SURYA GANGULI: I'm going to talk about statistical physics of deep learning, essentially. So this is some ongoing work in my lab that was really motivated by trying to understand how neural networks and infants learn categories. And then it sort of led to a bunch of results in deep learning that involved statistical physics.

So I wanted to just introduce my lab a little bit. I'm an interloper from the Methods in Computational Neuroscience Summer School where I tend to spend a month. And so there, you know, the flavor of research that we do there and the flavor of research we do in our lab is sort of drilling down into neural mechanisms underlying well-defined computations. And we've been working on that. You know, I spent a lot of time talking to neurophysiologists especially at Stanford where I am.

So we have a whole bunch of collaborations going on now involving understanding neural computation. So, for example, the retina itself is actually a deep neural circuit already, because there's an intervening layer of neurons, the bipolar cells and amacrine cells, that intervene between the photoreceptors and the ganglion cells. And oftentimes, what we do is we shine light on the photo receptors and we measure the ganglion cells, but we have no clue what's going on in the interior.

So we've developed computational methods that can successfully computationally reconstruct what's going on in the interior of this putative deep neural network even though we don't have access to these things. And we actually infer the existence and properties of intermediate neurons here. And those properties are sort of similar to what's been previously recorded when people do directly record from, say, the bipolar cells.

In the Clandinin Lab, we've sort of been unraveling the computations underlying a motion vision. So you know when you swat a fly, it's really hard. Because they can really quickly detect motion coming towards it. And they fly away.

You know, so there's been lots of work on what kinds of algorithms might underlie motion

estimation-- for example, the Reichardt correlator the, Barlow-Levick model and so forth. We've been applying systems identification techniques to whole brain calcium imaging data-- well, whole brain meaning from the fly visual circuit. And we just literally identify the computation. And we find that it's sort of none of the above. It's a mixture of all previous approaches.

So grid cells-- we have some results on grid cells. So these are these famous cells that resulted in a Nobel Prize recently. We can actually show that these grid cells maintain their spatial coherence, because the rat and the mouse are always interacting with the boundaries. And the boundaries actually correct the rat's internal estimate of position.

And were it not for these interactions with the boundaries, the grid cells would rapidly decohere on the time scale of minutes, you know, like less than a minute. And so it's actually quite interesting. We can show that whenever the rat encounters a boundary, it corrects its internal estimate of position perpendicular to the boundary.

But it doesn't parallel. And it doesn't, because it can't. Because when it hits the boundary, it receives no information about where it is parallel to the boundary, but it receives information in this direction.

And then with the Shenoy Lab, we've been looking at I think a really major conceptual puzzle in neuroscience. Why can we record from 100 neurons in a circuit containing millions of neurons? And do dimensionality detection and try to infer the state space dynamics of the circuit and claim that we've achieved success, right? We're doing dramatic undersampling recording 100 neurons out of millions.

How would the state space dynamics that we infer change if we recorded more neurons? And we can show that, essentially, it will not change. Because we've come up with a novel connection between the act of neural measurements and the act of random projections.

So we can show that the act of recording from 100 neurons in the brain is like the act of measuring 100 random linear combinations of all neurons in the relevant brain circuit. And then we can apply random projection theory to give us a predictive theory of experimental design that tells us, given the complexity of the task, how many neurons would you need to record to correctly recover the state-space dynamics of the circuit. And then in the Raymond Lab at Stanford, we've been looking at how enhancing synaptic plasticity can either enhance

or impair learning depending on experience.

So, for example, we think that synaptic plasticity underlies the very basis of our ability to learn and remember. So you might think that enhancing synaptic plasticity through various pharmacological or genetic modifications might enhance our ability to learn and remember. But previous results have been mixed.

When you perturb synaptic plasticity, sometimes you enhance learning, sometimes you impair learning. So I believe in the Raymond Lab, they're the first to show that in the same subject enhancing syntactic plasticity, for example, in the cerebellum can either enhance or impair learning depending on the history of prior experience. And we can show that in order to explain the behavioral learning curves, you need much more complex postsynaptic dynamics than people naturally assume.

We have to really promote our notion of what a synapse is from a single scalar, like a WIJ at a neural network to an entire dynamical system in its own right. So this relates to VOR learning. So this is sort of the low-level stuff that we've been doing.

I know that in this course you guys study higher level stuff. So I'm not going to talk about any of these. Each of them could be like a one hour talk. But I wanted to discuss some more high level stuff that we've been doing.

Oh, sorry. And sorry, the other sort of direction in our lab that we're looking at is actually the statistical mechanics of high dimensional data analysis. So this is, of course, very relevant in the age of the BRAIN Initiative and so on where we're developing very large scale data sets and we'd like to extract theories from this so-called big data.

So the entire edifice of classical statistics is predicated on the assumption that you have many, many data points and you have a small number of features, right? So then it's very easy to see patterns in your data. But what's actually happening, nowadays, is that we have a large number of data points and a large number of features-- so for example, where you can record from 100 neurons using electrophysiology maybe for only 100 trials of any given trial type in a monkey doing some task.

So the ratio of the amount of data to the number of features is something that's order one. So you know, the data sets are like three points in a three-dimensional space. That's the best we can visualize. So it's a significant challenge to do data analysis in this scenario. So it turns out

there's beautiful connections between machine learning and data analysis and the statistical physics of systems of quenched disorder.

So what I mean by that is in data analysis you want to learn statistical parameters by maximizing the log likelihood of the data given the parameters. In statistical physics, you often want to minimize. You know, this can be viewed as energy minimization.

And so there's beautiful connections between these. And so we work on that. So we've applied this to compressed sensing. We've applied this to the problem of what's the optimal inference procedure in a regime, in a high-dimensional regime.

We know that maximum likelihood is optimal in this regime. But something else is better in this regime. And we found what's better. It turns out to be a smooth maximum likelihood.

And, of course, we've applied this to a theory of neural dimensionality and measurement. So, you know, there's lots of beautiful interactions between physics, machine learning, and neuroscience. And, you know, this school is a lot about that.

