[SQUEAKING]

[RUSTLING]

[CLICKING]

SARA BEERY: I hope everyone is having a good fall. I don't know if any of all of you noticed, but it started to be really cold today for the first time. And I went outside, and I was like, it begins. So today, we're going to be continuing our series digging into representation learning. And specifically, I'm going to be introducing the concept of similarity-based representation learning.

So to start with a high-level roadmap, today, we're going to be talking about why we want to think about learning representations, why learning representations is a good thing, try to get a sense of what makes a good representation both qualitatively but also exploring some dimensions of quantitatively what might make a good representation. We'll introduce the concept of metric learning.

And then we'll dig into contrastive representation learning, which is essentially different creative ways to self-supervise the idea of building a metric space or using metric learning. And we'll talk about what contrastive learning does, and then some of the different potential models that are using contrastive learning. This is a pretty important point throughout this entire course because it turns out that many of our modern architectures are built on top of some form, or some component of contrastive learning and contrastive representations.

So why learn representations? What are some reasons why we might want to learn a representation of data? Yeah, over there.

AUDIENCE: Use it for any many tasks.

SARA BEERY: We can use it for many tasks, the idea that maybe some representation of data is broadly useful. Anything else? Yeah?

AUDIENCE: Maybe more compact.

SARA BEERY: It might be more compact, right. It might be more efficient to work with some representation of data rather than the original data itself, though, of course, that's a choice. You can actually build a representation that's larger than your data input. Anyone else? What else makes it-- yeah, in the back?

AUDIENCE: Generalization.

SARA BEERY: To improve generalization, right. We might want representations that actually are capturing some semantic meaning or some important components of the things we're trying to identify that will generalize well. So I mean, yes, to improve generalization-- I'm going to trust you that you weren't just reading the slides, but power to you if you were-- to do more learning things, like transfer learning. So often, we talk about this is that first point about working for many different tasks. We build some representation that then is more easily finetuned in a way that actually gives us something that's useful, maybe requiring less training data in that finetuning step.

And then another version is actually maybe to exploit something like geometric similarity for new data or queries. So what does geometric similarity mean? Well, one thing might have we seen the face of this person before, or is it new? There's actually something physical and structural about our face that should be similar, regardless of maybe the lighting, the pose, the orientation. Or another version of something like geometric similarity might be in retrieval. Can you build representations that let you find items similar to a given query? So if I say, hey, here's something I just found, have we ever seen something similar to this before?

Another example might be trying to do something improve clustering with side information, something about what makes pairs similar or dissimilar. And then, possibly, we want to do dimensionality reduction-- exactly. Can we actually make our-- can we store this information in a much more compact way? And we can often do dimensionality reduction in ways that are fundamentally unsupervised.

So what are some of the things we might expect from such representations? What makes a representation good? Yeah?

AUDIENCE: It should map objects in your higher dimensional space that are similar in semantic representation to the very close point in your embedding space.

SARA BEERY: Yeah, so the idea that objects that are semantically similar, we actually want them to be really close together in that representation space. We want them to be close together. Why would we want them to be close together? What does that do? Yeah?

AUDIENCE: Helps us reject disturbances and noise, potentially.

SARA BEERY: Yeah, it might help us be robust to things like disturbances and noise if you move a little bit in any direction, if you're not moving of far away from things that same semantic category. Why else might we want things that are similar to be close together? Yeah, sure?

AUDIENCE: [INAUDIBLE] the answer, but one reason might be that if you think of a downstream function which takes the embeddings, but then logically, if I'm trying to compute some feature of two inputs which are very similar, there's a good chance that I want f of x and f of x plus epsilon to have very similar y values.

SARA BEERY: Yeah, so what you said was basically we might want the downstream mapping of this embedding space to some maybe categorization to be kind of robust, to change, and to map things easily from one space to another. Actually, there's a different way to say that, which is basically it's useful if the space is linearly separable. It's a very easy function, if you want to learn categorization, if different objects that are the same are close together, and objects that are different are not close together because then you just draw lines between things. It's a very easy categorization problem. So a well-separated space is also potentially quite useful.

So Goodfellow, back in 2016, said that, "generally speaking, a good representation is one that makes a subsequent learning task easier." That's exactly what you were trying to get at. And so what could that mean? What is a good representation? So one framing of this is explicitly what we were just trying to capture. A good representation is both compact, which means it's not too big. We don't want to have more capacity than we actually need. This is getting at that efficiency angle. You can take in information theoretic perspective here and think about we want to keep the minimal amount of information, but we also need it to be what we call explanatory.

So that means that it's somehow sufficient to capture the dimensions that matter in our original data. And the really interesting part here is that of course this is going to be contextually related to the set of downstream tasks we might be interested in because something that is sufficient to be explanatory for one possible thing might be insufficient for another. And so a good representation, ideally, would be something that is kind of compact and explanatory in a way that makes it generalize to many possible downstream tasks.

So back in 2020, there was this work that tried to pose specifically the question of not what makes a good categorizer, but actually, can we build complexity measures that accurately predict the generalization of models? Can you build a complexity measure over a representation that will predict how well that representation will generalize?

And it turned out that a couple of winning strategies for this specifically looked at both the geometry of the representation, so that consistency and separation, how tight together, how close together were things that were the same, how far apart were things that were different, but also robustness to perturbations. So if you move a little bit in any dimension or any direction, do you actually change something fundamental about the representation? And this really maps quite well, I think, to some of our own intuition about what might make something easy to learn.

And if you look at this specifically on something like CIFAR, CIFAR-10, which is a very low dimensional image categorization problem where all of the images are very, very low resolution and small, and the categories are simple. If you train a classifier on CIFAR-10 with the correct labels, and then you look at a t-SNE visualization of the representations on the training data-- so this is the data that we've actually already trained the models on-- what you see is it learns to build a very consistent and separable training space.

The categories here are the same color. And so you can see the things that are the same color are very close together and they're very well-separated from everything else. And, interestingly, they're somewhat uniformly distributed in this space. And we'll actually talk later in the lecture about this other dimension of distribution throughout the space.

