GILBERT STRANG: I'm determined to tell you something about the convolution rule. I just get close to it, but haven't quite got there. And actually, I'd like to say something also about why convolution is so important. I mentioned here a paper about images in deep learning by-- it has three authors, and these are two of them. Maybe you recognize Hinton's name. He's originally English. He was in San Diego for quite a few years, and now he's in Canada. So Toronto and Montreal are big centers now for deep learning. And he's really one of the leaders, and so is Sutskever.

So maybe you know that the sort of progress of deep learning can often be measured in these competitions that are held about every year for how well does-- people design and execute a whole neural net. And this was a competition about images. So that is really demanding, because, as I said last time, an image has so many samples, so many pixels that the computational problem is enormous. And that's when you would go to convolution neural nets, CNN, because a convolutional net takes fewer weights, because of the same weight as appearing along diagonals. It doesn't need a full matrix of weights, just one top row of weights.

Anyway, so this is one of the historical papers in the history of deep learning. I'll just read a couple of sentences. We trained-- so this is the abstract. We trained a large deep convolutional neural network. I'll just say that it ran for five days on two GPUs. So it was an enormous problem, as we'll see. So we trained a large deep network, CNN, to classify 1.2 million high res images in ImageNet. So ImageNet is a source of millions of images.

And on the test data, they-- well, the last sentence is maybe a key. We entered a variant of this model in the competition, 2012 competition, and we achieved a winning top five test error rate of 15% compared to 26% for the second place team. So 15% error. They got 26% was the best that the rest of the world did.

And so that-- and when he shows the network, you realize what's gone into it. It has convolution layers, and it has some normal layers, and it has max pooling layers to cut the dimension down a little bit. And half the samples go on one GPU and half another. And at

certain points, layers interconnect between the two GPUs.

And so to reduce overfitting-- you remember that. It's a key problem is to reduce overfitting in the fully connected layers. Those are the ordinary layers with full weight matrices. We employed a recently developed regularization called dropout. So dropout is a tool which, if you're in this world, you-- I think Hinton proposed it, again, by seeing that it worked. It's just a careful dropout of some of the data. It reduces the amount of data, and it doesn't harm the problem.

So the neural network has 60 million parameters. 60 million. With 650,000 neurons, five convolutional layers, and three fully connected layers. I just mention this. If you just Google these two names on the web, this paper would come up. So we're talking about the real thing here. Convolution is something everybody wants to understand.

And I'd like to-- since I've started several days ago, and I'd like to remember what convolution means. Let me-- so if I convolve two vectors and I look for the k-th component of the answer, the formula is I add up all the c's times d's where are index i plus j adds to k. Why do you do such a thing? Because c might be represented by a polynomial, say x plus c Nx to the N. And d might be represented by another 1d 1x plus dm x to the m, let's say.

And convolution arises when I multiply those polynomials. Because for a typical-- and then collect terms. Because a typical power of x, say x to the k, the coefficients are-- well, how do we get x to the k in multiplying these? I multiply c0 times a dk. Somewhere in here would be a dk x to the k. So a c0 times a dk would give me an x to the k term. And a c1 times-- everybody sees this coming now? c1 has an x in it already. So over there, we would look at dk minus 1 with one less x. So it would be c1 dk minus 1.

This is just what you do when you multiply a polynomial. And the point is that the way we recognize those terms is that the exponents 0 and k, the exponents 1 and k minus 1, always add to k. So that's where this formula comes from. We take a c times a d hiding behind our cx to the i and a dj x to the j and when i plus j is k, this is x to the k. And that's the term we're capturing. So this is the coefficient of that term.

And let me write it as a slightly different way, where you actually see even more clearly convolution operating. So j is k minus i, right? So it's the sum of cidj, but the j has to be k minus i. So this is the way to remember the formula for the coefficients in c star d in the convolution. You look at c's times d's. It's a form of multiplication. It comes from ordinary

multiplication of polynomials. And when you collect terms, you're collecting c, the i-th c and the k minus id, and you're taking all possible i's. So it's a sum over all possible i's there to give you the k-th answer.