If you're interested, actually, we wrote like a 70-page review on statistical mechanics of complex neural systems and high-dimensional data. We cover a whole bunch of things like spin glasses, the statistical mechanics of learning, random matrix theory, random dimensionality reduction, compressed sensing, and so on and so forth. It was our attempt to sort of put some systematic order on the diversity of topics viewed through the lens of statistical physics.

But what do I want to talk about today? And why did I decide to branch out in the direction that I'm going to tell you about? Well, I think there's lots of motivations for the alliance between theoretical neuroscience, theoretical machine learning that lead to opportunities for physics, and math, and so on.

So this is the question that should haunt all of us, right? The question is, what does it even mean to understand how the brain works or how a neural circuit works? OK? You know, that's an open question that we really have to come to terms with.

A more concrete version of this question might be, or a specification of this question might be, we will understand this when we understand how the connectivity and dynamics of a neural circuit give rise to behavior and also how neural activity and synaptic learning rules conspire to self-organize useful connectivity that subserves behavior. OK? So, you know, various BRAIN

Initiatives are promising to give us recording some large numbers of neurons and even give us the connectivity between those neurons.

Now, what have theorists done? Often what theorists in computational neuroscience do is they often develop theories of random networks that have no function. But what we would like to do is we'd like to understand engineered networks that have function.

So the field of machine learning has generated a plethora of learned neural networks that accomplish interesting functions. Yet we still don't have a meaningful understanding of how their connectivity, dynamics, the learning rule, the developmental experience-- OK. So basically, we can measure anything we want in these artificial neural networks, right?

We can measure all the connectivity between all the neurons. We know the dynamics of all the neurons. We know the learning rule. We know the entire developmental experience of the network, because we know the training data that it was exposed to.

Yet we still do not have a meaningful understanding of how they learn and how they work. And if we can't solve that problem, how are we ever going to understand the brain, right, in the form of this question? OK? So that was sort of what was motivating me to look into this.

So this is the outline of the talk. The original entry point was trying to understand category learning in neural networks. And then at the end of the day, we made actually several theoretical advances that led to advances in machine learning and applications to engineering.

So, for example, we found random weight initializations that make a network dynamically critical and allow very, very rapid training of deep neural networks. We were able to understand and exploit the geometry of high-dimensional error surfaces to, again, speed up learning, like training deep neural networks. And we were also able to exploit sort of recent work in non-equilibrium thermodynamics to learn complex probabilistic generative models.

So it's a diversity of topics, but I'll walk you through them. And, you know, you can relax, because almost everything I'm going to talk about is published. OK. So let's start with the motivation, a mathematical theory of semantic development. I think this speaks to some of the high level stuff that you guys think about.

This part could be called the misadventures of an applied physicists who found himself lost in the psychology department. So I just sort of showed up at Stanford. Jay's a great guy. I was

talking to him.

And I learned about his work. And I realized it didn't understand it. And this is my attempt to understand that work with Andrew and Jay.

OK. So what is semantic cognition? So human semantic cognition, a rough definition of this field, is that we have an ability to learn, recognize, comprehend, and produce inferences about properties of objects and events in the world, especially properties that are not present in your current perceptual stimulus. So, for example, I can ask you does a cat fur and do birds fly, and you can answer these questions correctly despite the fact that there's currently no cat or bird in the room, right?

So, you know, our ability to do this likely relies on our ability to form internal representations of categories in the external world and associate properties with those categories. Because we never see the same stimulus twice. So whenever we see a new stimulus or we try to recall information from our brain, we rapidly identify the relevant category that contains the information. And we use that categorical representation to guide future actions or give answers. So category formation is central to this, right?

So what are the kinds of psychophysical tasks that people use to probe semantic cognition? So this is a very rich field. Psychologists have been working on this all the time. So one example is looking time studies to ascertain whether or not an infant can distinguish between two categories of objects at what age.

So, for example, they'll show a sequence of objects from category one, say, horses. And the first time the infant sees a horse, the looking time will go up. And then it goes down over time. It gets bored.

Then you show a cow. And if the infant is old enough, the looking time will go up and then go back down. And from that, we infer that the infant can distinguish between horses and cows.

But if it's not old enough, the looking time will not go up. So as the infant gets older and older, it can make more and more fine scale discriminations between categories it turns out. So property verification tasks-- you can ask, can a canary move? Can it sing?

And certain questions are answered quickly. Certain questions are answered late, which speaks to certain properties being central and peripheral to certain categories. Category membership queries-- is a sparrow a bird, or is an ostrich a bird? Again, there's different

latencies. And that suggests that there are typical and atypical category members.

And also, very, very important to us is inductive generalization. We can both generalize familiar properties to novel objects-- for example, a blick has feathers. Does it fly? Does it sing?

And we can generalize novel properties to familiar objects. A bird has gene x. Does a crocodile have gene x? Does a dog have gene x?

You know, so people have measured these patterns of inductive generalizations. And there's various theories that try to explain all of this stuff. So Jay has been working on this stuff from a neural network perspective.

And he wrote a beautiful book called *Semantic Cognition* where he uses neural networks to explain a whole variety of phenomena especially, for example, the progressive differentiation of concepts. So let me just walk you through that. And so this was, you know, a first encounter with a deep neural network.

So they were doing deep neural networks before they became popular. And so what they were doing was they asked, can we model the development of, say, concepts in infants? And so what they did was they had a toy data set where they had a bunch of objects and each object had a whole bunch of properties.

So, for example, a canary can grow, move, fly, and sing, right? And so what they did was they exposed this deep neural network to training data of this form. You know, they had a whole bunch of features and questions and objects.

And they just exposed the network to training data, trained it using back propagation. And they looked at the internal representations in the network, especially their evolution over developmental time or training time, right? And this is what they found.

So initially, the network started with random weights. So there was no structure. OK. So what did they do here?

They looked at the distances between the internal representations in this space. And they did hierarchical clustering or multidimensional scaling. And they found these plots or these plots, right?

So what you see is, early in developmental time, the network first makes a coarse-grain discrimination between animals and plants, right? And then later, it makes finer scale discriminations. And then eventually when it's fully learned, it learns the hierarchical structure that's implicit in the training data.

OK. And this is a multidimensional scaling plot where initially the animals move away from the plants, and then, you know, fish move away from birds, and trees move away from flowers. And then finally, individual discriminations are learned.

