[SQUEAK]

[RUSTLING]

[CLICKING]

SARA BEERY: So today, we're moving on from generative models, and we're going to start talking about out-of-distribution generalization. And this is, I think, one of the biggest open challenges for modern machine learning models, the ability to generalize to data that is not explicitly covered within the training distribution. And so, just a bit of motivation here.

We've seen a lot of success in machine learning, right? We have incredible models for image classification. This is a aggregate of the ImageNet data set. I would argue that, in many ways, things like simple categorization, the types of categorization we often try to do on data sets like ImageNet is pretty much solved. There are, of course, a few corner cases that are pretty difficult, or sometimes, even erroneously labeled or ambiguous that are still hard for modern models within ImageNet, but mostly, it's a solved problem.

We've seen pretty great success on image generation. We've been talking about that the last couple of days. Strategy games, things like machine translation, robotic manipulation, we're starting to see success in all of these.

But then there are a lot of cases where machine learning still isn't very successful. And sometimes, it can be surprising. So I think one interesting example here is in the space of self-driving cars. So we are starting to see self-driving cars be deployed. I think Waymo in particular is starting to actually have fleets deployed in many different areas.

And of course, there's complicated dynamics around why we're not seeing self-driving cars at scale. Regulation plays a huge role in that. But I think one of the main reasons is because we really don't have a good mechanism to guarantee the robustness of these models.

We can't guarantee that they are not going to make potentially harmful mistakes. And this really begs the question of, are these machine learning systems really ready for the real world? And what dimensions of deployment in the real world are we still missing? What do we need to be focusing on to be able to actually do this effectively?

And I do think there's an interesting point here, which is you could argue that a lot of machine learning is actually actively being deployed in the real world. I think the GPT models is a great example of that. People are using AI more and more day to day, but if I'm trying to characterize the types of AI that are ready for deployment versus the types of AI that, even if we try to deploy them, we invest billions of dollars into figuring out how to make them deployable, we're still not seeing that progress or actually really seeing it happen, I think it comes down to risk.

A lot of the cases where we see AI models deployed are what I would characterize as high precision human verification systems. So you need the model to be good enough to be useful, so it should work most of the time, but you also have a human verifier in the loop that is making corrections when something's not right. So you have this verification loop that's always happening with human intelligence.

ChatGPT is a great example of this. You prompt it with something. You get a result. Maybe it's useful. You probably edit it. You probably interact with it in some way. If it's just absolute trash, maybe you giggle about it, you share it with your friends, and then you prompt it with something completely different.

So there's this human brain that's in the loop, verifying that these things work well. And things like hallucination aren't so much of an issue if a human is making the corrections themselves. It's when we start trusting these models without being human verified that we start running into issues.

And I think another interesting example of that is things like a lot of the modern machine learning systems that are actively deployed in health care are what we call decision support tools. So they're a tool that maybe a radiologist uses to try to highlight or potentially give some advice on what, maybe, a diagnosis could be. And then the human is actually liable to verify that and make a decision.

And interestingly, there's been a bunch of fascinating work that's come out recently that really argues both sides of whether or not this is working. There's research that showed that these decision support tools are really great for trying to make sure that patient outcomes are beneficial, that doctors aren't missing things, and then there's conflicting research that explicitly says that the deployment of these decision support tools is actively harming patient outcomes. Because doctors are becoming compliant, they're not paying attention, and they're missing things in that long tail of the distribution that are actually quite important. So the jury's still out.

And fascinatingly, if you look at the regulation around what it actually takes to deploy decision support tool in a hospital, the metrics are kind of shocking. I'm not going to quote myself on the details, but essentially, it's like, you have to prove some percent accuracy on some small population of data. But it's not nearly what you would expect. I remember being quite surprised. Something like on the order of like 100 or 1,000 patients to be able to deploy a decision support tool. And you don't have to go through anything like clinical trials because you're not actually making the decision. You're just trying to support the decision of an expert.

So, really, when we have this case, this is where we start to see things working well. It's kind of the standard machine learning setting. And it matches a lot of the underlying assumptions we make in machine learning, even theoretically, that the data that we are training the model on is going to be IID, Independently and Identically Distributed to the data we're going to use the model on.

So you have some distribution over categories of interest, data, different representative visual characteristics within a category, and you're going to sample from that at random for training, and then you're going to do inference on the rest of the data. Training distribution equals test distribution. This is never true in the real world.

So basically, there has never been any such thing as a representative data set of all reality. And one of the reasons that this is true, even with our best efforts, things like these huge data sets lie on billions of image data sets that try to just tackle this problem with pure scale. We're going to train on all the data we have and hope that that is close to being representative and IID to the data we want to use the models on. It still doesn't work because the world is changing.

So if you capture all the data today, it's not going to necessarily be representative of a month from now, a week from now. Things happen, things change. And so we often have something like this. There's some fundamental difference between the training and test distribution once we move to a deployment setting.

And so here, maybe some version of that would be something like, all right, we've trained a model on Cambridge. So this is this data center. And now, we're trying to use that model in a different season. Now there's snow on the ground and the model starts to fail.