Now, if you take the same model and you randomly generate the labels on that training data, and then you train a model to recognize what are now basically random noise, you still will get clusters because the model can learn to memorize. But what you see is that those clusters are much less concise. They're much less compact. The colors that are the same color are more spread out in the space. And they are much less separable.

So you can imagine that this would be much less robust to perturbation because if you change in some direction, you're pretty quickly moving into another color, let's say. So this is really capturing that something that generalizes versus something that can't generalize will have this concentration or this consistency. Data from the same classes close together and separation classes are well-separated and robustness. If you move a little bit in any direction, you're not crossing what we might call a class boundary if you train a downstream classifier on this.

So what makes a good representation? You want it to be compact. We want it to be explanatory. We want to have concentration, data from the same classes close together. We want to have separation. Classes are well separated. And we want to have robustness to irrelevant perturbations, perturbations that we would not want to cross a class boundary. So a nice example of this-- and we'll get into this later-- but if you saw me from this angle, you'd want to be robust to me looking the other direction. It's still Sarah. It's not, somehow, a completely different class. So you might want to be robust to a perturbation that's something like pose.

So how could we encourage a model during training to actually achieve this type of representation? So one of the kind of high-level ideas that really has had a lot of traction here is the idea of explicitly encouraging good representations via feedback in terms of the similarity of data.

So you have pairs of either similar or dissimilar inputs. You're not just looking at one example at a time. You're actually looking at two or maybe three examples and giving information to the model about what should be similar and what should be dissimilar. So why might this be different than just looking at one image at a time? So there's actually a really nice example, I think, from psychology about the value of this contextualization or this comparison.

So tell me what an elephant looks like. Describe an elephant to me.

AUDIENCE: Very big, very bulky, and huge-sized.

SARA BEERY: OK, very big and bulky and huge-sized.

AUDIENCE: Large ears.

SARA BEERY: Large ears, OK. Anything else?

AUDIENCE: I think it has also-- yeah, I think I'd also like something in the front of it.

SARA BEERY: Like some big nose, trunk. So now if you knew nothing about elephants, you could draw almost an infinite set of things that might have those characteristics. And probably none of them would look very close to an elephant. So now you, in the front row, or in the second row, you, pretend I already know what a rhinoceros looks like. Now try to describe what an elephant looks like to me.

AUDIENCE: Instead of a trunk, it has a horn.

SARA BEERY: OK, so I know what a rhinoceros looks like. So now you're saying, OK, instead of having a horn in the front, take a rhinoceros, but add a long trunk, and then maybe add some big ears. I'm actually probably much more likely to have an understanding of what that thing is because I understand similarity and then these small contrastive differences. And I actually think this is a really nice, intuitive example of how we think about contrastive learning and why this type of contrastive signal is so valuable.

And I actually think, interestingly enough, I work a lot with species. One of the ways that we train people in identifying species is almost always contrastively. If I'm teaching you how to identify two different types of birds, I'm not going to say, oh, I'm going to describe this one from scratch with this massive, exhaustive list of independent attributes. I'll say it looks a lot like this other bird, but the important differences are the spot near the eye and the color of the legs. It's this comparative difference.

So the way that we actually try to train things based on similarity and dissimilarity, the original proposed approach was this thing called metric learning. And essentially, this was based on the idea that measuring Euclidean distance in your input data space. So for example, looking at Euclidean distance between all the pixels in one image and another might not be ideal. Instead, we might want to learn some metric, some representation that respects some desired properties.

And so our goal is to learn a metric where data points that belong together are similar, close together, and data points that are different are dissimilar far apart. And then the supervision becomes similarity information. So for example, maybe you'd have three images of your instructors, and you would want to explicitly say that the two images of Phil should be closer together because they're both Phil, and the image of me would be far apart because I'm not Phil.

And so, actually, how do we do this? So the very, very simple version, which is just in one-dimensional space, let's assume. Say you have some data points x1 through xn, and you have some form of supervision. And this supervision is not exactly the distances between things or how similar or dissimilar there are. We just have supervision in terms of which pairs are in the same category and which pairs are in different categories. So we have some set of similar pairs and some set of dissimilar pairs. And our goal is to learn a linear transformation, z equals Wx. So essentially, we're trying to learn this matrix W that will respect this similarity.

And so the idea here is that we can use Euclidean distance in representation space. And so now if we map out, OK, we want the distance between zi and xj to be equal to, factored out, the difference in terms of xi and xj transpose W transpose W the difference between xi and xj. We're were just taking that norm and factoring it out. Then you can define, basically, this A norm on xi and xj that will be W transpose W, which is a positive semidefinite matrix. We like positive semidefinite matrices because they're easy to work with mathematically.

And so this distance here actually is explicitly defined as the Mahalanobis distance. And so the Mahalanobis distance between two points is going to be defined by positive semidefinite matrix A. And it's defined explicitly as that. And so what you can show is that this, if we can learn some of optimal A, which essentially would be an optimal W for this z equals Wx, then that will actually be this transformation that preserves similarity and difference maximally.

So how could we actually phrase that as an optimization problem? How do we learn the right W here? Any ideas? So essentially, the first proposed approach, and one that was really introduced in the paper that defined the term metric learning, and they call it distance metric learning, was to specifically set this up as an optimization problem with a constraint.

So here, what we can do is we can say we want to find A. We want to find positive semidefinite A such that the minimum distance of similar-- we want to minimize the distance between similar points, so between those pairs that are known to be similar, while making sure that there is at least a distance of 1 or some epsilon between dissimilar points. So you're essentially setting this up as an optimization problem, where you're trying to bring the things that are similar as close together as possible, while enforcing that there is at least some margin around any point and any other point that is dissimilar to it.

And you can actually flip this as well. You can also think of it as keeping the distance between similar points within some constraint, and maximizing the difference between dissimilar points. And there's now a lot of different variations of this that have been proposed since 2003, which is now over 20 years ago. And it really is something that can be explicitly solvable. And you can, again, swap the objective and constraint.

And there's actually been a lot of related ideas and follow-ups to this, things like, for example, information theoretic metric learning, which explicitly proposed preserving distribution information. So for example, relative entropy between Gaussians, while trying to observe both upper and lower bounds as constraints.

