CS229a - Week 11

Written by hackerwins *on* July 30, 2019

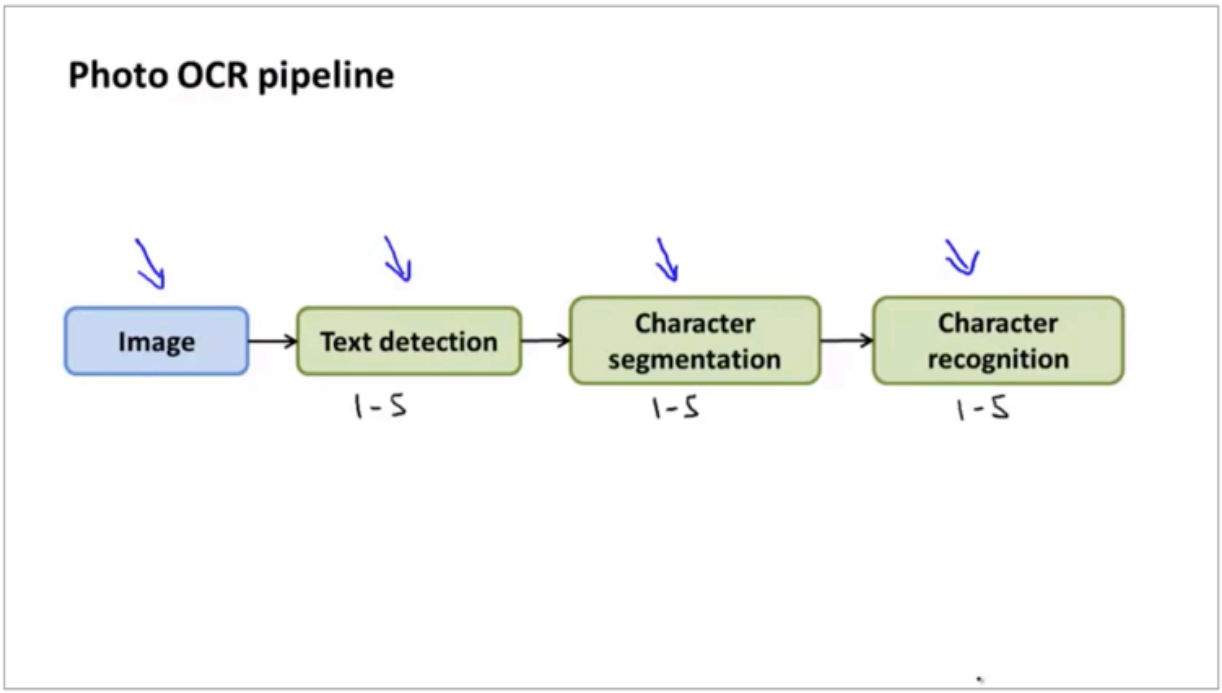
ML CS229a

Problem Description and Pipeline

The photo OCR problem focuses on how to get computers to read the text in images that we take.