And there's a lot of different ways that we can think about how those distributions can change. The data itself can become perturbed. That could be adversarially. We'll talk about that in a bit. But it also can be simple things like, h, the manufacturer of the cameras for Google Street View cars made an update to their system, and now there's some statistical difference in terms of the sensor itself.

You can get a different label distribution. So maybe the set of categories of interest has changed. You end up with something that used to be really rare becoming much more common, or something just disappears entirely.

And then there can be lots of other types of shifts. These can be shifts in the size of the sequences or the graphs you're modeling, weather, geospatial shifts from different countries or cities. Source of measurement. That's like the sensor example I said.

So what actually goes wrong when we have these distribution differences? So I thought this was a cute example. It's a little hard to read, actually, probably because I took a screenshot. So this is a tweet by hardmaru that was basically talking about essentially, there was a failure mode in Roombas where they kept rolling over dog poop and then smearing poop all over people's houses.

And so in order to try to avoid this happening, they went out and they collected a very large, diverse data set of dog poop. So I'm sure that was someone's like research internship. So they're talking about, oh, OK, in order to do this right, the way that we're going to generalize to of new out of distribution thing, which is like dog poop that we didn't expect to be on the floor in someone's house, we're going to collect this representative data set of dog poop.

And then the response was from Dmitry. And he said, man, I wish they'd also created a diverse data set of rugs so it didn't confuse black stripes with cliffs and I could finally get my whole house cleaned. So this is pointing out you can collect data for the distribution shift you do anticipate, but there's always going to be things you don't anticipate. And then you end up with Dmitry, who bought a Roomba and basically gets one tiny square of his rug cleaned a million times.

So this is a silly, silly example, but the point here is really that we cannot possibly characterize and anticipate every way that the world is going to change. And so this idea that we can collect representative, sufficient data to just ensure that where we're going to deploy the model is IID from what we train the data on is quite difficult.

This is reflected in quite concrete problems when you're talking about AI safety, the safety of things that are deployed, particularly in high risk scenarios, things like medicine or self-driving cars. This was a paper that was explicitly characterizing what they thought were the largest problems in AI safety, and robustness to distributional change was one of them.

There was this really interesting workshop that happened back in 2022 at Princeton that they called The Reproducibility Crisis in Machine Learning-Based Science. They looked at about 1,000 different papers across a huge diversity of scientific disciplines, and they essentially found that they diagnosed a bunch of different potential reasons why they weren't actually seeing reproducibility or generalizability in the papers that they were analyzing. And one of the biggest failure modes here was data poisoning.

So basically, the way that they constructed the evaluation for these machine learning systems was, in some way, artificially IID in a way that wasn't representative of reality. And this is something that I actually run into a lot. I review a lot of papers for my other half discipline, which is ecology or environmental science. And in those papers, often, what I see is a mismatch between the data that people are developing and training and evaluating machine learning methods on and the claims that they're making.

So it's fine to have a limited data set, but then you have to understand what that means about what you can claim the effectiveness of the model is going to do. If you have built a system that identifies all of the individual macaques in a national park, you can say something reasonable about your ability to monitor macaques in that national park going forward, as long as you've evaluated your ability to understand the changes in time for those individual macaques. And so somehow, the way you evaluate these models needs to be representative of the actual deployment use.

And I think this takes a lot of really careful construction and thought. Actually, the way we think about designing evaluation systems for machine learning models is really non-trivial. And if you use any of off-the-shelf tool, things like Roboflow, it just automatically assumes that random splitting is fine and you get, often, very falsely high expectations in terms of the performance of your machine learning model because you've got data poisoning, essentially, strong, spatial, temporal, or contextual correlations in your training and test data that make the model think it's doing well when it's actually not.

So for today, we're going to be talking first about specifically adversarial examples and thinking about how you might train to address adversarial examples. So this is small perturbations, ways that you think about robustness on the level of individual data points. And then, after that, we're going to be talking about distribution shifts. So actually, just these large-scale strips from entire distributions of data and some simple mechanisms that people use to try to address them.

Though I do want to point out that both of these are pretty active, open areas of research. So I can talk about progress, but we don't have solutions for these. So first, if we're talking about adversarial examples, assume we have a simple machine learning classifier right. We have this image of a pig. The model says, yep, that's a pig. 91% confidence.

What an adversarial example does is it adds some very small amount of random-- or not random, but what looks like random noise to this image. And you can see it's 0.005 times of this thing that looks random. And so when you're adding this noise, which is explicitly not random, it's actually optimized to be maximally adversarial. You can actually change the prediction of a model to be something completely different, maybe even with higher confidence than the initial prediction, even though, to me, this image looks identical. I can see no actual visible difference between the two images.

And so this points to the brittleness within, often, machine learning models. They can be mostly accurate, but they can be really brittle to things that a human visual system is not brittle to at all. And this is a big challenge.

So you can imagine that if you could do this of adversarial attack in something like self-driving cars, you could actually get models to explicitly change their decisions about street signs. And that would require that you're able to actually generate something that, in physical reality, is not just related to individual images. It's actually in physical reality. But that can also be done.

So let's see, there's a-- I have a movie here, but I wonder if it'll play. Yeah, there we go. So this is an example. So here's a classifier, right? It's classifying banana.

If you put a toaster in there, it doesn't change the prediction. But if you put this very specially constructed adversarial sticker in there, it really convinces the model that it is, in fact, a toaster. So if we watch that again.