But a really simple example of what this is, like why you might want to learn some projection, some linear projection that preserves these similarities and dissimilarities, is this one. I think this is incredibly intuitive. So say your original data was bimodal in some way. There's clearly red and blue categories. And then there's one mode, where there's red and blue categories over here, and one where they're over here. The very, very simple optimal projection here would be just removing the dimension. Here, it would be like along the y-axis. So now you're just projecting everything into the same plane. And now they're very easily separable from that perspective.

There's a lot of improvements and developments on top of this that you can do. So one thing is you can think about looking at nonlinear transformations. So instead of only learning a linear projection W, you can think about doing things like deep metric learning, where you're actually explicitly learning some of neural network that would map into a space that keeps things similar that we want similar and dissimilar that we want dissimilar. We've introduced these concepts of different types of contrastive losses. We'll get into those a little bit more.

And then another improvement and something that actually is just tracked to be quite helpful is the idea of normalizing your representations. So instead of then taking Euclidean distance, if you normalize representations to enforce that all of the distances have to live on the hypersphere, then distance would be essentially just equivalent to angle. And then you can look at inner products to measure similarity instead of needing to look at Euclidean distance. And this is nice because you don't end up having things that get really blown up by scale. You're removing the scale complexity.

So this deep metric learning idea is quite simple. Linear metric learning is you're trying to learn some linear transformation z equals Wx. Deep metric learning is you're trying to learn a nonlinear transformation z equals f of x, where now f of x is a neural network. And now you're not optimizing over the set of positive semidefinite matrices. You're optimizing over the weights of a neural network. And there, we can use things like stochastic gradient descent to do so.

So the intuition behind this introduction of contrastive losses is the following. So say I show you this moth. What kind of moth is that? Anybody really into moths in this room? I'm guessing no. Lepidoptera is the scientific term for the genus. Anyway, so that's a moth. I actually also don't know what kind of moth that is. There's another moth. Are these the same moth? Are they the same kind of moth?

AUDIENCE: No.

SARA BEERY: It's kind of hard to say, right? Well, OK, what if I showed you another moth? What if I showed you this moth? If you had to guess, and you had to say two of these were the same kind of moth, which two would you guess? Yeah, right.

So it might be really hard to explicitly say whether something is the same or different, especially because when you get into things like this, actually, there's this dimension of what is a category that gets complicated. Is the category all moths? Well, then, they're the same. Is the category white moths? They're still the same. Is it the category white moths on green leaves? Still the same. Is the category species of moth? Then maybe we don't know anymore. But it's actually very easy, regardless of what you want your downstream task and the dimension of granularity to be to say those two are definitely more similar than that one.

So the way we translate this into machine learning is we try to build a loss that explicitly says that the distance between dissimilar pairs must be much larger than the distance between similar pairs. And the very initial version that was introduced here is this idea of what's called triplet loss, which essentially demands that the distance between an anchor. So you have one point that you say is your anchor point, and then you have two other data points that are sent in too. And one of them is a positive example of the same as that anchor point. And the other one is a negative example that's different from that anchor point.

Now you enforce that the distance between the anchor and the negative example has to be bigger than the distance between the anchor and the positive example, plus some margin-- again, this kind of enforced margin. And so what this actually looks like in practice is you're just explicitly enforcing that you want to have some loss that penalizes if that condition is not met.

And so then, when you actually look at data examples here, so maybe you take in your initial data, and in that space, it turns out that, incorrectly, your model had put this first horse, the anchor horse x, and this monkey closer together than the anchor horse and this other horse. And now, after actually updating your model, taking a gradient descent step, the loss and that gradient would tell you that you needed to push, in that representation space, the horse and the monkey further apart and the horse and the other horse closer together.

And the way this actually is done is in a triplet network. You would have these shared weights. So you have three images that go in. You run all three of them through the same convolutional neural network. You calculate embeddings for all three. You take a triplet loss over those three different embeddings based on the anchor, the positive, and the negative example, and then you back propagate through that branch in the graph.

And so there are actually quite a few different improvements of this. One is actually pretty explicitly related to training efficiency. And that's just that you can now, instead of sending in all of these independent triplets, you can load an entire batch. And now you'll have some positive examples and some negative examples in that batch. And then you can construct every possible triplet of positive and negative and calculate the loss over those. And so that way you're actually getting more training efficiency. You're making the best use of the batch you've already loaded in.

And also, in this way, you're more likely to sample, for example, these hard negatives that give you a signal because, for example, if you send in an image or a set of three images and your representation already captures well that these two are-- the positive example is closer than the negative example with a margin, then you don't get any signal. So the easy stuff doesn't actually help you train because you don't have any signal saying, ah, no, you got that wrong. You need to move them apart. So you really need to see these hard examples, where your model is getting it wrong, to be able to get signal and to be able to actually get a gradient that trains your model to improve your representation.

So if you use this type of embedding, what you can see is get clusters. You have done this in a way where now things that are similar tend to be clustered close together. In this case, this is a triplet loss network that was trained over the Caltech UCSD Birds data set. And you can see that, in this embedding space, things that are close together are actually quite similar. Are there some other things that this thing has learned beyond maybe similar species? Yeah?

AUDIENCE: Environment.

SARA BEERY: Environment, yeah. So if you look at, for example, that cluster over there, it's a cluster where, yes, all of the examples of those birds are terns, the ones that are flying against a blue sky. I could tell you they're terns. Maybe some of you guys know they're terns. But they're also all pretty much birds that are flying and mostly birds that are flying against a blue sky. So you can imagine that similarity here has may be picked up on some more stuff than just maybe what we might be interested in terms of similarity.

And if you query using this, you say, OK, find me nearest neighbors in this embedding space, you also do start to learn similarity semantically. You start to learn that things that look similar are similar and things that look different are far apart. And this gets at this interesting question of what makes an image similar. So here, these are embedding spaces that are trained on much larger data sets. And we have some reference image. And then we have different things that are kind of coming in that are potentially similar or dissimilar.

