then I just see it as a function, like as an image in that sphere." Then the notion that they have N-particles completely disappears, and then you connect everything with everything.

I would say that it's not that there are one approach that is systematically better than the other. Really, there's context in which you might want to use one formulation versus the other. Why I see here is that we are just providing another tool that now our practitioners can use and maybe they can combine with the other one. We also have reasons to believe that in some context, one formulation might scale better than the other. For example, [inaudible 0:21:10.6] of having a momenta of four dimensions. I had momenta of like eight dimensions. In our case, the architectural changes to the model would be minimal. There's nothing in our scaling that it depends on the — Like it depends very weakly on the dimension of the input. Whereas if you had to do the quantization on the domain, you would pay a huge price for this module.

[0:21:39.1] SC: Interesting. You're going to give second example?

[0:21:39.1] JB: I can give a second example, or Michael can give an example, as you prefer.

[0:21:46.0] SC: Michael, why don't you give one?

[0:21:46.8] MB: Yeah. So I can give you as an example a paper that actually we'll be presenting tomorrow here at NIPS, and the example is recommender systems. Probably the most famous example is the Netflix problem, that Netflix is a movie rental company and they have probably tens of millions of users and probably hundreds of thousands of different movies.

The users can give different scores to movies whether they like the movie or not, let's say, on a scale between 0 and 9. Basically you can describe this information as a huge metrics that is very sparse assembled, because of course even if you watch continuously the movies throughout your entire lifetime, you'll probably watch just a small percentage of what they have.

They want to fill in the missing entries of this huge matrix. Basically they try to interpolate it. The standard approached is use algebraic techniques basically, try to fit a low dimensional to these data, minimizing the matrix [inaudible 0:22:40.0] or more correctly basically the convex [inaudible 0:22:42.3], the so-called nuclear norm.

This problem formation doesn't have any geometric structure, for example. We can shuffle the columns and the rows of the matrix or if you remove one of the columns, there are infinitely ways you can fill it in. If you have some either side information or [inaudible 0:22:59.8] from the data, some notion of affinity between users or items or actually both as you can represent as a graph. Think of a social network and maybe a little bit naïve model that friends have similar taste. This already allows you to think of the matrices some kind of space with geometric structure. You can talk about notions like smoothness. Basically we can say that the values in the matrix significantly or insignificantly when you go from one vertex on the graph to another vertex on the graph. We can actually learn optimal filters, spectral filters for example on this graph. Actually, it's a product of two graphs. The best analogy from signal processing will be a two dimensional Fourier transform.

[0:23:41.5] SC: Two dimensional — Okay. Two dimensional Fourier transform.

[0:23:44.2] SC: Yeah. Think of the Fourier transform of an image, right? You apply Fourier transform to the columns and rows of the image. Here, instead of Euclidean structure, its rows and columns have matrix that lives on two different graphs. Basically, it's rows, for example, represent items or movies and the columns represent users. These are two different graphs. You can apply filters in these frequency domain that is now basically characterized by the Eigenvalues of the two [inaudible 0:24:10.2] of the rows and the column graphs.

Basically this way you can't have a better way of filling in the missing elements of the matrix that accounts for the similarities between different users and movies.

[0:24:24.2] SC: How specifically does the frequency domain, the Fourier transform tied into the graph itself in this context?

[0:24:33.6] MB: It's very similar to what Joan described before. Basically, the spectral definition of convolution, right? It can do convolution in the spectral domain, the classical conversion. This is what we call the convolution theorem in signal processing that you can implement filtering or convolution in the frequency domain as a product point, [inaudible 0:24:51.9] transforms. This is what we usually do in standard signal processing, because we can efficiently compute the

Fourier transform using [inaudible 0:24:58.6]. You can do the same thing for images. You can do two dimensional filters in the frequency domain.

Basically, extend this analogy to a matrix data. Basically you can think of it as an image, but the domains where the rows and the columns of these image life, they have graph structure, sort of standard Euclidean structure.