So the interesting thing here is that I would argue from a classification standpoint, this is really ambiguous. You would expect a model would actually maybe even reasonably predict toaster. They're about the same size, whatever. Whereas here, this looks nothing really like a toaster to us.

And so the interesting thing here is that this demonstrates that it's not something about high-level image characteristics, that you can actually really convince these models or trick these models even in physical reality. And there are fun examples of this, even in 3D. So this is an adversarial 3D printed example of a turtle that most image classification models will categorize very confidently as a rifle.

And you can actually show, there's a fun video if you go look this up. Oh, here we go. Perfect. Yeah, so it stays-- the prediction stays rifle, even if you move it around. It exists in 3D. And all you're doing is changing the texture.

And they basically showed that their 3D turtle was really consistently predicted as either rifle or sometimes as something else, but it was never actually categorized as a turtle. And you can do this type of adversarial attack for speech as well. You can add just a little bit of very explicitly targeted noise in a way that perceptually, if we listened to these, or even visually, within the waveform, looks like it's made very, very little difference, and it will actually-- the model will predict that the translation of that speech is completely different.

So what does this mean about our models? Are they useless? What about them is so brittle? Why does this happen. And can you prevent it? So a bit of a history of adversarial examples.

So this has been something that people have been looking at for quite a long time. Some of the first works in this space were from back in 2004. And there they were looking at adversarial examples for things like simple linear classifiers. And one of the original motivating examples was a spam filter. So if you tried to build some classifier that tried to categorize whether an email was spam or not, could you, basically, from a red team, blue team perspective, try to actually build emails that would explicitly get through your spam filter?

And this has only gone on from there. There's been a significant amount of work. This was a review back from even 2018. But now, particularly maybe in the context of today being voting day in the United States, there's a lot of work in this type of adversarial dimension looking at misinformation filters now. So how do you-- how do you try to detect and categorize active misinformation or deepfakes on the internet, on social media platforms, and then the active antagonism of trying to actually develop deepfakes that are undetectable by these algorithms.

So it's important and it's difficult. So we'll talk about how do you actually do this, in maybe the simplest sense, how do you create an adversarial example. So at first, we're going to assume, at least for now, that we have some fixed machine learning model. And so now, what we're doing is we're actually optimizing over data related to that model.

So what we want to do is we want to find a small perturbation to a given image that doesn't change the meaning to a human, but does change the meaning to a machine learning model. And so the model is going to output some probability of given theta, of some given machine learning model weights and architecture, some probability of y the category given x. We might take some softmax to get that prediction.

So what an adversarial example does is instead, now, you try to solve this optimization problem that finds the maximum over all possible small delta, so small changes to x, that you now instead predict a different category. So some target category that is not the original category. And you can do this for any category. You can find the noise that would make this pig look most like an airliner, but you could also find the noise that made this pig look most like, I don't know, your friend Bob.

And so you pick some wrong class and then you have some given input image, and then you find a small perturbation that actually maximizes this. And this will be within the constraints of some set of, quote unquote, "allowed perturbations." A really common thing here is actually that we look at this taking an infinity norm over the perturbation. So some intuition here is, instead of just looking at the aggregate difference between the image and the new one, instead, you're actually constraining it such that the largest change for any given pixel has to be less than a certain amount.

So the infinity norm essentially is saying you're going to actually just keep the maximum change for any pixel small. And that does tend to actually result in things that are unperceptible to humans. Cool. So how do you actually do this? How do you, given some machine learning model that you assume is not changing, how do you find that maximum perturbation?

Instead of trying to solve this as an optimization problem, you can actually also basically flip SGD on its head and you can do this thing that people call projected gradient ascent, where we're updating that data perturbation over time. And so first, you would take a step in the direction of the gradient, and then you project the result back into your feasible set. So you're constraining the amount that these images can change by design.

You're making sure that it's obeying that optimization constraint that delta can't be too large. And then you just repeat this over and over again. It's quite simple and it actually works quite well.

So how do you actually defend against this? Is there any way to defend against these active adversarial examples? Another way to think about this adversarial example, instead of trying to-- yeah?

AUDIENCE: For the previous slides, how do you actually-- what kind of algorithms can you use to actually find the delta? Do you need, like, for example, access to the gradients of the original model?

SARA BEERY: Yeah. Yeah. So this projected gradient ascent assumes that you have the model weights and the architecture so you could actually calculate the gradients. Yeah.

AUDIENCE: Is there a method that doesn't need the gradients?

SARA BEERY: So the question is if there's methods that don't need the actual gradients of the model. Yeah. I mean, you could do it in a brute force way too, right? You could just think about trying a bunch of things and just picking the one greedily that the result was worst. Might be more expensive. So this is just maybe a more efficient way to find something.

Yeah. And then there are other methods that try to do this, maybe in different ways. So not necessarily assuming that the model can't change. But that's, of course, a bit more complicated.

A different way to think about finding an adversarial example is instead of trying to explicitly maximize the probability that your image is predicted as a given label, you can also think about it as just maximizing the loss between your model's prediction of the image plus that adversarial amount and the correct label, y. Here, you're not saying, I really want this to look-- I want the model to think it's this specific category. You're just saying, OK, what's the noise that will make this most wrong, it'll make the loss highest for this example. Whereas, in standard training, you're trying to minimize that loss over all the possible thetas in the model itself.