And this was a paper, actually, of Phil's, where they explicitly asked humans what they thought was the most similar, trying to get this sense of how does a human brain think about similarity. And it turns out, there's a lot of different ways that we think about similarity. We might think about similarity in pose, perspective, foreground color, foreground-background orientation, the number of items, object shape. There's a lot of different ways that things can be similar. And what's the most useful way to have things be similar or different based on a representation space is probably going to be contextualized by some downstream task that we're interested in.

So how do we actually try to make sure that we're showing these images that are hard to the models that we get training signal? So we need to find basically hard negatives. And this type of hard negative mining will often look for a misplaced negative. So we need to try to find things that have been put closer to the anchor than a positive example. And being able to do this hard negative mining helps us accelerate our learning. The more we can find the hard things, the more we train quickly.

And one nice example of that is this. So here's our moths again. You can imagine that if you saw those first two images based on a lot of dimensions of similarity, like the pose, the background color, the orientation, these are actually more similar than this one on the bottom, but in some of semantic categorization space, we want to know the type of moth. These ones should actually be more similar, and that one should be further away. And those things like pose and background color are actually confounding dimensions right there. There are ways that we want our model to be invariant. And so finding examples like this would be explicitly ways to train your model faster to recognize what you want to be similar.

So this assumes, though, that we already know the right answer. It assumes that we know that these two moths at the bottom are the same and then that one at the top is different. But actually, the idea of contrastive representation learning and specifically self-supervised contrastive representation learning thinks about how we might take these ideas from metric learning and also self-supervision to build representation spaces that are not explicitly supervised. How do we get a good starting point or good representation if we don't necessarily have labels? And why might that, in some ways, be better than actually training something with explicit labels?

So a common setup for this type of self-supervised representation learning would be that we're going to have some encoder, some function that will map our data onto a hypersphere. So again, we're normalizing to enforce everything to be on the hypersphere. And we'll define some crossentropy for a softmax classifier that will discriminate classes based on similarity.

So if we know that these two images at the bottom are the same class, and this one is different, then we're going to try to explicitly pull positive pairs together and push negative pairs apart. So essentially, this is something like a crossentropy loss. We have an activation of a logarithm of something. And this kind of looks to us maybe close to a softmax, something close to a softmax.

So we can use inner products to actually do this comparison because we've constrained it to the hypersphere. So we can look at the angle instead of the distance. So now we're maximizing the inner product, which is explicitly trying-- sorry, what am I saying? No, no, no, so we want we want to make sure that the inner product is as close as possible to 1 because that would mean that they were as close as possible on this sphere. And now instead of minimizing distance, we can maximize that inner product that minimizes the angle between the two of them.

And then, essentially, what we want to do is we want to make this green part as big as possible. So that's maximizing the inner product between those two things. And we want to make the red part as small as possible, which is going to give us the largest possible overall thing.

Now we assume, for example, that we have a couple dimensions of symmetry, right. You can say that we want to have symmetry where we would have essentially if the first one is your anchor and the positive example is the second one, we want the distance between those two be the same as if you flipped the two. So it's enforcing some symmetry between them. The distance should basically be the same no matter whether your first one is the anchor or the second one is the anchor.

And we also want to match the marginal. And kind of what we mean here is for the distribution over all those positive pairs, we assume if we marginalize over one of those partners, we would end up with the data distribution. So it's starting point distribution over the data. And so now a really common-- so yeah, this noise contrastive estimation InfoNCE loss, there's a lot of similar losses in metric learning that introduce these types of ideas.

And essentially, if we want to try to do this specifically self-supervised, so we don't necessarily know already that these two things are the same category. And instead, we're going to do some fancy stuff to try to force our model to learn that an input data point is the same as itself after running it through a bunch of different augmentations, so perturbing it in a bunch of different ways. This actually can outperform some supervised pretraining, which is interesting.

So intuitively, for me, it's like, why would we want to not have supervision? Well, one thing is if our supervision is not well-aligned with the actual task that we want to use the representation for, we might learn dimensions of invariance or dimensions of variance that are explicitly unhelpful. But if we just do self-supervision, so we're just trying to learn something without any explicit definition of what is similar or dissimilar based on category, that might actually be better for a bunch of different possible downstream tasks, but not necessarily better for everything. It's a little bit nuanced.

And again, just talking about why we might map to a hypersphere-- so here, one thing is more stable during training. Why do we not want to have scale be a dimension of variance? It's kind of like logistic regression. It helps you regularize. And also we get this really nice behavior. So if you enforce that everything is on a hypersphere. And you enforce that similar things are close together on a hypersphere and different things are very far apart on a hypersphere. Then you can always build a linear classifier, just a slice through that hypersphere, that will actually be linearly separable.

So there's this nice kind of thing where, no matter what, we can always enforce that. We build a linear classifier on top of that hypersphere representation.

How do we actually make it self-supervised? How do you get this signal if you don't actually have labels about what's similar and different? So basically, what people do is they take all the random stuff, randomly uniform drawn from the data as negative examples. And as positive examples, you take your image, and you do a bunch of weird stuff to it.

So here, we have this original image of a dog, and we can do crop and resize, crop, resize, and flip. You can distort the color. You can drop or jitter the color. You can rotate it. You can cut chunks out of it. You can add Gaussian noise. You can add Gaussian blur. You can do things like Sobel filtering and get you this rough edge kind of representation.

And so then something like SimCLR, which is one of the original methods that explicitly tried to teach this type of self-supervised contrastive learning-- here, for every data point in the batch, you would generate two different random augmentations of that data point. And those two random augmentations are the positives. And essentially, the intuition here is you want the exact same thing under some reasonable perturbations to be mapped very close together, and then some random set of all of the other things in the batch are used as negatives for that pair. So can anyone think of any limitations of this? Yeah?

AUDIENCE: If you happen to have multiple pictures of the same thing in the data set.

SARA BEERY: Yeah, so what if you have multiple pictures of the same thing in the data set? So it turns out, and we'll get into this, that that actually is a challenge. And the assumption is that, well, maybe there's something in terms of similarity that we would still possibly want to capture if you have two dogs that are exactly the same under perturbation versus another dog, for example, or another lab or even another picture of that exact same dog. But definitely these types of positive negatives, almost in a way, are confusing when you're actually learning these representations. And there's explicitly work that shows that if you can remove those, even by using something like explicit supervision, it can actually really improve a representation. Yeah?

