Whether to use end-to-end deep learning

(SPEECH)
Let's

(DESCRIPTION)
Text, End-to-end deep learning. Whether to use end-to-end learning.

(SPEECH)
say in building a machine learning system you're trying to decide whether or not to use an end-to-end approach.

Let's take a look at some of the pros and cons of end-to-end deep learning so that you can come away with some guidelines on whether or not an end-to-end approach seems promising for your application.

(DESCRIPTION)
New slide, Pros and cons of end-to-end deep learning.

(SPEECH)
Here are some of the benefits of applying end-to-end learning.

First is that end-to-end learning really just lets the data speak.

So if you have enough X,Y data then whatever is the most appropriate function mapping from X to Y, if you train a big enough neural network, hopefully the neural network will figure it out.

And by having a pure machine learning approach, your neural network learning input from X to Y may be more able to capture whatever statistics are in the data, rather than being forced to reflect human preconceptions.

So for example, in the case of speech recognition earlier speech systems had this notion of a phoneme which was a basic unit of sound like C, A, and T for the word cat.

And I think that phonemes are an artifact created by human linguists.

I actually think that phonemes are a fantasy of linguists that are a reasonable description of language, but it's not obvious that you want to force your learning algorithm to think in phonemes.

And if you let your learning algorithm learn whatever representation it wants to learn rather than forcing your learning algorithm to use phonemes as a representation, then its overall performance might end up being better.

The second benefit to end-to-end deep learning is that there's less hand designing of components needed.

And so this could also simplify your design work flow, that you just don't need to spend a lot of time hand designing features, hand designing these intermediate representations.

How about the disadvantages.

Here are some of the cons.

First, it may need a large amount of data.

So to learn this X to Y mapping directly, you might need a lot of data of X, Y and we were seeing in a previous video some examples of where you could obtain a lot of data for subtasks.

Such as for face recognition, we could find a lot data for finding a face in the image, as well as identifying the face once you found a face, but there was just less data available for the entire end-to-end task.

So X, this is the input end of the end-to-end learning and Y is the output end.

And so you need all the data X Y with both the input end and the output end in order to train these systems, and this is why we call it end-to-end learning value as well because you're learning a direct mapping from one end of the system all the way to the other end of the system.

The other disadvantage is that it excludes potentially useful hand designed components.

So machine learning researchers tend to speak disparagingly of hand designing things.

But if you don't have a lot of data, then your learning algorithm doesn't have that much insight it can gain from your data if your training set is small.

And so hand designing a component can really be a way for you to inject manual knowledge into the algorithm, and that's not always a bad thing.

I think of a learning algorithm as having two main sources of knowledge.

One is the data and the other is whatever you hand design, be it components, or features, or other things.

And so when you have a ton of data it's less important to hand design things but when you don't have much data, then having a carefully hand-designed system can actually allow humans to inject a lot of knowledge about the problem into an algorithm deck and that should be very helpful.

So one of the downsides of end-to-end deep learning is that it excludes potentially useful hand-designed components.

And hand-designed components could be very helpful if well designed.

They could also be harmful if it really limits your performance, such as if you force an algorithm to think in phonemes when maybe it could have discovered a better representation by itself.

So it's kind of a double edged sword that could hurt or help but it does tend to help more, hand-designed components tend to help more when you're training on a small training set.

(DESCRIPTION)
New slide, Applying end-to-end deep learning.

(SPEECH)
So if you're building a new machine learning system and you're trying to decide whether or not to use end-to-end deep learning, I think the key question is, do you have sufficient data to learn the function of the complexity needed to map from X to Y?

I don't have a formal definition of this phrase, complexity needed, but intuitively, if you're trying to learn a function from X to Y, that

(DESCRIPTION)
An inset photograph of a human hand x-ray is displayed.

(SPEECH)
is looking at an image like this and recognizing the position of the bones in this image, then maybe this seems like a relatively simple problem to identify the bones of the image and maybe they'll need that much data for that task.

Or given a picture of a person, maybe finding the face of that person in the image doesn't seem like that hard a problem, so maybe you don't need too much data to find the face of a person.

Or at least maybe you can find enough data to solve that task, whereas in contrast, the function needed to look at the hand and map that directly to the age of the child, that seems like a much more complex problem that intuitively maybe you need more data to learn if you were to apply a pure end-to-end deep learning approach.

So let me finish this video with a more complex example.

You may know that I've been spending time helping out an autonomous driving company, Drive.ai.

So I'm actually very excited about autonomous driving.

So how do you build a car that drives itself?

Well, here's one thing you could do, and this is not an end-to-end deep learning approach.

You can take as input an image of what's in front of your car, maybe radar, lighter, other sensor readings as well, but to simplify the description, let's just say you take a picture of what's in front or what's around your car.

And then to drive your car safely you need to detect other cars and you also need to detect pedestrians.

You need to detect other things, of course, but we'll just present a simplified example here.

Having figured out where are the other cars and pedestrians, you then need to plan your own route.

So in other words, if you see where are the other cars, where are the pedestrians, you need to decide how to steer your own car, what path to steer your own car for the next several seconds.

And having decided that you're going to drive a certain path, maybe this is a top down view of a road and that's your car.

Maybe

(DESCRIPTION)
A small drawing is created of a road with three lanes, and a car.

(SPEECH)
you've decided to drive that path, that's what a route is, then you need to execute this by generating the appropriate steering, as well as acceleration and braking commands.

So in going from your image or your sensory inputs to detecting cars and pedestrians, that can be done pretty well using deep learning, but then having figured out where the other cars and pedestrians are going, to select this route to exactly how you want to move your car, usually that's not to done with deep learning.

Instead that's done with a piece of software called Motion Planning.

And if you ever take a course in robotics you'll learn about motion planning.

And then having decided what's the path you want to steer your car through, there'll be some other algorithm, we're going to say it's a control algorithm that then generates the exact decision, that then decides exactly how much to turn the steering wheel and how much to step on the accelerator or step on the brake.

So I think what this example illustrates is that you want to use machine learning or use deep learning to learn some individual components and when applying supervised learning you should carefully choose what types of X to Y mappings you want to learn depending on what task you can get data for.

And in contrast, it is exciting to talk about a pure end-to-end deep learning approach where you input the image and directly output a steering.

But given data availability and the types of things we can learn with neural networks today, this is actually not the most promising approach or this is not an approach that I think teams have gotten to work best.

And I think this pure end-to-end deep learning approach is actually less promising than more sophisticated approaches like this, given the availability of data and our ability to train neural networks today.

So that's it for end-to-end deep learning.

It can sometimes work really well but you also have to be mindful of where you apply end-to-end deep learning.

Finally, thank you and congrats on making it this far with me.

If you finish last week's videos and this week's videos then I think you will already be much smarter and much more strategic and much more able to make good prioritization decisions in terms of how to move forward on your machine learning project, even compared to a lot of machine learning engineers and researchers that I see here in Silicon Valley.

So congrats on all that you've learned so far and I hope you now also take a look at this week's homework problems which should give you another opportunity to practice these ideas and make sure that you're mastering them.