So when I learned about this, I was at once excited and also mystified. Because this is sort of qualitatively behaving like the way that an infant behaves, yet it's a really stupid neural network with like five layers. Yet I don't understand how it's doing this, right?

So I wanted a theory of what's going on here, right? Oh and by the way, you know, there's lots of reasons to believe that semantic relationships are encoded in the brain using relatively simple metrics like Euclidean distance between neural representations for different objects. So, for example, this is a famous study which I'm sure you've seen.

What they showed was a whole bunch of objects to both monkeys and humans. And they clustered the objects or looked at similarity matrix of the objects measured using a Euclidean distance in neural electrophysiology space, so fine rates of neurons here and voxel activity patterns in the human. And they showed the same set of objects to monkey and human. And the matrices aligned, essentially.

So basically, the similarity structure of internal representations of both monkey and human is the same. So we tend to encode semantic information using the similarity representations. So this is the hierarchical clustering view. So this sort of seems to actually happen in real live animals and humans.

OK. There's actually something else that happens. It's that different properties are learned on different time scales. So, for example, the network can learn that canaries can move much more quickly than it learns that a canary is yellow.

So some properties are much easier to learn than others. And the properties that are easier to learn for the network are also easier to learn for the infant. OK. So these are the theoretical questions we'd like to answer.

What are the mathematical principles underlying the hierarchical self-organization of internal

representations in the network? You know, this is a complex system. So what are the relative roles of the various ingredients?

There's a non-linear input-output response. There's a learning rule, which is back propagation. There's the input statistics.

Is the network somehow reaching into complex input statistics in the training set, or can it really rely on just second order statistics? You know, what is a mathematical definition of something called category coherence? And how does it relate to the speed of category learning?

So what determines the speed at which we learn categories? Why are some properties learned more quickly than others? And how can we explain changing patterns of inductive generalization over these developmental timescales?

OK. So how do we get a theory? Well, it turns out if you look at the activations of this network as it's training over time, so these are sigmoidal units. And the activations don't really hit the saturating regime that much during training, because you start from small weights.

So we started with an audacious proposal that maybe even a linear neural network might exhibit this kind of learning dynamics. OK? Now, it's not at all obvious that it should, because it's a simple linear neural network. And this learning dynamics is highly non-linear, right?

But it turns out that even in a linear neural network, the dynamics of learning on synaptic weight space is non-linear. And so there might be a hope that it might work and we might be able to get a coherent theory. OK. So what we did was we analyzed just a simple linear neural network that looks like this that goes from input layer to hidden layer to output layer. So the composite map is linear.

OK. So we can write down dynamical equations and weight space for the learning dynamic. So this is the training data. And we can adjust the weights using back propagation. And these are the back propagation equations.

And if we work in a limit where the learning is slow relative to the time it takes to cycle through the data set, you can take a continuous time limit, and you essentially get a non-linear set of equations in weight space, right? And the equations are cubic in the weights, right? And that's because the error is quartic in the weights, right?

The error is the output minus w, w times the inputs squared. So the error is quartic in the weights. And so if you can differentiate the weights on the right-hand side, the gradient descent equations will be cubic in the weights.

But there is one simplification that happens. Because the network is linear, it's learning dynamics is sensitive only to the second order statistics of the data, right? So in particular-- the input-input covariance matrix and the input-output covariance matrix.

OK. So essentially, this network knows only about second order statistics. In our work here, the input statistics is white. So it's really only the input-output statistics that drives learning. OK? So this is a set of coupled non-linear differential equations.

They're, in general, hard to solve. But we found solutions to them. We can express the solutions in terms of the singular value decomposition of the input-output covariance matrix.

You know, any rectangular matrix has a unique singular value decomposition. In this context, we can think about the input-output covariance matrix as a matrix that maps input objects to feature attributes. And the singular vectors have an interpretation where these singular vectors essentially map objects into internal representations.

The singular values amplify them. And then the columns of you are sort of feature synthesizers. The columns are sort of modes in the output feature space.

OK. So this is the SVD. But the question is, how does this drive the learning dynamics? So what we did was we found exact solutions to the learning dynamics of this form where the product of the layer one to layer two weights and the layer two to layer three weights are of this form. Where, essentially, the system, what it's doing-- the composite system-- is building up the singular value decomposition of the input-output covariance matrix mode by mode.

And each mode alpha, associated with singular value alpha in the training data, is being learned in the sigmoidal fashion. OK? So at time zero, A is sort of small and random, you know, some initial condition A zero. But over time, as time training time goes to infinity, the A approaches the actual singular value in the input-output covariance matrix.

So basically, this is the learning dynamic. So nothing happens for a while. And then suddenly, the strongest singular mode defined by the largest single value gets learned.

And then later on, a smaller singular mode gets learned. And later on, an even smaller

singular mode gets learned. And the time it takes to learn each mode is governed by one over the singular value. So just intuitively, stronger statistical structure as quantified by singular value is learned first. That's the intuition.

Often time when we train neural networks, we see sort of these plateaus in performance where the network does nothing and then suddenly drops, plateaus and drops. And this actually shows that. You can see very, very sharp transitions in learning.

And you can actually show that the ratio of the transition period to the ignorance period can be arbitrarily small. Infants also seem to show these developmental transitions. OK. So, yeah, you can have arbitrarily sharp transitions in the system.

OK. So the take-home messages so far is the network learns different modes of covariation between input and output on time scale inversely proportional to the statistical strength of that covariation. And you can get these sudden transitions in learning. Now the question is, what does this have to do with the hierarchical differentiation of concepts? All right, that's what we'd like to understand first.

So now we've come up with a general theory of learning, the non-linear dynamics of learning in these deep circuits. Now we want to connect this back to hierarchical structure. So one of the things with Jay's work is that we're just working with toy data sets. And we didn't have any theoretical control over those toy data sets.

But we sort of understood implicitly that these toy data sets have hierarchical structure. So we need a generative model of data, a controlled mathematically well-defined generative model of data that encodes the notion of hierarchy. OK? So can we move beyond specific data sets to general principles of what happens when a neural network is exposed to hierarchical structure? That's what we'd like to answer.