AUDIENCE: Is there an analogy to the perturbations for nonimage data types, for example, that are commonly used?

SARA BEERY: Yeah, so for example here, we're talking about SimCLR, which is explicitly an image-focused contrastive thing. So those perturbations are going to be data-specific. So for example, reasonable perturbations on images are things that we actually have some intuition that we want to be invariant to. We want to recognize the object is the same and is invariant under flips, for example.

This is a strong assumption that we're building in. And there are actually some cases where we explicitly don't want to build in those assumptions. If your downstream representation needs to be able to detect the difference between right and left shoes, if you use random flipping as an augmentation, it makes that almost impossible because if you say this is a right shoe and this is a right shoe and this is a left shoe and this is a left shoe, your model is incredibly confused.

So the same would be possible for any other type of data. Say you had, for example, data of specific molecules. You then would have to inject some domain knowledge in terms of what types of augmentations of those molecules are reasonable, where we would kind of want to maintain similarity, and what types of augmentations are actually potentially unreasonable. Same for text. Same for video. So the choice of augmentations actually does become quite important.

You can think of a lot of different variations of this type of contrastive learning that use self-supervision-- yeah?

AUDIENCE: On the previous slide, I just have a question. Why doesn't the model just learn something really stupid, like what about this setup? Is it going to guarantee, for instance, that two golden retrievers are going to-- say you get a different image of a golden retriever, and you make two augmentations of it, and then you pick a negative sample that's now a horse. There's nothing about this, I think, that says that those two golden retrievers are going to end up close to this picture golden retriever in your representation space.

SARA BEERY: Right, so if you augment this one, you augment that one, that one, and you force it close together, there's nothing from this particular training step that would say that either of them should be closer to the original image. But now you're going to sample these over and over during training in many different combinations and with many different randomizations.

And so the assumption is, essentially, though there might be some things in there that are more or less confusing, that generally, over time, you're going to learn that golden retriever-like things should all be somewhat close together because, more often than not, you're going to get two augmentations of a golden retriever and a boat, not two augmentations of a golden retriever and another golden retriever. Does that make sense? So I think it comes out in the noise kind of over time and, over many, many different types of categories, does equalize or get to something that has learned the signal you want it to learn.

So there are a lot of different clever ways to think about how to get that similarity signal that isn't necessarily just massively augmenting. So one example is if you actually have data of something like a scene and the depth image for that scene, then you can, for example, try to learn representations that put things close together if it's actually-- the image of the scene closely matches the depth of the scene. So you can learn representations that capture something about scene structure by explicitly leveraging RGB-D during training.

Or you could think about representation learning over video, where you take frames randomly sampled from the same video as positives and a frame from another video as negatives. Now, your model needs to learn something about possibly potential similarity over time. And there's lots of different works that do of thing.

You can think about using vision language as views in some way of the same scene. You have the text description of an image and the image itself. You try to push those together, and you try to push them far away from an image that doesn't match that text description. So there's a lot of ways of trying to think about how you can get this self-supervised signal. And a lot of modern work actually really is just people coming up with creative ways to supervise models without requiring labels because labels are-- handcrafted, human-generated labels are very expensive.

So essentially, what is this method doing? There's kind of two ingredients here to making this work. One is the contrastive loss, so which specific form of contrastive loss you use. And the other one is the data. And by the data, I don't just mean the underlying data set. I also mean specifically which types of positive and negative pairs we surface to the model over time.

So if we dig into the contrastive loss, and we think of this, again, as something like a cross entropy loss that tries to distinguish between data points, another way that you can motivate this loss mathematically is you can try to frame it as maximizing the mutual information between positive examples in the pairs.

And so essentially, if you think of it that way, you're really just trying to maximize a lower bound on mutual information between views of the same thing. And there's been some interesting work, specifically this work that showed that, actually, if, instead of using the standard contrastive losses, you just explicitly maximize mutual information alone, it actually worsened downstream performance. And looser bounds with simpler critics led to better representation. So essentially, the contrastive loss has to be doing something else. It's not just maximizing the mutual information between things that are similar. So what else is it doing?

So we'd said that we had a couple of properties that we liked that made representations good. We had concentration alignment. So data from the same class is close. We had separation, which means data from different classes are far apart. We don't lose information. And then we had robustness to irrelevant perturbations. We don't want to change something in some way that we think is irrelevant and have it change or move really far in our space.

So you can see how, explicitly, the alignment is really built into that loss from the beginning. We're saying we want to pull similar samples together. So that's enforcing alignment. Similar samples should have similar features. And then you can talk about this uniformity, which is essentially saying we want to push to similar samples apart, and we're pushing all the dissimilar samples apart kind of equally. You can imagine that the optimal way to do that would be to cover the entire hypersphere because if we don't use part of the hypersphere, then there's actually a way we could have pushed even further apart things that are different.

So here, this actually pushes apart all of the classes. We're here. Every class is, again, one point because we've said that the only information we have is that this image is the same as itself under perturbations. So we're really pushing apart all the data points on that hypersphere, which is kind of pushing for that uniformity in our feature space. And that could be as seen as a way to enforce separation. So you can actually see this, which is really interesting.

So if you take CIFAR-10 and you build these encoders that encode them into the S1 feature space with the hypersphere on one dimension or two dimensions, it's just a circle, and you visualize feature distributions on the validation set, you can see that if you use unsupervised contrastive learning, you get this distribution that is properly uniform around that circle and that embedding space.

Whereas if you actually use supervised predictive learning, you get a feature distribution that's not uniform. And if you just randomly initialize the network, it's far from uniform. We really don't get anything close to this uniformity. So you can see that almost it's actually happening. Something in the way that we're doing these unsupervised contrastive learning is really pushing and enforcing uniformity across the data points.

So how can we quantify this? So the idea is that you might want to explicitly be able to study alignment and uniformity, so things being close together and things being spread apart, using metrics. We want, actually, a way to quantify how aligned or how uniform a representation is. So for uniform, for alignment, we just do something like looking at the expectation over the distance between features for positive pairs. This is pretty intuitive. So you're just saying my alignment metric is how far apart do I think any two positive examples are going to be in expectation.