So then what you can think about doing is like, if you have standard training where you're using SGD and you're trying to minimize the loss over all of the different data points in your model, you can think about training adversarially, trying to actually get your model to be robust to the types of perturbations that we're expecting this adversarial model might have-- this adversarial approach might have. So here, now, you have this two-loop, this nested optimization.

So now, you're trying to find the model such that for every image in your data set, the worst case perturbation is still not too bad, essentially. You're trying to, during training, find the worst case perturbation and then find the model that for all of the different-- all the different data points in your training data set, is still as good as possible, even in those worst cases. Does that make sense?

This also can be called adaptive data augmentation. And it's an interesting example of finding these hardest cases. So we learned a while ago about hard example mining. It's maybe most informative, sometimes, if you show models explicit examples of things are going to be hard and you want it still do well. Yeah?

AUDIENCE: What happens if there are two similarly hard examples, but very different from each other?

SARA BEERY: So the question is, what happens if you have two hard examples that are different from each other.

AUDIENCE: In this case, like, the maximally hard example for the turtle was rifle, but besides rifle, there's apple that is on a similar scale as being the rifle example, but basically, you've only trained for an adversarial example where it would be something related to a rifle.

SARA BEERY: OK. I think I get what you're asking. No, no, no. I actually think it's a pretty prescient question, it's just worded a little complicatedly. So essentially the question is like, what if the data changes in a way you didn't anticipate. And so this is really explicitly constrained to be robust to the types of perturbations that you've defined as within your reasonable set of perturbations, right?

The cases we talked about were something that looked very small amounts of noise added to images. But we'll talk about in a bit, most perturbations of data are not of that structure in the real world. Often, if you have something that's out of distribution in some way, adversarial in some way, it's not specifically the static noise. And for example, this type of approach would probably not work very well for those stickers because that's a different type of perturbation of an image. Did you have a question?

AUDIENCE: I was basically going to ask [INAUDIBLE] is that this won't help your self-driving car detect a pedestrian crosswalk. It's like, that's not a perturbation that would be modeled mathematically.

SARA BEERY: Yeah. Or, I mean, pedestrian in a crosswalk, probably, you could reasonably say you could think about ways to augment your data to capture that in a bunch of different ways, but then what you end up with is weird stuff. Like, there's this classic example of a bunch of stop signs being transported by a truck in the back of a truck, and there's a self-driving car behind a truck full of stop signs, and it's just freaking out because what is it supposed to do?

And so, yeah, this is, again, like I said before, it's just really hard to anticipate every way the world is going to change. So if you could do that, then you could reasonably try to be robust, either just through standard training, or if you didn't have the right training data, but you knew something about the way the world might change, you could try to be robust explicitly to those types of things.

You could use things empirical risk minimization. But that is often not the case in reality. And so a lot of the work in adversarial example, adversarial robustness, they have to make assumptions, and usually, those assumptions are violated in real world. Unfortunately. Yes?

AUDIENCE: Is there anything stopping you from a continuous loop of minimizing the loss and finding an adversarial example that maximizes the loss and just continue doing the back and forth? Or can a model eventually be trained so that you can't?

SARA BEERY: So for a reasonable set of perturbations, and assuming that maybe you try a bunch of different hyperparameters, it usually is often possible to get these to converge. But it can be difficult. Because you basically have two different optimization objectives that are conflicting. And so they can be pretty finicky to train. Yeah. Cool.

So now, if you're actually doing that type of adversarial training, what you actually do in the training loop is you would sample some data point x with the ground truth y. You would compute now the approximate optimal adversarial perturbation. And you could do that with that projected gradient ascent. And then you would compute the gradient given that perturbation.

And then you would update your loss given that gradient for the parameters. So essentially, for every model, you're trained-- for every data point, you're training another model. And we'll talk a bit about it next lecture, but this also is similar to things like meta learning, where, potentially, in your outer model training loop now for every data point, you actually have an inner training loop. So it's also quite computationally expensive to do this.

Cool. So what do these examples tell us? What does the fact that these adversarial examples exist tell us about the models itself? Arguably, I would say it tells us something about the input features, the structure, the statistical characteristics of the data that we're training the model on that become critical for the model's decision.

There's an interesting example here. This is training data trying to teach a model to categorize 4's versus 9's. And the interesting-- so this is MNIST, essentially. But the interesting point here is that what they've done is they've added spurious correlations that are very small.

So for every 4, they've added a white pixel in the top left corner, and for every 9, they've added a white pixel just one next to that. So now, what ends up happening is if you do these adversarial perturbations that are very targeted, you basically just change the location of that pixel, you get the exact opposite prediction. And this really gets at this question of, what are these models learning?

And they're going to learn whatever signals are there, and ideally, whatever signals are easiest to pick up. So this becomes just a binary classifier if you consistently have these correlations. The model just ignores the rest of the image. It learns all it has to look at is just that top pixel and it's a perfect categorizer.