[0:25:19.6] SC: If someone has a problem where they've got some set or sets of entities that have some inherent structure one relative to the other as supposed to some of the more traditional image, an image or their data types. That's an area where they can be looking at an approach like this and it might give them some advantage.

[0:25:42.8] JB: Absolutely. Yes. You mentioned a setting that is now approached and defined through different names. Now there's something that is becoming very popular recently, it's called meta-learning, where it's this idea that maybe rather than using a traditional supervised learning where I have very few number of samples and very few number of labels and then the environment is [inaudible 0:26:06.5] so we change it all again and again. Maybe what I can learn is a rule that by looking at how maybe the inputs relate to each other, even if the individual distribution of the labels and the images change, maybe the underlying relationship between labels and images that I need to learn is something that I can leverage by exploring more and more data.

Once you start setting up problems using this formulation, you end up with a task where you have to learn how to relate different things, different objects. This is a terrain where it's very natural to look at our model. We have a recent paper that is currently on the review where we think about what's called a [inaudible 0:26:48.2] learning problem using this model, using a graph neural network.

What's interesting is that somehow it generalizes and it includes these particular cases some of the models that people have been using and developing in the recent years to attack this problem. This is just to say that there are many tasks that you could imagine across AI and across sciences. The underlying thing that you need to learn is how to relate objects to each other.

Once you have to do this relational task, then a natural data structure for that is a graph or just a point cloud. I expect that we'll see more and more applications of the technology in the near future.

[0:27:32.4] SC: One of the things that this conversation brings to mind is a recent conversation I had in fact here at NIPS with [inaudible 0:27:39.9], who studies statistical relational AI. Are you familiar with that line of research and how that relates to this?

[0:27:48.2] JB: Not really.

[0:27:49.7] SC: Okay. You should check it out. It's similar thinking. He's also looking at graph-based approaches and applying them to the healthcare domain and other domains.

In the case of — Any of the examples we've talked about, I can — We've already talked about some of the advantages of this approach over traditional approaches. One of them is that in the case of the particle physics work, it's maybe more intuitive for the physicist, because it's much more closely to the tools and the way they're used to thinking about the domain. What are the other advantages of this approach? Are there performance advantages in terms of either computational or mode accuracy or things like that?

[0:28:34.9] MB: Yes. I would say that the main advantage is definitely the fact that it's more general. You can — There are problems in which it might be your only choice. There's no Euclidean alternative to do. I would say that whenever you're — You could ask the question, "Well, if I just forget about the grid structure of an image and just treat it as a graph and I run my model, this model, is it going to do better than the CNN?" I would say that the answer is no.

[0:29:03.9] SC: Yeah, and that's not really the point.

[0:29:05.8] JB: It's not really the point. I would say that the main strength of the model is really to respect the invariance of the data hash. For example, if you are treating — If you are just observing a point cloud, you know that the order [inaudible 0:29:20.8] the point cloud is completely irrelevant. Therefore, if you can certify that your model is going to exactly — Exactly

the same model independent of how the input are permitted, then you are kind of respecting this natural environment of the data. I would say that relative to models that are, for example, maybe using sequential networks, like for example recurrent neural networks. Here, it could be that eventually these models based on graph and sets are going to be more sample efficient and maybe they can give you better performance precisely because they are a bit more tailored to the data format, like to the input data format.

[0:30:00.1] MB: Maybe a good example would be applications from the domain of graphics and computer vision, and in particular analysis of deformable 3D shapes. This is actually — I think it's a good illustration, because you can treat such objects in two different ways. You can treat them as Euclidean objects, basically the things that live in three dimensional space, right? You can apply standard, let's say convolutional neural networks maybe on [inaudible 0:30:22.0] representations of these objects, or you can think of them intrinsically from the perspective of differential geometry, basically model them as manifolds. What you gain in this way or resourcing to this kind of architectures is you gain a deformation in variants. Basically, your model is by construction in variants in elastic deformations of the shape. If your task, for example, is deformation in variant correspondence or deformation in variant similarity, it means that you can deal with way less training example, because you don't need to show to the network all the possible deformations that the shapes can undergo in order to learn these deformations from examples. Basically you'd have invariance built into the model. The difference can be very dramatic. The difference can be in order of magnitude, less training samples.