So we consider a hierarchical generative model. And a classic hierarchical generative model is-- yeah, so essentially what we want to do is we want to connect the world of generative models to the world of neural networks. And, you know, that will connect the methods in computational neuroscience to this course eventually, right?

Yeah. So we have data generated by some generative model. We take that data, and we expose it to a neural network. And we'd like to understand how the dynamics of learning depends on the parameters of the generative model.

OK. So a natural generative model for defining hierarchical structure is a branching diffusion process that essentially mimics the process of evolution where properties diffuse down a tree and instantiate themselves across a set of items. So what do I mean by that? OK. OK. So basically imagine, for example, that your items are at the leaves of a tree, right?

And you can imagine that this is a process of evolution where there is some ancestral state maybe for one property. So we do one property at a time. And the properties are independent of each other.

This might be an ancestral state like can move, right? And then each time this property diffuses down the tree, there's a probability of flipping, OK? So maybe in this lineage which might correspond to animals, this doesn't flip, right? And in these lineages corresponding to plants, it does flip.

So these things cannot move. And these things can move. And then maybe it doesn't flip. So all of these things inherit the property of moving.

So these are the animals. And these things cannot move. So these are the plants.

And then we do that for every single property independently. And we generate a set of feature vectors. So that's our generative model.

So what are the statistical properties of the generative model? So essentially, because we know that we're analyzing these deeper linear networks and we know that the learning dynamics of such networks is driven only by the input-output covariance matrix, to understand the learning dynamics we just have to compute the singular values and singular vectors of hierarchically structured data generated in this fashion. And it's actually quite-- I mean, we did it.

So here's what happened. So imagine a nice symmetric tree like this. So these are objects. If we look at the similarity structure of objects measured by dot product in the feature space generated by the features under this branching diffusion process, we get this nice blocks within blocks similarity structure where all the items on this branch-- you know this item and this item-- are slightly similar. This item and this item are even more similar. And, of course, each item is most similar to itself.

So you have this hierarchical hierarchy of clusters that naturally arise because of this

branching diffusion process. So what are the singular values and singular vectors of the associated input-output covariance matrix? Well, they turn out these are one set of singular vectors, the so-called object analyzers, which are functions across objects. There's another set of singular vectors that are functions across features, which I'm not showing you.

But there's, of course, the duality, right? So that you get pairs of singular vectors for each single value. OK. So what's the singular vector associated with the largest singular value? Well, it's a uniform mode that's constant across all the objects.

But the most interesting one, the next largest one, is the most lowest frequency function, essentially. It's constant along all the ancestors of this branch and a different constant along all the ancestors of this branch. So this singular vector, essentially, makes the most coarse grain discrimination in this hierarchically structured data set.

The next set of singular vectors-- there's a pair of them-- discriminate between this set of objects and this set of objects and don't know about these ones. And the next one discriminates between this set of objects and this set of objects. And then as you go down to the smaller singular values, you get individual object discriminations, right?

So this is how the hierarchical structure is reflected in the second order statistics of the data. And these are the singular values. So this is the theory for the singular values in a tree that has five levels of hierarchy. And you can see that the singular values decay with the hierarchy level of the singular vectors.

OK. So there's a general theory for this in which singular vectors are associated with levels of the tree. OK? So now you can see the end of the story. If you put the two together, you automatically get the results that we were trying to explain, right?

So essentially, the general theory of learning says that the network learns input-output modes on a time scale given by 1 over the singular value. When the data is hierarchically structured, singular values of broader hierarchical distinctions are larger than singular values of finer distinctions. And the input-output modes correspond exactly to hierarchical distributions of the tree.

So that essentially says the network must learn broad scale discriminations before it can learn fine scale discriminations. So then actually what we did was we just analytically worked out that the dynamics of learning for hierarchically structured data. And we computed the

multidimensional scaling.

And this was theory. We never did a single simulation to get this plot. We generated a branching diffusion process that was essentially this one. And we just labeled these nodes arbitrarily with these labels.

And this is what we get, right? So we see the multidimensional scaling plot that we sort of see here. And essentially, just to compare, this is what was done with a toy data set over which we had no theoretical control and a non-linear neural network over which we had no theoretical control. And this is a well-defined mathematical generative model under a linear neural network. And we see that we qualitatively explain the results.

So this is the difference between simulation and theory, right? Now we have a conceptual understanding of effectively what was going on in this circuit. And now, it's no longer a mystery. So now I think I understand what Jay and collaborators were doing.

It would be lovely if, for all of the stuff that's going on in this course, we could obtain such a deep rigorous understanding. It's much more challenging. But it's a goal worthy of pursuit I think.

OK. So conclusions-- progressive differentiation of hierarchical structure is a general feature of learning in deep neural networks. It cannot be any other way. OK? Interestingly enough, deep, but not shallow, networks exhibit such stage-like transitions during learning. So if you just do no hidden layers, you don't get this, actually. You need a hidden layer to do this.

And somewhat surprisingly, it turns out that even only the second order statistics of semantic properties provide powerful statistical signals that are sufficient to drive this non-linear learning dynamics, right? You don't need to look at the higher order statistics of the data to get this dynamic. Second order statistics suffice, which was not obvious before we started.

OK. So in ongoing work, we can explain a whole bunch of things, like illusory correlations early in learning. So, for example, infants, they don't even know that, for example, pine trees don't have leaves. Then at an intermediate point, they think that pine trees have leaves. And then at a later point, they correctly know that pine trees don't have leaves, right?

So we can explain these non-monotonic learning curves. We can explain these familiarity and typicality effects. We can explain inductive property judgments analytically. We're looking at basic level effects. We have a theory of category coherence, and so on.

But in the interest of moving forward, I wanted to give short shrift to this stuff. And essentially, we can answer why are some properties learned faster? Basically, properties that are low frequency properties on the leaves of the tree get learned faster. Properties whose inner product with the singular vector as a larger singular value get learned faster. That's the story.

Why are some items more typical? We have a theory for that. How is inductive generalization achieved by neural networks? We have a theory for that and so on.