So what this can tell us is that for any given model, the features that are captured in that model, some amount of them will be useless, some amount of them will be robust. So these are features that are correlated with the label, even if they've been perturbed in some way. So here, they're showing things that are quite semantic, things like specific attributes of a cat ear, a dog nose, or the structural shape of an animal.

And then there's another set of features that you could argue are not robust. And these are going to be features that are correlated with the label in the training data set, but can be flipped with perturbation. So perturbing these non-robust features can actually change your prediction.

And so many features can be correlated with the label in your training data set, and thus, be predictive. And they might also be correlated with the label in your test data set and be predictive. And so they can help with accuracy. Like, your accuracy is good, but they're features that are beyond what a human would use, or sometimes, maybe, in some subtle cases, things that we might use based on stereotypes, but if we think rationally about it, we might not want to use them.

There's a lot of examples of these types of stereotypes or biases in training data sets that are against women or gender minorities. Like if you show-- early ChatGPT was very, very popular. Oh, if you asked it to generate-- or not ChatGPT. Diffusion models.

If you asked it to generate images of a doctor, it gave you a bunch of white men. You asked it to generate images of a homemaker, it would give you a bunch of women. Like, biases that are built into the data sets, that might be correlated with society, but unfortunately, are things that, if we think about it from an ethical perspective, we don't want models to make decisions on.

This gets really complicated. So there's this entire deep literature on fairness in machine learning models and how you think about equity when the data we're training models on is just as biased as our modern society. And these models tend to exacerbate societal biases.

So correlative predictive features are not always something we want the model to use, both because it can make it non-robust, but also because it can exacerbate bias. And where do the correlations come from? They really come from the data.

This is a fun example, dogs and cats. Turns out, if you put a dog in a bow tie, the model will predict it's a cat because there's this weird bias where people put bow ties on cats more than dogs. Or this is a really fun one. If you look at all the images of fish in the ImageNet training data set, you get grumpy dudes in sunglasses.

Maybe very happy dudes in sunglasses. They're going fishing. They love it. But the model's not learned fish. It's learned like fish in the hands of a person. And then, basically, this really is just rooted in how we create data sets.

So in an ideal world, we would have representative real-world images that captured the distribution of everything we would ever care about. We would have expert annotators who were then able to build perfect annotations on top of that, and then we could use that to carefully and meaningfully construct a benchmark where we could evaluate machine learning performance and happy sun and rainbows. But in the real world, we often have images scraped from the internet, and then we have some form of automated or crowd-collected labels, and then that results in these noisy, complicated, biased annotations, and that might create an easy to optimize benchmark, but it's not necessarily something that is rooted in reality.

And one argument is that all these models are learning are shortcuts. It's all different types of pattern matching. And so shortcuts here are features that are correlated with the label in the training data, but maybe not correlated with the label under realistic distribution shifts. And it turns out that these models will use them and will not generalize if then those features are no longer correlated.

Interestingly, there's a dimension of imbalance here. In some of the things that I try to do, I'm doing species identification in static cameras, and what we get is day-night bias for nondiurnal species. Raccoons show up a lot at night, and we still have some images of raccoons during the day during training, but the vast majority are at night.

So the model's learned that it's like, nighttime, kind of like, fat, cat blob thing. Probably a raccoon. And if you show it, daytime raccoon images, even though it's seen them during training, the performance is just much worse. Because this correlative predictive factor is not there.

So the models will use these shortcuts when they're provided because we train them based on likelihood. And this is almost-- this is often related to the data itself. And so, interestingly, adversarial examples, those types of perturbations we talked about can transfer across models trained on the same data set. So the same types of shortcuts will often be learned, even if you use different architectures, because it's trained on the same data set.

And what can these shortcuts look like? Yeah, things like this is a picture of a green hillside, and this model is explicitly saying a herd of sheep grazing on a hillside, but there's no sheep. This one says a flock of birds flying in the air.

These are instead these really nice tree-based goats. I don't know if you guys have seen these. They're amazing. They're goats that climb trees in the Mediterranean.

And then, things like this. It's people holding goats, but the model predicts they're dogs because it's seen less of these examples of people holding dogs. And these can be-- these are all silly, but they can be pretty problematic as well.

There was some early work that was showing that you could do a lot of really nice predictive diagnostics on X-rays, and then there was this pushback work that showed that, actually, what the models were learning was to detect where the X-ray was acquired, so which hospital, and then calibrate the predictions according to the distribution of diseases at that hospital, which is, again, probably the type of useful, but unwanted correlation.

Or similarly, it's learning that if an image has a ruler in it, an algorithm is more likely to call a tumor malignant. And so you can imagine now that, essentially, what this is meaning is like you're much less likely to detect the examples where a doctor hasn't already thought that it might be malignant and is measuring it with a ruler. So not all predictive patterns are desirable. And we talked about those fairness examples as well.

And there are tons of examples of this. You can think about this in interesting ways. So here, this is essentially going from IID to OOD-- independently identically distributed to out-of-distribution. You can see domain shift, adversarial examples, types of distortions, pose warping, texture change, background shifts. All of these will often result in a human looking at these images and saying it is the same category, but the model ends up not predicting the same category.

So this is things that we want the model to generalize on, but it doesn't. And then there's the opposite too, which is things where the machine learning model is very convinced it's the same category, but a human would not be. And this happens for transformer models as well. This was a case where people had trained transformer models to look at reasoning, but it turned out that they pretty explicitly couldn't generalize out of the distribution of the number of the sequence lengths that they were trained on.