[0:31:08.3] SC: In other words, the approach you're taking to model, because it's more tailored to the problem domain, it kind of restricts your search base and you don't have to provide examples for things that wouldn't really exist in the domain itself, but do exist geometrically in Euclidean space.

[0:31:28.1] MB: Exactly, or maybe a better way to say it is that you try to model axiomatically as much as possible or as much as makes sense in your specific problem and everything that cannot be modeled axiomatically because, of course, series limitation to what you can model basically in an hand-crafted way. Everything that deviates from your model, you learn.

[0:31:49.1] SC: Okay. Are there other topics that you covered in the tutorial that we haven't touched on yet?

[0:31:54.0] JB: Just very briefly. One other potential area of application that we are currently exploring to what extent if you now have a language to learn our graph-based structure, you can use it for combinatorial optimization problems that are naturally [inaudible 0:32:09.2] graphs. This is a completely different domain of application, because there the goal is not so much to solve a task that you don't know how to solve. It's more to solve to approximate the task faster.

[0:32:20.8] SC: We're talking like traveling salesman and these kinds of graph.

[0:32:23.3] JB: Yes. Exactly. We briefly touched upon one of such problems. The tutorial, which is the quadratic assignment problem.

[0:32:31.4] SC: The what?

[0:32:32.2] JB: The quadratic assignment problem, which contains the travel status. Travel [inaudible 0:32:35.9] problem is that you can think of it as a particular case of that problem. There, the general set up is really an instance of this trend that you can always have this analogy between an algorithm to solve a task and a neural network, and this analogy works by looking at the algorithm and then unrolling, typically the algorithm involves a series of iterative steps, iterations. You can just see these iterations as being different layers of a network. Then once you this analogy then you can try to study like tradeoffs between a computation and accuracy by just replacing the guarantees that the algorithm gives you by just a data-driven approach where you just feed the parameters of the network to a dataset of solved problems.

This is an interesting and potentially also useful and — Because in many domains, especially when it comes to combinatorial optimization, there are problems in which it's still an open research area how to come up with efficient approximation of impractical problems.

Here, one of the potential uses of what we presented is well now we are providing a family of, let's say, trainable architectures that can be used to guide and to provide good tradeoffs

between accuracy and complexity for problems such as the traveling salesman or other [inaudible 0:33:56.5].

[0:33:56.6] SC: Interesting. To paraphrase that, what you've done is your research is kind of providing a way to express graph-oriented problems in terms of neural networks. Traveling salesman, map coloring, all these other combinatorial kind of graph problems. They're very typically very difficult to solve exactly, and so there all kinds of approximations and heuristics. For a certain level of complexity, those don't work very well. Now, your research applied to them gives you a way to solve these using neural networks.

[0:34:32.2] MB: Yes, I would not —

[0:34:33.6] SC: Potentially.

[0:34:34.1] JB: Potentially, exactly. It's a question mark and I think it's a question mark that its worth exploring, because in some applications it could be useful that, for example, not just have a single algorithm with a single heuristic, but to have like a toggle that you can select between how many cycles do you want to spin, versus how much accurate do you want the solution to be and be able to learn adaptive tradeoffs and all these things are interesting. Then there's another declination of this area of research that is a bit more going into the theoretical computer science, namely what extent the models that we learned could be interpretative as algorithms that we still don't know. This might be — It could well be that it doesn't work, because it relies on this fact that can we interpret or can we uncover what the neural network is learning. We know that this is typically a hard thing to do. Even for a convolutional neural network. We don't really know how the network figures out how to solve the problem. What I'm saying is that in some context, it could be interesting to try to understand and analyze kind of things that the network learn.

[0:35:47.6] SC: We have these graph problems. There are graph problems in computer science as well. Your research allows us to express those as neural networks. If we could then peer in to the neural network, that might give us some insight into these computer science problems that we're trying to model in the first place.