And, you know, what is a useful mathematical definition of category coherence? So, for example, you know, there are some things that are just intuitively called incoherent categories. "The set of all things are blue" is a very incoherent category. In fact, it's so incoherent we don't have a name for such a category.

"The set of all things that are dogs" seems to be a very coherent category. And it's so coherent that we have a well-known name for it. The name's quite short actually, too. Actually, I wonder if there's a theory where shorter words correspond to more coherent categories and that's like an informative or efficient representation of category structure.

But anyways, we have a natural definition of coherent category that's precise enough to prove a theorem that coherent categories are learned faster. And actually, this also relates to the size of the categories. So frequency effects show up. Anyways, so there's a lot of stuff there. But that was sort of the entry point.

So now, what about a theory of learning in much deeper networks that have many, many layers? OK? So, again, I'm going to make a long story short, because it's all published. So you can read all the details. But I wanted to give you the spirit or the essence or the intuition behind the work.

OK. So the questions we'd like to answer are, how does training time scale with depth? How should learning rate scale with depth? How do different weight initializations impact learning speed? And what we'll do is once we understand these theoretically, we'll find certain weight initializations that correspond to critical dynamics, which I'll define, can aid deep learning and generalization.

So the basic idea is in a very, very deep neural network, right, you have a vanishing, exploding, or gradient problem. And that's one of the issues that makes deep neural network

learning hard. So if you're going to back propagate the error through multiple layers, the back propagation operation is a product of Jacobians from layer to layer. And that product of Jacobians is fundamentally a linear mapping, right?

So if the singular values associated with that linear mapping-- so essentially, if the singular values of the Jacobian in each layer are large, bigger than one, the product of such matrices will lead to a product matrix that has singular values that grow exponentially with depth. Similarly, if the single values are less than one, they'll decay with depth, right? So that's a vanishing gradient in the latter case and an exploding gradient in the former case.

That seems to be one of the major impediments to understanding deep learning. So what people often did was they tried to scale the matrices to avoid this question, right? So what they often do is they initialize the weights randomly so that W is a random matrix where the elements W, I, J, are IID and Gaussian with a scale factor scaled precisely so that the largest eigenvalue of the Jacobian or the back propagation operator is one. OK?

So that's like scaling the system so that if you place a random error vector here, the desired output minus the actual output, and back propagate it through a random network, a error vector will preserve its norm as it's back propagated across. And this is the famous sort of Glorot and Bengio initialization. And it works pretty well for depth four or five or whatever, right?

OK. So we would like a theory of that for the learning dynamics of that. And as I said, there's no hope for a complete theory at the moment with arbitrary non-linearities. OK. So what we're going to do is we're to analyze the learning dynamics. Just-- we'll get rid of the non-linearities, right?

So, again, it might seem like we're throwing the baby out with the bathwater, but we're actually going to learn something that helps us to train non-linear networks. OK. So the basic idea then is that we have a network which is linear. So y, the output, is a product of weights, right? OK. So then the back propagation's, again, just a product of matrices, right?

The gradient dynamics is non-linear and coupled and non-convex. And actually even in this linear network, you see plateaus and sudden transitions, right? And actually, interestingly enough, even in this very deep linear network, you see faster conversions from pre-trained initial conditions, right?

So basically, if you start from random Gaussian initial conditions, you get slow learning for a while and then sudden learning here, relatively sudden learning here. Whereas, if you're pre-train the network using greedy unsupervised learning, so this is the time it takes to pre-train, you get sudden learning and a drop here. So remember, if you go back to the original Hinton paper, this was the phenomenon that started deep learning.

Greedy unsupervised pre-training allows you to rapidly train very, very deep neural networks. So the very empirical phenomenon that led to the genesis of deep learning was present already in deep linear neural networks, right? So deep linear neural networks, in terms of their expressive power, are crappy. Because the composition of linear operations is linear.

They're not a good model for deep non-linear networks in terms of input-output mappings. But they're a surprisingly good model, theoretical toy model, for modeling the dynamics of learning in non-linear networks. OK? Because very important phenomena also arise in the deep linear networks. And we're focusing on learning dynamics here.

OK. So we can build intuitions for the non-linear case by analyzing the linear case. OK? So we went through the three layer dynamics already. What about the multiple layer dynamics?

So, again, the Jacobian can back propagate or explode, right? OK. So, again, I'm going to make a long story short. But what we find is that if you take-- OK, I'll tell you the final result. What we find is that we find a class of weight initializations that allow learning time to remain constant as the depth of the network goes to infinity.

Now, I'm measuring learning time in units of learning epochs, right? So, obviously, to train a deep neural network, very, very deep neural network, it just takes longer to compute each gradient, right? So in terms of real time, of course, the time will scale with the depth of the network.

But you might imagine in terms of number of gradient evaluations, as the network gets deeper and deeper, it might take longer and longer to train it. And we show a class of initial conditions for which that's not true. As the network gets deeper and deeper, the number of gradient evaluations you need to train the network can remain constant even as the depth goes to infinity even in a non-linear network. OK. So let me give you intuition for why.

OK. So, for example, in the classical initialization, this Glorot and Bengio initialization doesn't have that. But our initialization does. So basically what we did was-- we'll start off with a linear

networks.

We trained deep linear networks on MNIST. And we scaled that depth like this, right? And we started with random Gaussian initial conditions and then ran back propagation, but scaled random Gaussian initializations.

And we found that the training time, as you might expect, grew with depth. This is training time measured in number of learning epochs or number of gradient evaluations. But here what we did was we initialized the weights using random orthogonal weights, right?

And then we found that the learning time didn't grow with depth. And also, if you pre-train it, it doesn't grow with depth. OK. So there's a dramatically different scaling and learning time between random Gaussian initialization and random orthogonal initialization.

Why? OK? And the answer is the following. Let's think about the back propagation operator. Let's say you want to back propagate errors from the output to the input. So the back propagation operator in a linear network is just the product of weights throughout the entire network. OK?

So if you do a random Gaussian weight initialization here, then this is a product of random Gaussian matrices. So to understand the statistical properties of back propagation, you need to understand the statistical properties of the singular value spectrum of random Gaussian matrices. There isn't really a general theory for that, but we can look at it numerically and get intuition for it.