So adversarial training, when you're training it to be robust to these types of non-robust features, and now, here, we're assuming that we know the dimensions of robustness, which, of course, we don't often know, they do tend to teach models that have outputs that are hopefully stable under these types of adversarial perturbations. So you get invariance to these non-robust features.

And of course, this type of just noise addition is not necessarily representative of every possible way that this data could be perturbed, but there are some nice papers that show that if you train in this adversarial way, it can be much more expensive, but you can get features that tend to be more human intuitive. I'm always a little bit iffy on the use of things like these activation heat maps as concrete explainers of what a machine learning model is doing, but these do look interesting.

So here, if you look at a primate, here's just some-- if you train with an L infinity or an L2 norm, you can see that the features are a little bit more semantically meaningful, they look a little bit more like the object of interest. And similarly, if you're trying to build these adversarial examples for robust models-- now, if you've trained it to be adversarial adversarially robust, then the thing that you're-- the data that your model actually generates to be adversarial has to look a little bit more like the semantic category you're trying to make adversarial to.

So here, if you start from these pigeons and you try to find the adversarial example that makes them look maximally like dogs, this kind of looks like dogs. It just really doesn't look like pigeons anymore at all. And so these are things that a human is more OK with these types of mistakes.

And there are some papers that show that adversarial trend models maybe build representations that transfer better to other data sets, at least specifically with some of the data sets shown here. Cool. So adversarial training you can think of as like another very targeted form of maybe regularization or a different type of domain adaptation, but one that's quite targeted to try to make the models robust to some explicitly constructed form of perturbation.

So now, what if we're not talking about perturbation on individual data points, but we're instead talking about perturbations over entire distributions? So here, instead of having the same distribution for training and inference, now, we have a pretty different distribution for training and inference. So this is basically the world I live in.

Here, this on the left, that's a map of an estimate of alpha diversity globally, which is an estimate of, essentially, the number of unique species that exist in any given place. And then, on the right, this is a heat map of the three billion existing species occurrence records in the Global Biodiversity Information Facility. And you can see that they are clearly not super well-aligned. They're almost anti-correlated.

So you can imagine that if this is my training distribution, and somehow, that's the distribution of the data maybe I want to use the model on that, that's a pretty significant distribution shift. And then, add on top of that the fact that all of it's changing all the time and it gets pretty messy. And this was the motivation, actually, behind the creation of this Wilds benchmark, which I built with many other very talented people.

Specifically, it was reflecting the fact that a lot of the literature around distribution shift in the machine learning community were these types of not realistic distribution shifts, like, adding noise to things, or taking MNIST and rotating it by 45 degrees, or making it a different color. Not necessarily super representative. Even some of the examples that were designed on real data-- I don't know if you've ever heard of the waterbirds data set.

It's basically a data set of waterbirds where the training data has waterbirds not in the water, and the test data has them in the water. It's a background shift problem. But it's pretty artificial. You've constrained it so that the categories are balanced, that you're only looking at one dimension of shift, and the real world is quite messy.

So this is a benchmark that looks at distribution shifts over a pretty diverse set of real-world domains, everything from different scans from hospitals, to different locations of species categorization, generalizing mapping of poverty across countries, looking at comments across demographics. And essentially, what this benchmark highlighted was that when you move out of distribution in these realistic ways, basically, model performance degrades significantly. And the way you actually think about measuring this is you can try to explicitly construct benchmarks that look at performance over realistic distribution shifts.

So here, we looked at the gap between a control, in distribution test data-- and note that this is not actually identical. It's just randomly sampled from within the training data locations. And then you have some new cameras that are held out that are out of distribution. And you can see that macro F1 performance degrades by like, 16%, which is pretty significant.

And if you're thinking about what are the factors that could lead to this, there's a bunch of different things. It could explicitly be things like background. It can be things like pose of cameras. So the top and the bottom are more horizontal. The one in the middle is posed a little higher, so you're getting different angles on species.

Here, you can see some of them have passive infrared at night. Some of them have color. So the dimensions of shift in the real world can often be quite complex. But not just that visual shift. We also have pretty explicit what can be termed subpopulation shift.

So essentially, for any given location or any given camera, it has its own unique distribution over the categories of interest. It makes sense. That camera is placed in a microhabitat. That microhabitat has a local set of species that's interacting with it.

And it turns out that this is basically true for every ecological problem I've ever worked on. So different ecosystems will have both subpopulation and visual distribution shifts no matter what. This is looking at detecting individual trees from aerial data across the National Ecological Observatory Network. Trees both look different and are at different densities.

You can similarly try to think about categorizing detecting and counting fish in different rivers across the Pacific Northwest, or things like categorizing urban trees across different cities in North America. And you can see that both visually and from the sense of subpopulation here, the colors are mapping the 20 most common genera of trees. And the differences in colors between cities capture that subpopulation distribution.

And it turns out the performance often has a very strong correlation with subpopulation distribution similarity. So this is if you're looking at training on one city for urban forests and testing on another, you actually get these kind of block-wise structures where cities that have similar training distributions will-- you'll generalize better within the set of cities that have similar training distributions. It's kind of intuitive. These models are trained on likelihoods, so you're almost building a prior based on the subpopulation distribution within any given training set. And if you're matching that more closely, your model is more likely to predict well.