And then, for uniformity, instead of trying to explicitly say the expected difference between negative pairs, you can do a transformation of this using the Gaussian potential. And then you can look at using the logarithm of the expected pairwise Gaussian potential, which is essentially e to the minus squared distance. And this puts more focus on the smallest distances. And this objective of goal has a lot of connections throughout math. And it turns out the solution that optimizes this is the one that maximally spreads out the points on the hypersphere. So the uniform distribution on the hypersphere is explicitly the solution to this.

And so now the uniform distribution on the hypersphere becomes the unique measure that minimizes expected pairwise potential between negative pairs. So now if we want to dig into our explicit contrastive loss, one way to do that is we take asymptotics. We assume, OK, what happens as you take things to infinity? That's a good way to understand behavior generally when you're looking at mathematical properties.

So here, without getting too formal, we want to make connections between alignment and uniformity and our contrastive loss. So since our definitions are expectations, we'll look at the asymptotic behavior, specifically taking the limit as the number of negative samples goes to infinity. So say we have infinite negative examples for any given positive.

Then you can reformulate this to getting basically the combination two different terms that are expectations. And we're putting them here in green and in red because, basically, the first component of this gets minimized if every single point of the same category has the exact same embedding. So essentially, if the distance between positive pairs is just 0 and the second term gets minimized, if those collapsed points are-- all the points are spread out as far as possible, which is that uniform distribution.

So the first term gets minimized if and only if f is perfectly aligned. All the things that are same are exactly the same embedding. And then the second one is minimized if we get perfect uniformity, which is explicitly the uniform feature distribution, pushing everything as far apart as possible. And now these are actually measurable quantities.

So now you can actually look at representation quality as defined by alignment versus uniformity and look at how accurate, for example, prediction is on a validation set. And what you can see is you take a bunch of different models, so every single one of these points would be a different representation space or different thing that's been learned, and you can plot them in terms of their alignment versus their uniformity as measured by these quantities.

And you can show that the things that have the highest validation accuracy are the best possible minimizers of both alignment and uniformity. So this kind of red that goes into blue that goes back into red explicitly tells us in some nice, measurable way that alignment and uniformity are in fact good qualities for a representation space, and they map towards good accuracy on downstream tasks. And you can show this over several different data sets, though you will see that simpler data sets show this better than more complicated ones.

So what's the contrastive loss doing? So the loss function itself, we've just shown, explicitly encourages alignment and uniformity or concentration and separation. So then what is the selection of positive and negative pairs encouraging? We talked about how we generate those positive and negative pairs is through maybe all these different augmentations. So what are we teaching a model via our choice of pairs?

So augmentations of the same data point should be close. And then we want a learned representation that's invariant to perturbations induced by data augmentations. So we're trying to learn invariance to the set of augmentations we explicitly put in. And it turns out that finding the right invariances can be really hard. We talked about the molecule thing. We talked about in the right and left shoe thing. And so there might actually be some explicit invariances like, for example, the ones we talked about in the geometric deep learning lecture that you could actually explicitly hardcode. If you want to be symmetrically invariant, you could build symmetry directly into your representation.

And maybe there are some cases where you'd want to use one or the other. And it turns out that there are a lot of types of invariances that we might want to have that can be really difficult to define mathematically. And so this kind of experimental or maybe slightly more brute force approach of just explicitly saying, hey, if I do this to this thing, I want it to be the same, you can build those invariances in without actually being able to mathematically define explicitly what it means to be, for example, invariant to pose within a scene or distance in these complex geometric spaces, particularly for nonrigid objects.

So what we're saying here is essentially that the data is explicitly the dimension that encourages the robustness, that third category that we really wanted in our representation. That robustness to perturbations is coming just from our data.

So this means that to make all of this work, you specifically want to have this set of things that help us get those losses to converge and encourage the right types of robustness. And it turns out, for example, that these four things, really heavy data augmentation, like not just simple stuff, doing really, really weird things to your inputs within reason, having these projection heads-- we'll talk about this in a second-- having very large batch sizes, and selecting the correct data pairs and hard negative examples, these are really the ingredients that make this self-supervised contrastive learning work much better.

And you can actually see this in the literature. So here, they're explicitly ablating in the SimCLR paper how these different transformations affect the performance. And it turns out that the performance of your downstream classifier, if you do self-supervised representation learning using only crops, for example, is significantly lower, like 15% almost lower, than if you use all of the entire set of augmentations shown here on the left. So the data augmentation definitely plays a huge role here.

Another thing that actually plays a huge role is this idea of a projection head. So here, what people found experimentally is that if, instead of applying your contrastive loss directly to the feature representation you're trying to learn, you enable some learned projection g that's quite simple from h, which is the representation space you're planning to use, and some z where you're actually going to maximize the agreement between similar pairs, you're actually applying that contrastive loss, that this ends up also really helping performance downstream.

And g will be something quite simple, a linear projection or a very small multilayer perceptron. And then so you don't actually use z, which is like this projected space where you're calculating your loss as your representation. Instead, you use h. And this projection had improved performance.

And I think one of the ways that people have intuitively tried to explain it is that instead of-- it lets the model learn a representation that keeps a little bit more variance. So for example, if you're trying to explicitly enforce this contrastive loss, you're really trying to push things completely together that are augmentations of the same object. But actually, maintaining some dimensions of variance to some of those differences, it seems, may be a helpful thing. This isn't something that's been super formalized, but it's some high-level intuition for why we might want to use this representation as opposed to the thing we're actually calculating loss over.

And actually, I see some synonyms to this in transformers because when you think about a transformer, there's one set of projections where you're calculating similarity between things. And then there's this other set of projections that you use to actually pass information. And so it kind of, again, seems to suggest that it might be valuable to have differences between the representations we actually use to keep information and the representations we use to calculate similarity or difference.

So if you actually look at this, you can see you have these gains between no projection at all versus different dimensionalities of those projections and linear versus nonlinear projections. And again, h maybe gets to keep some invariance to dimensions of augmentation.