[0:36:04.1] JB: Yes.

[0:36:05.4] SC: Interesting. How about implementation? I'm imagining at this point in time you can't just do TensorFlow, tf.graphsolver and do this. How does that work?

[0:36:18.2] MB: To some extent, we are trying to leverage existing tools not to reinvent the wheel. Basically, the underlying framework is a standard one. We use TensorFlow, for example. We just create some custom things and then boil down again to some standard operations like matrix multiplication. Basically, the short answer is yes.

[0:36:38.1] SC: It is that easy.

[0:36:39.3] MB: It is built on top of standards frameworks.

[0:36:42.3] SC: Okay.

[0:36:44.3] JB: There are minor, but potential profound differences in the fact that the — Scaling up, like using these models on large graphs or large domains involves matrix multiplication with matrix that are locked. The structure that we have is that this matrixes are sparse, and so hopefully we see more and more integration of sparsely neural algebra into [inaudible 0:37:08.5] in TensorFlow , etc., but there's a fundamental difference that maybe the hardware, like GPUs, they are excelling at a specific form of operation that is not very fairly with sparse matrix multiplication. Again, I'm not an expert in like this low level implementation. This, I would say, is one of the main differences between running a complement or running a graph convolution.

[0:37:38.4] SC: So you've released some code that works under TensorFlow, but it isn't necessarily amenable to scaling up just yet. There are stuff that needs to be figured out. Maybe one way to get a sense for the complexity to this, has anyone like beyond the two of you and folks that are like deep in this research use this? Are you aware of any like external arms-length applications?

[0:38:04.5] JB: Different people from different communities try to use — I wouldn't say that it's extremely popular yet, but probably it starts to become. Many different domains, many different

applications can be — Problems in these domains can be formulated using graphs, and they're very generic and very convenient representation of any kind of relations or interactions that you can think of. That's really very generic framework of describing certain types of data.

Yeah, people that are even not experts in machine learning that come from different domains [inaudible 0:38:39.6] certain applications, basically they see that graphs allow to formulate their problems in a natural way and they are curious to try out these approaches.

[0:38:50.2] SC: Okay. Great. This is really exciting stuff. Where can folks go to learn more about it, download the TensofFlow code or read some of the papers?

[0:39:01.1] JB: We have a dedicated website that is easy to remember, it's geometricdeeplearning.com.

[0:39:05.8] SC: Geometricdeeplearning.com. Awesome.

[0:39:08.2] JB: Yeah. Where I think the idea is to have all the tutorial material. We have the review paper that Michael mentioned. We have also the recent literature by not just us, but other groups that use the tools. Then we're also going to have — In two months, we're going to have an [inaudible 0:39:24.1] workshop here at Los Angeles where I think I'm also looking forward to it, because it's what you are saying, that people from different domains and people from different disciplines will come together and essentially present their data problem of their formulation, and what I'm expecting is that we are going to see more and more this realization that actually the models and the tools can be used across more domains than maybe we are expecting.

[0:39:55.5] SC: Okay. Awesome. Great! Joan, Michael, thank you so much for taking the time to chat. I enjoyed the conversation.

[0:40:02.8] JB: Thank you very much.

[0:40:02.8] MB: Thank you.

[END OF INTERVIEW]

[0:40:07.5] SC: All right everyone, that's our show for today. Thanks so much for listening and for your continued feedback and support. To follow along with the NIPS Series, visit twimlai.com/nips2017. To enter our TWiML 1 Mil context, visit twimlai.com/twiml1mil.

Of course, we'd be delighted to hear from you either via a comment on the show notes page or via a Tweet to @twimlai or @samcharrington.

Thanks again once again to Intel Nervana for their sponsorship of this series. To learn more about the Intel Nervana NNP and the other things Intel has been up to in the AI arena, visit intelnervana.com.

As I mentioned a few weeks back, this will be our final series of shows for the year, so take your time and take it all in and get caught up on any of the old pods you've been saving up. Happy holidays and happy New Year. See you in 2018. Of course, thanks once again for listening, and catch you next time.

[END]