Well, just to see if you got the idea, what would be the convolution of two functions? Suppose I have a function f of x. And I want I convolve that with a function g of x. OK. And notice that I have not circled this symbol. So I'm not doing periodic convolution. I'm just doing straightforward convolution. So what are we going to have in the case of two functions? What would that mean, a convolution of functions?

I'm in parallel here with a convolution of two vectors. So think of these now have become functions. The case component has become-- really, I should say f star g at x. That's really the parallel to this. So let me. So I'm telling you the answer at x. Here I told you the answer at k. The k-th component looks like that. What does the x value of the convolution look like for functions?

OK, I'm just going to do this. I'm going to do the same as this. Instead of summing, what will I do? Integrate. Instead of c sub i, I'll have f of x. The index i is changing over to the continuous variable x. And now g instead of dk minus i, what do I have here? So it's the k minus i component. That will go to-- let me just write it down-- t minus x.

So in this translation, f is being translated to c. Or sorry, f corresponds to c. g corresponds to d. k corresponds to x. Oh no, sorry. i corresponds to x. And k minus i corresponds to t minus x. So k corresponds to t. This would be the convolution of two functions. Oh, it's a function of t. Bad notation.

The t is sort of the amount of shift. See, I've shifted g. I've reversed it. I've flipped it and shifting it by different amounts t. It's what you have in a filter. It's just also always present in signal processing. So that that would be a definition. Or I could, if you like, if you want an x variable to come out, let me make an x variable come out by exchanging t and x. So this would be x minus t dt. I like that, actually, a little better. And it's the integral over t minus infinity to infinity if our functions were on the whole line.

So there will be a convolution rule for that. This will connect to the Fourier transform of the two functions. Over here, I'm connecting it to the discrete Fourier transform of the two functions. And I've been making the convolution cyclic. So what does-- can I add cyclic now? This is ordinary convolution. This is what you had in the first lab, I think, from Raj Rao.

The first lab, you remember you had to figure out how many components the convolution would have? And you didn't make it cyclic. So a cyclic convolution, if this has n components and this has n components, then the convolution has n components. Because keeping n is the key number there, the length of the period.

And similarly, over here, if f is 2 pi periodic and g is 2 pi periodic, then we might want to do a periodic convolution and bring it-- get an answer that also has 2 pi period 2 pi. So you could compute the convolution of sine x with cos x, for example.

OK, let's stick with vectors. So what's the deal when I make it cyclic? When I make it cyclic, then in this multiplication, I really should use-- I've introduced w as that instead of x. So cyclic. x becomes this number w, which is e to the 2 pi i over n and has the property then that w to the n-th is 1 so that all vectors of length greater than n can be folded back using this rule to a vector of length n. So we get a cyclic guy.

So how does that change the answer? Well, I only want k going from 0 to n minus 1 in the cyclic case. I don't want infinitely many components. I've got to bring them back again. And let me just say what the rule would be. You just ask, say, i plus j. You would look at that modulo n. That's what a number theory person would call it. We only look at the remainder when we divide by n. So now the sums go only from 0 to n minus 1, and I only get an answer from 0 to n minus 1. Well, I've done that pretty quickly. That's if I wanted to do justice to--

So the difference between non-periodic. So non-periodic and periodic will be the difference between-- so I have some number t0 on the diagonals. t1, t2, t minus 1, t minus 2, and so on. Constant diagonals. So the key name there is Toeplitz. And if it's periodic, then I have, I'll say, c, c, c. And then the next one will be c1, c1, coming around to c1. And c2 coming around. So it's n by n period n. So it's a circulant matrix. N by N.

OK. That's the big picture. And I think in that first lab, you were asked to do the non-circulant case. Because that's the one where you have to do a little patience. What will be the length? Yeah, what would be the length of a non-circulant? So not circulant. Now, suppose the c vector has p components and the d vector has q components. How many components in their convolution? Shall I write that question down? Because that brings out the difference here.