Batch size has also been shown to be really important. So SimCLR uses all the points in a batch as negative examples. And it turns out that the number of negative examples you have at every given point really significantly improves performance, particularly when you have-- early on in training, you can see that the difference is close to 10% between having a batch size of 256 versus a batch size of 8192.

This is another reason why all of us are building GPUs-- well, I say all of us-- NVIDIA is building GPUs that have bigger and bigger and bigger memory because we need to be able to capture batch sizes that are really big to be able to maximize performance. And it turns out this is very expensive. Needing to have huge batch sizes is costly from a computational perspective. And there's a lot of efforts to try to make that more efficient. So Moco, for example, you can go read. It was an improvement upon SimCLR that explicitly tried to make that the batch size dimension a little bit more efficient.

And then, finally, we talked about the selection of negative examples, how it's really important to try to make sure that negative pairs are informative. We want to find the hard cases. We're actually getting signal from our loss.

And it turns out that if you actually have labels-- and so here, labels would be maybe categories from your set of classes that you're interested in using your downstream representation to categorize-- that. If you can explicitly remove those, quote unquote, "false negative" samples, which is basically like you have golden retrievers, and you want to not tell your model that another golden retriever should be really far apart. And particularly, we want that far apart with a margin.

So it's not just saying it should be a little farther apart. It's really trying to push it outside of a margin far away from these points, which is something we don't necessarily want. If you remove all of those and you use only the true negatives, you actually end up getting much higher top one accuracy downstream. But again, this assumes that you have access to labeled data for all of your data points and that the labeled data probably needs to align with the downstream task you're interested in.

And so then there's a lot of people who've looked at how we can leverage some supervision, maybe not requiring all the points to be labeled, but maybe if you have labels for some of them, how we can look at supervised or even semi-supervised contrastive learning. And so here, the idea is that having the contrastive learning signal, that triplet loss signal, you get some more geometric and robust feedback than if you just do crossentropy loss.

So then if you have images from the same class, you can use those as positive pairs. Or if you don't actually have labels for everything but you do have labels for some things, you can use those positive examples as anchors and then enforce dimensions across, like whether you do or don't have supervisory signal that tries to get a little bit more structure and handle negatives and positives reasonably. And there's a lot of work along these lines. Cool. So representation learning using similarity seems to be really powerful.

So just to finish up the class, I want to do a bit of a case study with one particular data set. This is the iNaturalist 2021 data set. It has images of species, so 10,000 different species of plants, animals, and wildlife, insects. About 2.7 million training images-- so it's not on the order of billions, but it's pretty big-- about 50,000 validation images, and 500,000 test images.

And the interesting thing about this data set is it has pretty explicit structure. It's species, so you can look at the taxonomic tree over the set of categories. And here, if you start at the middle, this is high-order branching in terms of taxonomic tree. So it's very coarse grained. This would be something like the difference between Animalia and Plantae. These are animals versus plants. They're very, very different, and this difference is very coarse-grained.

And then as, you move closer to the edge, you would get things that are increasingly more and more similar. So different species within the same genera are going to probably be quite visually similar, though, of course, there are some exceptions from things like mimicry. So at the very fine edge, that's where you get things like the two moths I keep showing. These are different species, but from the same genus.

And so you can of unroll this level of granularity that's-- we've got this proxy for the level of visual granularity in the classes coming from the taxonomic tree. So you can roll this out on a graph and say, OK, kingdom is very coarse-grained. Going down to species is very finegrained.

So one of the things that's quite interesting is if we look at a supervised model versus SimCLR or MoCo, these two different approaches for self-supervised contrastive learning, and we look at accuracy now using a linear probe-- so that means basically taking a representation space and then just learning a linear projection that's going to actually be our categorizer. And we'll actually learn that linear projection with supervision. Then we can look at the accuracy across these different granularities.

And so this black line up top, that's supervised training from scratch. We're not starting with ImageNet or anything. It's just all of this is only trained on this data. And then, at the bottom, we see what we get from self-supervised learning, again with that linear probe on the entire data set, so on the whole training data set with the linear probe.

And what we do is we train both the supervised and the self-supervised, then the linear probe for the self-supervised representation on the finest level, so on the species level. And then instead of explicitly training them for these higher-order levels, we just roll them up. So you would say, OK, we're going to predict at the species level. Now you just aggregate all of your category predictions up a level based on that graph, on that taxonomic tree. You use that as your prediction for the next level, and et cetera. So we're basically training on the finest level, but then we're evaluating our ability to categorize a coarser and coarser levels.

And it turns out that if you look at this type of data set, you can really see that, at this finegrained levels, there's really not a small difference. There's about a 30% gap between a supervised and self-supervised training on iNaturalist. And if you do the same experiment on ImageNet, the gap is much, much smaller. It's more like 7%. And so one reason for that is this-- ImageNet is a very coarse grained data set. The categories are not very similar to each other. And you see the same performance or the same trend here. The gap is very small when you look at these very coarse groupings-- class, phylum, or kingdom. But then the gap actually grows as the evaluation is made more fine grained.

And so this interestingly captures some of those dimensions that we were talking about in terms of the ways that we have chosen our data augmentations, the pairs that we give the model or don't give the model, and the effect that might have. So first, if we look at this and we actually try to probe these representations that are learned using this kind of query approach or this retrieval approach, and we say, OK, we're going to start with this image. Now if we take our supervised model and we find the nearest neighbors in the entire data set, it turns out this is a hard problem.

And so there are a few examples outlined in blue, where we do actually get nearest neighbors that are the same species. But then we also get a lot of nearest neighbors that are different species. So even with supervision, this problem can be quite difficult. And it turns out that as you aggregate up the category set, no, these are not all the same species, but they are in the same of genus or family.

But now, if you take the same sort of SimCLR representation and you do the same thing, you use the same image, and you query into the representation space, you can see that none of them are the same species. What has the model learned? Yeah?

AUDIENCE: Oh, I had a question.

SARA BEERY: Yeah.

AUDIENCE: It relates to a couple slides back. You had the accuracy of the supervised and the contrastive learning. So earlier in the lecture, you were saying that the representation space that contrastive learning learns is uniform. In some sense, that's really good. Whereas the supervised version won't necessarily learn a uniform embedding space. But you're showing that supervised is doing much better, so maybe it's not that important to have a uniform embedding space. I'm trying to reconcile those two points.