And there's some nice examples of simple approaches you can try to do to actually match the expected distribution of your test data, even without additional training. So things like if you can build or if you can represent a prior, if you have some way to construct a prior over the set of categories you expect to see, then you can just do a really simple Bayesian remapping of the predicted probabilities to take into account that prior. I'll talk about that a little.

So what do you do about distribution shift? So one thing that people have looked at is this that's quite similar to the adversarial example we talked about before is something called distributionally robust optimization. So far, we were saying we were allowed to perturb each data point by some limited amount, but an alternative is that you can actually think about perturbing the entire training distribution by some amount together.

And so now, what this looks like is distribution-- robust optimization essentially just assumes that the worst case data distribution shift might occur. So you might have some different samples in terms of the densities. You might take some of the data away, et cetera, et cetera. And so they think about ways you can perturb the distribution of your training data set.

And then they take a similar approach to what we did with the adversarial examples, but now, you're trying to find the model such that-- in expectation over all of the possible distribution shifts on a data set, you're still going to minimize the loss. And so you're basically, again, finding that worst case scenario and trying to make sure that the model is still robust to that for any given-- for any given distribution. But again, this assumes that now you can only move the data distribution a certain amount.

And measuring the difference in pixels is one thing, but actually trying to think about measuring the distance between probability distributions can be quite complicated. So there's Chi squared distance, people use things like Wasserstein distance, maximum mean discrepancy. But how you actually measure how far away two distributions are, if they're simple, like measurable distributions in one dimension is one thing. If you're trying to measure the difference in distribution between 1 million image data set and another million image data set, it gets a lot more complicated.

And a lot of times, what people will do is they'll actually leverage learned representations of those data sets and then try to make some metric for the similarity across, maybe, the clustering, the representation space. This stuff is pretty tricky. It's not definitely something that works super well in practice.

So when you're actually trying to think about what this means for generalization, essentially, DRO optimizes for this set of possible training data sets and distributions. And so if you have some underlying data distribution, p, and your empirical training data is p sub n, so some subset of that or some specific characterization of that, then if you've built some distance metric and you say that the new distribution of where you're going to use the model is within some distance of the true distribution, then you're guaranteed to perform well on the true distribution. So you're guaranteed to generalize.

But of course, I'm sure you can see the challenge of the limitation here, which is that you have to be able to measure the distance between distributions, and you have to be able to sample these hypothetical distributions in a way that's realistic. And of course, the more complicated your data structures are, the harder that gets. So interestingly, this has been applied pretty effectively when you're talking about some of these class imbalance challenges.

So here, you can use DRO. So you assume your population has some number of subgroups. Population here could be any given category. So let's say the raccoons. And now, maybe our raccoons have two subgroups, like nighttime and daytime images of raccoons.

Traditionally, you would think about trying to minimize the empirical risk or the average error across these different subgroups, assuming you have access to the actual labels of those subgroups, which maybe for day versus night, we do. Because we have a clock for our time or for our camera. But it turns out that if you have a minority subgroup, if you get 50% error on that minority group in terms of accuracy, that's only 10% overall error. So your model might learn, if it's effective, to just ignore the minority group.

So you can ignore the minority group and still get decent loss. And so the idea is that DRO can learn to effectively automatically reweight your data to pay more attention to those minority classes. And so what that might actually look like is it's forcing the model to place less emphasis on noisy features and more emphasis on useful and predictive features.

And it also seems to be explicitly, empirically, at least, shifting the distribution so that they're more separable. But there's some really nice work, actually, that shows that you can do this, and there's adaptations of DRO to do this, even if you don't know what the subgroups are or how many subgroups there are.

So people use DRO a lot for some of these fairness type of problems. You want a model to learn to recognize doctors, and you don't want it to be biased against female doctors or Black doctors, even though the training data is biased against those groups. And so in that worst case distribution shift, you could imagine, you might only sample those minority groups, and then your model performance would be significantly worse if you aren't making sure that you're robust to that.

So what do we do about distribution shift? First, you can try distributionally robust optimization, or there's many other ways to think about building robustness, extensions of this, cetera, that take an algorithmic approach to trying to reduce-- to trying to make models more robust to these types of distribution shifts or make them more fair. You can also do things explicitly learn priors over the distribution shifts that you see.

So, for example, subpopulation shift. So this is a case-- this is an example by [INAUDIBLE] where they looked at species identification, and they explicitly learned separately a modular, like, spatiotemporal prior over places and times that then they use to remap the likelihoods of any given probability or any given species based on where the image was taken. Makes sense. This is what ecologists do.

Often, there are categories that look really similar. And if you just saw the image without any context about where, when it was taken, it might be hard even for an expert to be able to distinguish them. But if you have that additional information, it's quite useful. And so this is just addressing this generalization problem as an image categorization with this kind of specialization that's coming from a prior based on space and time.

So you can think about capturing some of the information you might have about the structure of your domain or the distributions spatially, temporally, et cetera, within your domain, and actually building that directly into your model as a way to do a better job. And those models improve species accuracy by like, 10% on a naturalist. You can think about mechanisms like domain adaptation.