So if I have p, if c has P components, d has q components, then the convolution of c and d has how many? So I'm multiplying. So it's really this corresponds to a polynomial of degree p

minus 1, right? Polynomials of degree p minus 1.

And this guy would be degree q minus 1. Degree q minus 1. And when I multiply them, what's the degree? Just add. And how many coefficients? Well, one more I have to remember for that stupid 0 order term. So this would have p plus q minus 1 components. So that would have been the number that you've somehow had to work out in that first lab.

So that if this had n components and this had n, this would have 2n minus 1. It's just what you would have-- like you say 3 plus x times 1 plus 2x. In this case, p is 2, q is two, two components, two components. And if I multiply those, I get 3x and 6x is 7x and 2x squared. And so I have 2 plus 2 minus 1 equals 3 components. The constant x and x squared. Yeah, clear, right.

Yeah, so that's not the-- that's what I would get if I multiplied these matrices, if I had a two diagonal matrix, Toeplitz matrix, times a two diagonal Toeplitz matrix, that would give me a three diagonal answer. But if I am doing it periodically, I would only have two. That 2x squared would come back if I-- come back as a 2. so I just have 5 plus 7x. Right, good, good, good.

OK. So that's a reminder of what convolution is. Cyclic and non-cyclic, vectors and functions. OK, then eigenvalues and eigenvectors are the next step, and then the convolution rule is the last step. So eigenvectors. Eigenvectors of the circulant. Of course, I can only do square matrices.

So I'm doing the periodic case. So the eigenvectors are the columns of the eigenvector matrix. And I'm going to call it F for Fourier. So F is-- the first eigenvector is all 1s. An x eigenvector is the fourth root of 1, then the square root of 1, i6, i8, i fourth, i6, and finally, 1 i cubed i sixth i ninth. OK, that's F. Those are the four eigenvectors of the permutation p and of any polynomial in p. So my circulant is some c0 i plus c 1p plus c 2p squared and c3 pq.

OK. And finally, this is the step we've been almost ready to do but didn't quite do. What are the eigenvectors-- what eigenvectors are its eigenvectors? So those are the eigenvectors of p. And now we have just a combination of p's. So I think the eigenvectors I just multiply. I take that same combination of the eigenvectors. Does that look right?

So sorry. I'm sorry. Its eigenvectors, they're the columns of f. The question I meant to ask is what are its eigenvalues? That's the key question. What are the eigenvalues? And I think that if I just multiply F times c, I get the eigenvalues of the matrix C.

That's the beauty. That's the nice formula. If my matrix is just P alone, then this is 0, 1, 0, 0, and I get 1, i, i squared, i cubed. But if c is some other combination of the p's, then I take the same combination of the eigenvectors to see-- yeah. Do you see it?

So I'm claiming that I'll get four eigenvalues of C from this multiplication. So of course, if there's only c0, then I only get c0, c0, c0, c0. It's four times repeated. But if it's this combination, then that matrix multiplication takes the same combination of-- this is a combination of the eigenvectors. And that gives us the right thing. OK. Now I just have one more step for this convolution rule, and then I'm happy.

Really, the convolution rule is stating what we-- it's stating a relation between multiplication, which we saw here, and the convolution, which we saw for the coefficients. So the convolution rule is a connection between multiplying and convolution. And so let me say what that convolution rule is and let me write it correctly.

So here I take a cyclic convolution. I'm dealing with square matrices. Everything is cyclic here. And then I get-- if I multiply by F, what do I have now? What does that represent? This was c and d, and I convolve them. So I got another circulant matrix.

So up here, the multiplication of matrices is C times D. I want to connect multiplying those matrices with convolving the c's. I want to make that connection. And that connection is the convolution rule. So this would be the eigenvalues of CD.

Let's just pause there. Why am I looking at the eigenvalues of CD? Because if I do that multiplication, I get another Toeplitz matrix, C times D. And the polynomial-- the coefficients associated on the diagonals of C times D are the coefficients of the convolution. So its diagonals come from convolving c with d cyclically. OK.