SARA BEERY: Yeah, and actually, I think this will actually help. So it depends a little bit on the signal. So basically, yeah, it's usually good to have this separability. But for contrastive learning to work well, you need to have a good similarity measure for your problem of interest.

And so here, what I'm trying to show, actually, is that what we've learned is we've taken an image and we've augmented it in some way, and then we force those things to be close together. And it looks like that the types of augmentations we're doing is giving us more information about contextual similarity-- so, in this case, birds being held in human hands-- than it is giving us information about species similarity, which is what we're hoping this model will be good at and what we're evaluating it on.

So if you don't have a similarity measure, where you're choosing those positive pairs and saying these are similar and this is different that's well aligned with your downstream goal, which in this case is identifying species, then these contrastive objectives, they're giving you uniformity, but they're maybe not giving you the uniformity that you want, or they're not giving you the similarity that you want. Does that make sense?

AUDIENCE: So, in this case, if you were able to, say, control for birds versus birds in hands, like maybe segment out the birds, then do you expect that accuracy to be pushed down closer to the supervised accuracy?

SARA BEERY: Yeah, so I think the point here that I'm trying to make is that if you don't have a self-supervised signal that is capturing the task that you actually have in mind, then it's possible that the contrastive objective really will not build a representation that is good in the context of your downstream task.

Yeah, so if you did, if we had-- for example, if we had a supervised objective here, then you would expect it to be much closer to a supervised learning objective. And I actually think one of the things that I didn't do or-- this is some work that was led by Elijah Cole. It would be interesting to look at taking a contrastive objective, where now we're using supervised contrastive learning and see how that actually affects performance, though I actually wouldn't anticipate it would be so different from crossentropy loss in this case.

So I think this is interesting because it kind of points out how the benchmarks that we use really influence our idea of what's good enough because it turns out that, with ImageNet, we're saying, wow, without any labels at all, we're still building representations that are very, very close to as good as supervised representations. But it turns out that that's, again, contextualized on the task of interest. With ImageNet, our task is coarse-grained.

And it turns out, the augmentations were building in, they're capturing things that make it easy to separate coarse grained visual characteristics. Maybe they're relying on the fact that, at a coarse-grained level, all those birds on hands would be put together. That's fine because we just care about bird. We don't care about bird categories. But as soon as you have some of these categorizations that need to be much more subtle, then the augmentations would somehow need to capture that well. Yeah, any other questions about this? Yeah?

AUDIENCE: [INAUDIBLE]

SARA BEERY: So the question was, does SimCLR or does-- I mean, I think SimCLR is just one example of a self-supervised contrastive objective. Does that usually do bad on unseen data?

AUDIENCE: [INAUDIBLE] so you don't know what it is?

SARA BEERY: Sorry, can you try to repeat that one more time.

AUDIENCE: So in the negative samples, we are not doing any augmentations. We are not doing anything. So I'm just trying to understand how the model sees the negative samples [INAUDIBLE] not doing anything on it. So when it sees something which is not seen, how does it act?

SARA BEERY: So, I mean, in a lot of these cases, what we're actually showing results on is test data, which is unseen during training. So what we are actually showing that these things do for some things, like ImageNet in particular, do generalize well to unseen data.

But the point you brought up about augmentations on the negative examples-- and it is an interesting one-- and I think, actually, one of the things that is kind of funny is if you only did augmentations on the positive samples, you never do any augmentations on the negative samples, which in practice is not necessarily what people are doing, you can imagine that the model might learn something about statistic weirdness in terms of the input space versus not, like more natural statistics, and start picking up on that signal. So that is an interesting bit of intuition you brought up.

But no, generally, I think, actually, because we're showing this on unseen test data, I think there is something reasonable about that. And we've actually found that these types of contrastive objectives, or the triplet loss objectives, often will generalize better. So the example, I'm specifically going to bring up is this idea of individual re-identification.

In almost all, for example, like Celeb A, this recognizing celebrities again, almost all the best methods on this type of individual identification, where you explicitly want to be able to say, even if it's not a category you've seen before-- so, for example, a human that hasn't been seen by the model-- you want a picture of that human and another picture of that new human to be close together.

And it turns out that these metric spaces, particularly learned through contrastive objectives, things like triplet loss and then building up to more and more complex contrastive objectives, those explicitly are better at that open set clustering problem in a way that actually a standard crossentropy loss based on a fixed number of categories, there's almost nothing you can reasonably say about what will happen when you look at new categories. And in practice, actually, similarly to this, what we find is that if you use classification losses versus contrastive losses, the contrastive losses significantly benefit for that open set in terms of making things similar close together.

AUDIENCE: I guess the reason I got confused initially [INAUDIBLE] discussed in the lecture that for positive samples [INAUDIBLE], but for the negative samples, we just randomly pick any images. So I thought we were not doing augmentations, and it looks like we are.

SARA BEERY: I mean, not everything does. But yes, there are cases where people do augmentations on negative examples. Yeah, but I think that is interesting. Any other questions? Cool.

So yeah, to summarize again, we tried to capture some dimensions of what makes a representation good, and specifically that good representations might actually be really based on similarity and dissimilarity. We said that we want to make sure that representations are well clustered, that they're compact, and that they're separated, so the classes are spread out. We want to preserve relevant information, and we want to teach relevant invariances. We want the model to almost forget the irrelevant information.

Where here, we're explicitly defining what's irrelevant based on, for example, in those self-supervised objectives, showing different augmentations that we want the model to be invariant to, and in the supervised case, what we're saying is, OK, we want the model to be invariant to the things that are different about these two images of the same category. So that might, for example, be the context, the fact that the bird is being held in a hand.

And then we can do this in a supervised or a self-supervised way. And it all comes down to the difference between supervised and self-supervised, the way that we decide what's a similar pair. And so we can actually build a lot of knowledge into the systems around what should be similar and what should be different based on our downstream goals. Cool.

All right, any other questions? If so, you can come up. And otherwise, we'll see you guys next on Thursday.