So the basic idea is if you have one random Gaussian matrix, the singular values of W are the eigenvalues of W transpose W. That's a famous distribution called the Marchenko-Pastur distribution. And you know, they vary in a range that's order one. OK?

So if you back propagate through one layer, you're fine. You don't get vanishing exponent gradients. OK. But if you look at the singular values of a product of five random Gaussian matrices, the singular value spectrum gets very distorted. You've got a large number of similar values that are close to zero and a long tail that, you know, extends up to four.

OK. But if you do 100 layers, you get a very, very large number of singular values that are close to zero and very much longer tail. OK? Now, this is a product of random Gaussian matrices. So if you feed a random vector into this, on average, it's norm will be preserved. The vector's length will not change.

But we know that preserving the length of a vector is not the same as preserving angles between all pairs of vectors. OK. So actually, the way that this product of random Gaussian matrices preserves the norm of the gradient is it does it in a very anisotropic way. It takes an error vector at the output, and it projects it into a low dimensional space corresponding to the singular values that are large. And then it amplifies it in that space.

So the length is preserved, but all error vectors get projected onto a low-dimensional space and amplified. So a lot of error information is lost in a product of random Gaussian matrices. OK. So that's why the Glorot and Bengio initial conditions work well up to five or six or seven, but they don't work well up to depth, say, 100 or in recurrent neural networks as well. OK?

So what can we do? Well, a simple thing we can do is we can replace the matrices, these random matrices, with orthogonal matrices. OK? So we know that all the singular values of an orthogonal matrix are one, every single one. And the product of orthogonal matrices is orthogonal.

So therefore, the back propagation operator has all of its singular values equal to one. And there's generalizations of orthogonal matrices to rectangular versions when the layers don't have the same number of neurons in each layer. OK? So this is fantastic.

So this works really well for linear networks. OK. But how does this generalize to non-linear networks? Because then you have a product of Jacobians, right? So what happens here?

OK. So what is the product of Jacobian? OK. So if we imagine how either errors back propagate to the front or how input perturbations back propagate to the end, it's the same thing. So it's easier to think about forward propagation.

Imagine that you have an input and you perturb it slightly. How does the perturbation grow or decay? Well, what happens is there's a linear expansion or contraction due to W.

And then this nominator is usually compressive. So there's a non-linear compression due the diagonal Jacobian passing through the point y's non-linearity. And then, again, linear modification and non-linear compression, linear modification, and non-linear compression.

OK? So what we could do is just simply choose these again to be random orthogonal matrices. And then what happens is the growth or decay of perturbations-- and we scale the random orthogonal matrices by a scale factor to combat the non-linear compression. And then the

dynamics of perturbations is like this.

You rotate, linearly scale, non-linearly compress, rotate, linearly scale, non-linearly compress, and so on. That's essentially the type of dynamics that occurs in dynamically critical systems that are close to the edge of chaos. You get this alternating phase space expansion and compression that's in different dimensions at different times. OK?

So now you can just compute numerically under that initialization. How does the singular value spectrum of the product of Jacobian scale? And it scales beautifully.

So this is the scale factor for the type of non-linearity that we use, the hyperbolic tangent non-linearity. The optimal scale factor in front of the random orthogonal matrix is one. And you see when you choose that-- this was 100 layers, I believe-- even for 100 layers, that end to end Jacobian and from the input to output has a singular value spectrum that remains within the range of order one.

If g is even slightly less than one, the singular values exponentially vanish with depth. If g is larger than one, the singular values grow, but actually not as quickly as you'd think. So this is the critically dynamical regime that at least preserves not only the norm of back propagated gradients, but all angles between pairs of gradients, right?

So it's an isotropic preservation of error information from the end of the network all the way to the beginning. OK? So does it work? And it works better than other initializations even in non-linear networks.

So we trained 30 layer non-linear networks. And the initialization works better. And so also, interestingly enough, at this critical factor you also achieve better generalization error.

And we don't have a good theory for that actually. The test error and the training error, of course, goes down. OK. So that's an interesting situation where a theory of linear networks led to a practical training advantage from non-linear networks.

OK. So here's another question that we had. OK? There's a whole world of convex optimization. We want our machine learning algorithms to correspond to convex optimization, so we can find the global minimum. And there are no local minima to impede us from finding the global minimum, right?

That's conventional wisdom. Yet the deep neural network people ignore this conventional

wisdom and train very, very deep neural networks and don't worry about the potential impediments to the local minima. They seem to find pretty good solutions why. OK?

Is the intuition that local minima are really an impediment to non-linear non-convex optimization in high dimensional spaces really true? OK? And you might think that it's not true for the following intuitive reason, right? So, again, it's often thought that local minima, at some high level of error and training error, stand as a major impediment to non-convex optimization.

And, you know, this is an example-- a two-dimensional caricature of a protein folding energy landscape. And it's very rough, so there's many, many local minima. And the global minima might be hard to find.

And that's true. If you sort of draw random generic surfaces over low dimensions, those random surfaces will have many local minima. But, of course, our intuition about geometry derived from our experience with a low-dimensional world is woefully inadequate for thinking about geometry in high-dimensional spaces.

So it turns out that random non-convex error functions over high-dimensional spaces, local minima are sort of exponentially rare in the dimensionality relative to saddle points. Just intuitively, imagine you have an error function over 1,000 variables, say 1,000 synaptic weights in a deep network. That's a small, deep network.

But anyways, let's say there's a point at which the gradient and weight space vanishes. So now there's 1,000 directions in weight space you could move away from that extreme. What are the chances that every single direction you move has positive curvature, right?

If it's a fairly generic landscape, the answer is exponentially small in the dimensionality. Some directions will have negative curvature. Some directions will have positive curvature, unless your critical point is already at the bottom. In that case, most directions will have positive curvature. Or unless your critical point is at the top higher, then most directions will have negative curvature, right?

So statistical physicists have made this intuition very precise for random landscapes. And they've developed a theory for it. So this is a paper in *Physical Review Letters* by Bray and Dean.

So what they did was they imagined just a random Gaussian error landscape. So what they

did was they looked at an error landscape that's a continuous function over n dimensions, but there is correlations. It's correlated over some length scale.