Now I want to find the same eigenvalues in a second way and match-- and the equation will be the convolution rule. So how can I find the eigenvalues of CD? Well, amazingly, they are the eigenvalues of C times the eigenvalues of D. I'm going to test this rule on 2 by 2. So you'll see everything happening. So this is the main-- this is the fact that I want to use.

Because C and D commute. C and D commute. They have the same eigenvectors. And then the eigenvalues just multiply. So I can multiply. I can get that in a second way by taking the eigenvalues of c and multiplying those by the eigenvalues of d. And I multiply component by component. I multiply the eigenvalue for the all 1s vector by the eigenvalue for the all 1s

vector.

Do you know this MATLAB command? Component by component multiplication? This is an important one. There's a guy's name is also associated with that. So that's a vector. That's a vector. And what comes out of that operation? If I have a vector with three components. So n is 3 here. And I do point star or dot star. I'm not sure what people usually say. Component by component, a three component vector times a three component vector, I get a three component vector, just like that. So this is the convolution rule. That's the convolution rule.

And the proof is the fact that when matrices commute, the eigenvalues of the product are just these eigenvalues times these eigenvalues, because they have the same-- the eigenvectors are always the same here for all these circulants. So there's the convolution rule that I can convolve and then transform. Or I can transform separately and then multiply.

So I just maybe better right that convolution rule. Let's call it the C rule. Convolve then transform by F. Or transform separately by F. And then multiply point one. Element by element. Component by component. OK. So that's the convolution rule.

And why is it sort of-- why is it so important? Because transforming by F, multiplying by the Fourier matrix, is extremely fast by the FFT. So it's useful because of the FFT, the Fast Fourier Transform, to multiply. Or to transform. Whichever. Equal to transform. Multiply by F transform. So it's the presence of the FFT that makes this-- it gives us really two different ways to do it.

In fact, which is the faster way? So we can produce the same result this way or this way. And if I don't count the cost of-- if the cost of multiplying by F is low, because I have the FFT, which would you do? Which would you do? So let me just think aloud before we answer that question, and then we're good.

So my vectors have n components. So one way I can do is to do convolution. How many steps is that? If I take a vector with n components and I convolve with a vector with n components, how many little multiplications do I have to do? N squared, right? Because each of the c's has to multiply each of the d's. So that takes N squared. And Fourier is cheap. It's N log N. Log to base 2. So the left hand side is effectively N cubed.

What about this one? How many to do these two guys? To find the Fourier transform to multiply by the matrix F. OK, those are fast again. That's just I've got two multiplications by F.

So that's 2 N log N. And what's the cost of this? I have a vector with n components. Dot star vector. Another vector with n components.

How many little multiplications do I have to do for a Hadamard product or a component by component product? N, only n. Plus N. Yeah, maybe I should have made that plus. I had two. No, I had one N log N. Plus it took N squared to find that vector. And then N log N. So it's effectively N squared. But this one where I do the N log N twice and then it only takes me N more. So this is the fast way.

So if you wanted to multiply two really big, long integers, as you would want to do in cryptography, if you had two long integers, say, of length 125, 126, 128 components, to multiply those, you would be better off to separately take the cyclic transform of each of those 128 guys and do it this way. Take the transforms, do the component by component product, and then transform back to get that. The convolution rule is what makes that go.

Oh, one more thought, I guess, about all this convolution stuff. Suppose we're in 2D. We have to think what is a two dimensional convolution? What does this become in two dimensions? Suppose we have functions. So now I'm gonna do 2D functions of x and y. Periodic or not periodic. But what's a convolution? What's the operation we have to do in two dimensions?

Well, it's a double integral, of course. t and u. We would do f of t and u times g of x minus ty minus u dtdu. And that would produce a function. So I'm convolving a function of x and y with another function of x and y. And again, I'm looking for this. This is the key to watch for. x minus t. y minus u. That's the signal of a convolution integral. So that's what we would have in 2D.