So this is actually an entire category of models that tries to learn given a source domain and a target domain where maybe you don't necessarily have labels for your target domain, how to align their representations during training so that you can enforce generalization or adaptation to a new target domain, even if you don't have a lot of data or labeled data. I'll talk a little bit more about that in next lecture.

This is pretty interesting. You can come up with creative ways to try to diagnose the failure modes of your model, which often teaches you more about the biases in your initial training data than anything else, but they can be really interesting. And so there's ways to think about doing that by just striving your evaluation. So breaking down evaluation per category, along different dimensions, like night versus day, or different sexes of the participants, assuming you have the labels to be able to do that, break that down and look at the relative maybe call it fairness across the category set.

This is an interesting example coming from Aleksander Madry's group, where they actually use modern generative AI models to generate hypothetical counterfactuals to try to understand or diagnose failure modes in training. And so here, for example, what they found-- one interesting example that they found through this diagnosis, and then they actually were able to demonstrate it on real data, is that these models are really good at recognizing plates if they have food on them, but if it doesn't have food on it, it's consistently being categorized as like, trays or buckets. So you can try to diagnose the failures, and that might be so that you can try to then collect training data to be robust to those failures. It also can just be useful if you're trying to think about how and in what way you should trust a model in deployment.

I would argue that probably the best thing you can do in any of these cases is just get training data that's representative of your test domain. So in practice, a lot of these algorithmic approaches, these creative ways to try to train these models to be robust, to be effective, usually, they're just significantly worse than just more training data that's representative of your test domain. So from a practical perspective, if you really need something to work well, you have to invest in the right kind of training data. And if that's something that you just cannot get for various reasons, various constraints, maybe the species is super endangered, or you're trying to get a model to work on a planet that we've never landed something on, or we're trying to get models to work under the Arctic or in the deep sea where we just are not actually able to collect large-scale training data, then you're probably going to have to have a human look at a lot of the data still.

This kind of where do we actually see these things work in practice, often, it's really just where we have in-distribution training data. And so this is a really active area of research. And for any of you that are working on trying to bring deep learning to your own scientific challenges or problems, I would totally recommend that you look into this literature when you think about, how do I get the model to work where I want it to work? But I also think that if you have limited time and effort, sometimes, it's just worth it to label more data. At least, practically, that's what I've found.

Cool. So, in summary, out-of-distribution generalization is a big challenge. It's still a big challenge. But digging into it helps us understand what neural networks learn, and it also really helps us try to understand this very tight and complex relationship between the training data and where we can effectively use models. And then we talked about adversarial examples and distribution shifts today. Are there any other questions? Yeah?

AUDIENCE: Has anyone looked at activation functions? Like, do activation functions change how easy or difficult it is to fool one of these?

SARA BEERY: So you mean like things like Grad-CAM?

AUDIENCE: [INAUDIBLE]

SARA BEERY: Oh.

AUDIENCE: There's a stark non-linearity between [INAUDIBLE] it's not.

SARA BEERY: Gotcha. The question was, have people explored how the choice of activation functions, so something like a ReLU or a GELU Or something, how that affects generalization or robustness. I bet, but I don't know any literature on it.

I think my intuition says it wouldn't have a huge effect. But it would be interesting to dig in and try and see. And it might have a bigger effect on simpler data sets than more complex ones. So there might be papers that show that has an effect on MNIST style training or CIFAR style training, but that might not actually generalize versus all of the other different complicated ways that models and architectures affect performance.

But that's totally just my best guess. I'm actually kind of curious. I'm going to go look into it. Does anyone else-- does anyone else know? Have people looked into robustness and--

AUDIENCE: Sorry, that's another question.

SARA BEERY: Yeah, OK.

AUDIENCE: So this you talk about, this more like supervised fine-tuning. So having robust-- finding robustness examples. But would that be more cost saving to just reinforcement learning when you have that noise and then add whatever noise happens and then just label it by just like, human feedback. Seems like a much more cost saving instead of this example by example.

SARA BEERY: So the question was, instead of trying to do this type of robust training to hypothetical different perturbations, both to the images or the distributions, does it make sense to use other types direct feedback, like active human intervention, things like human in the loop learning, or possibly something like reinforcement learning, or DPO might be an example, Direct Preference Optimization.

So, yeah, people totally do that. It can be much more efficient if you have a specific target data set in mind. This idea of how do you get to a good result on that data quickly is, again, a pretty active area of research, and people do lots of different things to do that effectively.

Domain adaptation that I mentioned is one. Fine-tuning, if you're talking about supervised fine-tuning, assuming you have labels already, or you can try to be more efficient in collecting the labels to get to that good result on that data. But I think a lot of the work I talked about today was really with this goal of being generalizably robust, like, robust to any imagined hypothetical change.

And of course, then the limitation is you don't know the way the data is going to change. And so if you know exactly the way the data is going to change, you have a target in mind, then optimizing for that efficiently is totally possible. Does that distinction make sense?

So it's maybe the difference between trying to be robust to distribution shifts versus trying to adapt to a specific distribution shift, or specialize to a specific distribution shift. Yeah. Any other questions? Cool. All right. Take the extra time and go vote.