So it's a single draw from a random Gaussian process where the kernel of the Gaussian process is falling off with some length scale. So the error at 0.1 is correlated with the error at 0.2 over some length scale. And that correlation falls off smoothly. So it's a random smooth landscape. OK. So the correlations are local, essentially.

And then what they did was they asked the following question. Let x be a critical point, a point where the gradient vanishes. OK? We can plot every single critical point in a two-dimensional feature space.

What is that feature space? Well, the horizontal axis is the error level of the critical point. At how high on the error axis does this critical point sit?

And then this f is the fraction of negative eigenvalues of the Hessian at that critical point. So it's the fraction of directions that curve downwards. OK. So now a priori, critical points could potentially set anywhere in this two-dimensional feature space, right?

It turns out they don't. They concentrate on a monotonically increasing curve that looks like this. So the higher the error level of the critical point, the more the negative curvature directions you have. OK?

And to be an order one distance away from this curve, the probability of that happening is exponentially small in the dimensionality of the problem. OK? Now, what does that mean? It automatically implies that there are no local minima at high error, or at least they're exponentially rare relative to saddle points of a given index. OK?

So basically, you typically never encounter local minima at height error, right? That would be stuff that sits here. And there's nothing here. OK?

Second, if you are a local minimum, which means on this axis you're at the bottom, then your error level must be very, very close to the global minimum. OK? So if you get stuck in a local minimum, you're already close in error to the global minimum.

AUDIENCE: Can you repeat this last element?

SURYA GANGULI: Yeah. So if you're a local minimum, your error level will be close to the error level the global

minimum.

AUDIENCE: Why?

SURYA GANGULI: Because what does it mean to be a local minimum? It means that f is zero. The fraction of negative curvature eigenvalues of the Hessian is zero.

And this is the distribution of error levels of such critical points. They're strongly peaked at this value, which is the value of the global minimum. Essentially, there's nothing out here. OK?

All right, now in physics there is this well-known principle called universality. There are certain questions whose answers don't depend on the details. For example, certain critical exponents in the liquid-gas phase transition are exactly the same as critical exponents in the ferromagnetic phase transition.

Because the symmetry and dimensionality of the order parameter density in the case of liquid and magnetization in the case of ferromagnets are the same. So there's certain questions whose answers don't depend on the detail. They only depend on the symmetry and dimensionality of the problem.

So one might think that this qualitative prediction is true in just generic high-dimensional landscapes. Now, the computer scientists would say, no, no, no, no, no, no. Your random landscapes are a horrible model for our error landscapes of deep neural networks trained on MNIST and CIFAR-10 and so on and so forth.

You're completely irrelevant to us, because we're doing something special. We're not doing something random. We have a lot of structure.

OK. The physicists might counter, well, you know, you just have a high-dimensional problem. The basic intuition that in high dimensions it's very hard to have all directions curve up at a critical point high error should also hold true in your problem. OK? But, of course, we'll never get anywhere if we stop there, right?

We have to move over into your land which is also my land and just simulate the system. So oftentimes, you know, biologists and computer scientists don't believe a theory until they see the simulation. So what we'll do is we'll search for critical points in the error landscape of deep neural networks.

And that's what we did. So what we did was we used Newton's method to find critical points. So it turns out that Newton's method is attracted to saddles, right?

So Newton's method will descend in the positive curvature direction. But it will ascend in the negative curvature direction. Because Newton's method is gradient descent multiplied by the inverse of the Hessian.

So if the Hessian has a negative eigenvalue, you take a negative gradient and multiply it by the negative eigenvalue, and you turn back around and you go uphill. So Newton's method uncorrected is attracted to saddle points. OK? So what we did was we looked at the error landscape of deep neural networks trained on MNIST and CIFAR-10, and we just plotted the prediction of random landscape theory, right?

And what we found was exactly qualitatively their prediction. We took each critical point and plot it in this two-dimensional feature space. And we found that the critical points concentrated on a monotonically increase in curve which, again, shows that there are no local minima at high error.

And if your a local minimum, your error is close to at least the lowest error minimum that we found. We can't guarantee that the lowest error minimum we found is the global minimum. But qualitatively, this structure holds. OK?

Now, the issue is what can we do about it. So what this is telling us-- that even in these problems of practical interest, saddle points might stand as the major impediment to optimization, right? Because saddle points can trap you.

You know, you might go down here. And then there might be a very slowly curving negative curvature direction that might take you a while to escape. In fact, in the learning dynamics that I showed in these transitions in learning hierarchical structure, the thing controlling the transitions was the existence of saddle points in weight space of the linear neural network.

And so the part of no learning corresponded to sort of falling down this direction slowly. And then the rapid learning corresponded-- eventually coming out this way. OK. So how do we do that?

Well, what we can do is we can do a simple modification to Newton's method, which instead of dividing by the Hessian, we divide by the absolute value of the Hessian. And, again, I should say that this was done in collaboration with Yoshua Bengio's lab. And a set a fantastic

graduate students in Yoshua Bengio's lab did all of this work on the training and testing of these predictions.

OK. So what we suggested was, you know, the offending thing is dividing by negative eigenvalues. So just take the absolute value of the Hessian, which by definition I mean take the Hessian, compute its eigenvalues, and replace each negative one with its absolute value. OK? So that will obviously get repelled by saddles, all right?

And that actually works really, really well. And there's a way to derive this algorithm in a way that makes sense, even far from saddles by minimizing a linear approximation to f within a trust region in which the linear and quadratic approximations agree. OK? So let me just show you first that it works.

So this is the most dramatic plot. So basically what we did was we did stochastic gradient descent for a while. And then it seemed like the error as a function of training time plateaued both for a deep auto encoder and a recurrent neural network problem.

So when the error as a function of training time plateaus, that's sort of interpreted as the fact that you're stuck in a local minimum, right? But actually, when we switched to this, what we call, the saddle-free Newton method, the error suddenly drops again. So this was an illusory signature of a local minimum.

It was actually a saddle point with probably a very shallow negative curvature direction that was hard to escape. And when we switched to our algorithm, we could escape it. And, you know, what these curves show is that we do do better in the final training error as while.