In general, So maybe now my final thought is to move to think about two dimensional matrices and their products and so on. And this is why you need them. Because if you have two dimensional signals, then the components fit into a matrix. And we just have to operate in both dimensions.

So the key operation in 2D is in MATLAB. The MATLAB command that you need to know to get-- if you know what you're doing in 1D and you want to do it in 2D, the MATLAB command is Kron. So imagine we have one dimensional matrices A and B. And so those are in 1D, and we want to produce a natural two dimensional matrix. So these will be N by N. And we want the sort of natural product, let me call it K for Kron, which will be N squared by N squared.

I want to create a 2D matrix connected to an image that's N in each direction. So it has N

squared pixels. These are 1D signals, and K is a 2D one. And this K would be the-- this is the operation to know. Given two one dimensional n by n matrices, Kron produces an N squared by N squared matrix. It's the operation to know. So I'll just write it, and if you know what Kron is, then you know it before I write it.

So I want to produce a big matrix, N squared by N squared. Somehow appropriately multiplying these two guys. And the appropriate way to do it is to take a11 and multiply it by B. So there, what do I have? What size have I got there already just in that one corner? N by N, right? It's a number of times an N by N matrix. Then a12 times B. That's another N by N matrix. Up to a1N times B. So I have now-- sorry, cap N.

So I have cap N matrices in a row. Each of those matrices is N by N. So that row has length n squared. And of course, the next row is-- I've allowed myself to number from 1 to N, but very often that numbering should be 0 to n minus 1. And finally on down here down to an1 B to aNN B. So so that's the N squared by N squared matrix that you would need to know.

For example, if you wanted to do a two dimensional Fourier transform, that would be-- yeah, so what would a two dimensional Fourier transform produce? What matrix? Is this the matrix you would use for a 2D? I haven't sort of got started properly on 2D Fourier transforms. So would it be F times F? So let me write down the full name of this guy. Kronecker. So it's called the Kronecker product. It's just the right thing to know in moving from one dimension to two dimensions.

For example. Let me do an example. Oops, that's full. Have I got one board left? Yeah. So here's a standard matrix. Call it A. 2s and minus 1s. So that corresponds to a second derivative or actually minus a second derivative.

Now, suppose I have another, the same matrix, corresponding to second derivatives in the y direction. Same. And what I really want to do is both. I want to have a matrix K that corresponds to minus the second in the x direction minus the second in the y. So this is the Laplace. Laplacian. Which is all over the place in differential equations.

At a typical point, I want to do minus one of these, two of these minus one of those in the x direction, and I want to add to that minus 1. Now that 2 becomes a 4 and minus 1 in the y direction. So I'm looking for the 2 by 2 matrix-- sorry, the two dimensional matrix that takes-- that does that five point scheme. Five weights at each point. It takes four of the-- on the diagonal and minus 1 on the four neighbors.

And the operation that would do that would be you would use Kron. It wouldn't be Kron of A B. That would just-- K of A B is not what I-- a Kron of A B is not what I want. Yeah, that would do one and then the other one. And then that would probably produce nine non-zeroes. I want something that adds here. So I want Kron of A times the identity. That gives me the two dimensional thing for this part.

And then I'll add on Kron of I B for the vertical derivative, the derivatives in the y direction. So that's called a Kronecker sum. The other was a Kronecker product. So that would be a Kronecker product. This would be another Kronecker product, and the total is called the Kronecker sum.

OK. I wanted just to get those notations out. Because really, Fourier transforming is such a central operation in all of applied math, and especially in signal processing.

OK, so I'm good for it today. Let's see. I've got one volunteer so far to talk about a project. Can I encourage an email from anybody that doesn't-- you don't have to be a superstar. You're just willing to do it.

Tell us something about what you've learned. Get comments from the audience. And 10 or 15 minutes is all I'm thinking about. OK, I'll let you send me an email if you'd like to tell us that and get some feedback. OK, good. So I'll see you Wednesday. Thanks.