So now, how do we train deep neural networks with thousands of layers? And actually, how do we model complex probability distributions? So we want to sample from very, very complex probability distributions and do complex distributional learning, right?

So this was done by a fantastic post-doc of mine, Jascha Sohl-Dickstein. So we were going to Berkeley for this non-equilibrium statistical mechanics meetings and things like that. And there's been lots of advances in non-equilibrium statistical mechanics where you can show that, roughly speaking, the second law of thermodynamics which says that things get more and more disordered with time can be transiently violated in small systems or short periods of time so you can spontaneously generate order.

OK. So I'll just go through this, again, very quickly. So here's the basic idea. Let's say you have a complicated probability distribution.

Let's just destroy it. Let's feed that probability distribution through diffusion to turn it into a simple distribution, maybe an isotropic Gaussian. And we keep a record of that destruction of structure.

And then we try to reverse time in that process by using deep neural networks to reverse time and then essentially create structure from noise. And then you have a very, very simple way to sample from complex distributions if you can train the neural network, which is you just sample noise. And you feed it through a deterministic neural network. And that constitutes a sample from a complex distribution.

And so this was inspired by recent results in non-equilibrium stat mech. So the basic idea, again, is let's imagine that you have a very complex distribution corresponding to this density of dye. You diffuse for a while. It becomes a simpler distribution. Eventually, they become uniform.

You keep a [AUDIO OUT] Now, if you reverse process of diffusion, you'll never go from this back to this. But if you reverse process a neural network trained to do it, you might be able to do it.

So that's the basic idea. So that's what we did. And I'll just show you some nice movies to show that it works.

This is the classical toy model. We'll go to more complex models. This is a sample distribution in two-dimensional space. And so what we do is we just systematically have the points diffuse under Gaussian diffusion with a restoring force to the origin. So the stationary distribution of that destructive process is an isotropic Gaussian.

OK? And that's what happened. So that's our training data. The entire movie is the training data. OK? Then what we do is we train a neural network to reverse time in that movie.

So it's a neural network with many, many layers-- hundreds and hundreds of layers, right? So classically training a network with hundreds of layers, you have the credit assignment problem. Because you don't know what the intermediate neurons are supposed to do.

You can circumvent the credit assignment problem, because each layer going up to the next

layer just has to go from time t to time t minus 1 in the training data. So you have targets for all the intermediate layers. Therefore, you've circumvented the credit assignment problem.

OK? So it's relatively easy to train such networks. And so once you have such a network, what should you be able to do? You should be able to feed that neural network an isotropic Gaussian, and then have that Gaussian be turned into the data distribution.

So that's what happens. This on the right is a different Gaussian. And we just feed it through the trained deterministic neural network. And out pops the structure.

It's not perfect. There are some data points that are over here. But this is roughly the distribution that it learned, which is similar to what it was trained on. OK?

So now we can look at slightly more complicated distributions. OK. So that's that. So now we can train it on a toy model of natural images, right? So a classic toy model of natural images is the dead leaves model where the sampling process is you just throw down circles of different radii.

So you get a complex model of natural images that has long range edges, occlusion, coherence over long length scales, and so on and so forth. So we can train the neural network on such distributions. We train it in a convolutional fashion by working on local image patches. And we convolve, so information will propagate over long ranges.

And so, again, we take these natural images and turn them into noise, keep a record of that movie, and then reverse the flow of time using a deep neural network. OK. So once we train that, we should be able to turn noise into the networks best guess as to what a dead leaves model would look like. So this is what happens.

It's taking noise, and it turns it into a gas. OK. So it's not a perfect model. But it turns out log probability of dead leaves under this generative model that consists of 1,000-layer deep neural network, that's higher than any other model so far.

So this is currently state of the art. OK. And as you can see, it gets long-range coherence and sharp edges. And moreover, it gets long-range coherence in the orientation of [AUDIO OUT] often hard to do in generative models of natural images.

OK. Now, we can actually do something somewhat practical with this is we can sample from the conditional. So then we also trained it on textures. OK? So, for example, textures of bark,

right? And we can also sample from the conditional distribution.

So what we can do is we can clamp the pixels outside of a certain range, replace the interior with [AUDIO OUT] make it blank. And because the network operates in a convolutional fashion, information from the boundary should propagate into the interior and fill it in, right? So if we look at that, so that's white noise.

And the network is filling it in. And it fills in the best guess image. OK. And so, again, it's not identical to the original image, but it does get long-range edge structure, coherence in the orientation of the edge, and smooth structure as well.

And, again, this is like 1,000-layer neural network. Now, there's some lessons here. OK? Oftentimes when we model complex data distributions, what we try to do is we try to create a stochastic process whose stationery distribution is the complex distribution, right?

Now, if your distribution has multiple modes, you're going to run into a mixing problem, because it can take a stochastic process a long time to jump over energy barriers that separate the multiple modes. So you always have a mixing problem. And oftentimes when you train probabilistic models, you have to sample and then the samples to train the model.

So that makes training take a long time. So what we're also doing in addition to circumventing the credit assignment problem and training very deep neural networks, we're circumventing the mixing problem in training the generative model. Because we're not trying to model the data distribution as a stationary distribution of a stochastic process. That would have to run for a very long time to get to the stationary distribution.

We're demanding that during training the process get to the data distribution in a finite amount of time, right? So because during training we demand that we get to the data distribution a finite amount of time, we're circumventing the mixing problem during training. And that's the idea.

That's [AUDIO OUT] an idea. There's lots of results now that show that you can attain information about stationary equilibrium distributions from non-equilibrium trajectories. OK. So now, I'm done.

So let's see. OK. So there's that. OK. So, again, you can read about all of this stuff in this set of papers. Again, I'd like to thank my funding and just the key players.

So Andrew Saxe, you know, did the work with me on non-linear learning dynamics and learning hierarchical category structure. Jascha Sohl-Dickstein did the work on deep learning using non-equilibrium thermodynamics. And the work on saddle points was a nice collaboration with Yoshua Bengio's lab.

And again-- fantastic graduate students in Yoshua Bengio's lab. OK. So I think there's a lot more to do in terms of unifying neuroscience, machine learning, physics, math, statistics, all of that stuff. It'll keep us busy for the next century.