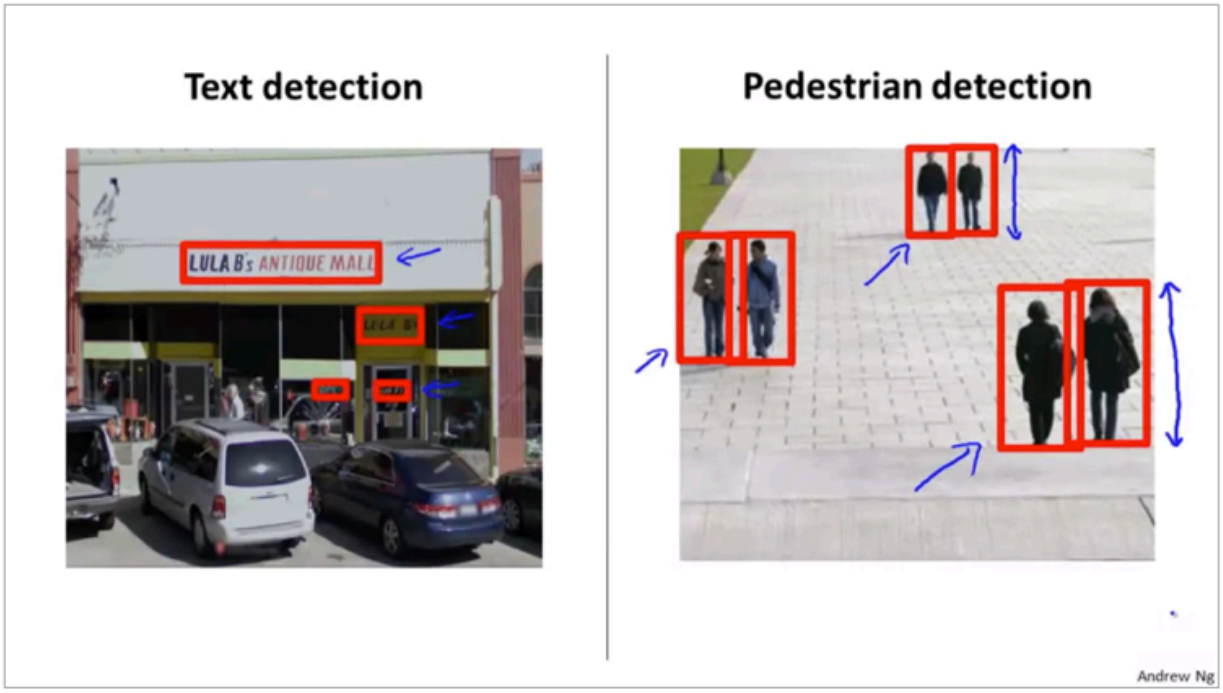
Photo OCR pipeline

1. Text detection
2. Character segmentation
3. Character classification

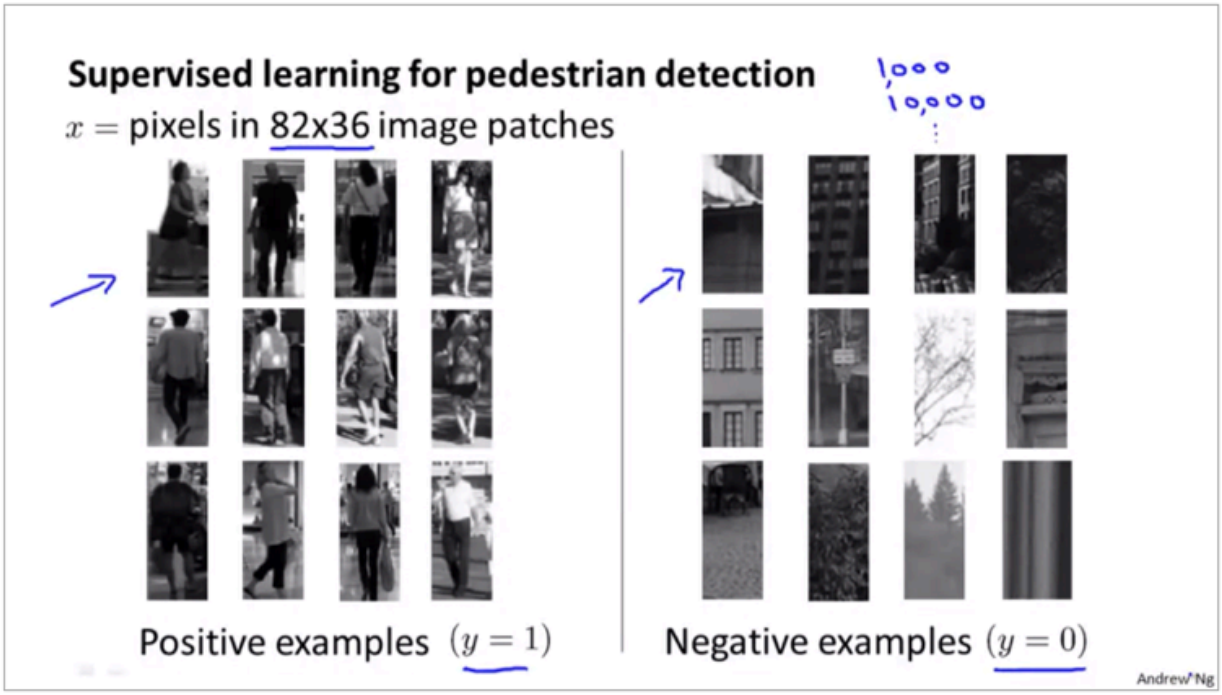


In many complex machine learning systems, these sorts of pipelines are common, where you can have multiple modules–in this example, the text detection, character segmentation, character recognition modules–each of which may be machine learning component, or sometimes it may not be a machine learning component but to have a set of modules that act one after another on some piece of data in order to produce the output you want, which in the photo OCR example is to find the transcription of the text that appeared in the image.

Sliding windows

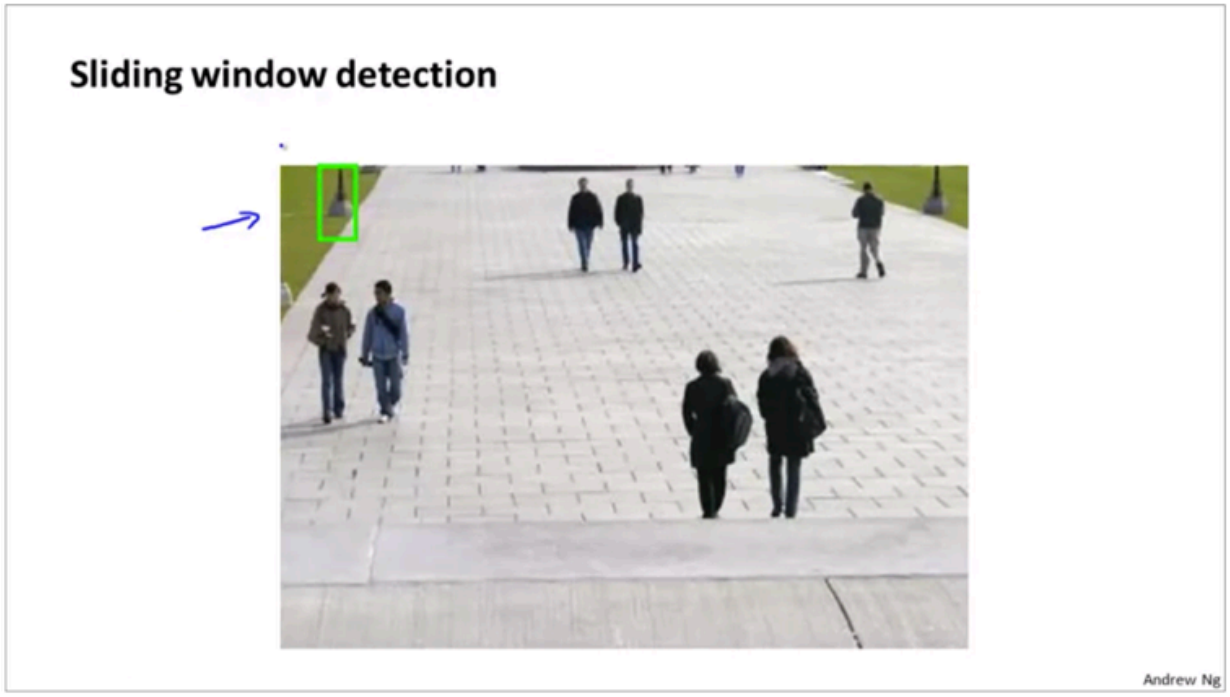


For different pedestrians but for text detection the height and width ratio is different for different lines of text Although for pedestrian detection, the pedestrians can be different distances away from the camera and so the height of these rectangles can be different depending on how far away they are. but the aspect ratio is the same.



In a more typical pedestrian detection application, we may have

anywhere from a 1,000 training examples up to maybe 10,000 training examples, or even more if you can get even larger training sets. And what you can do, is then train in your network or some other learning algorithm to take this input, an image patch of dimension 82 by 36, and to classify 'y' and to classify that image patch as either containing a pedestrian or not.

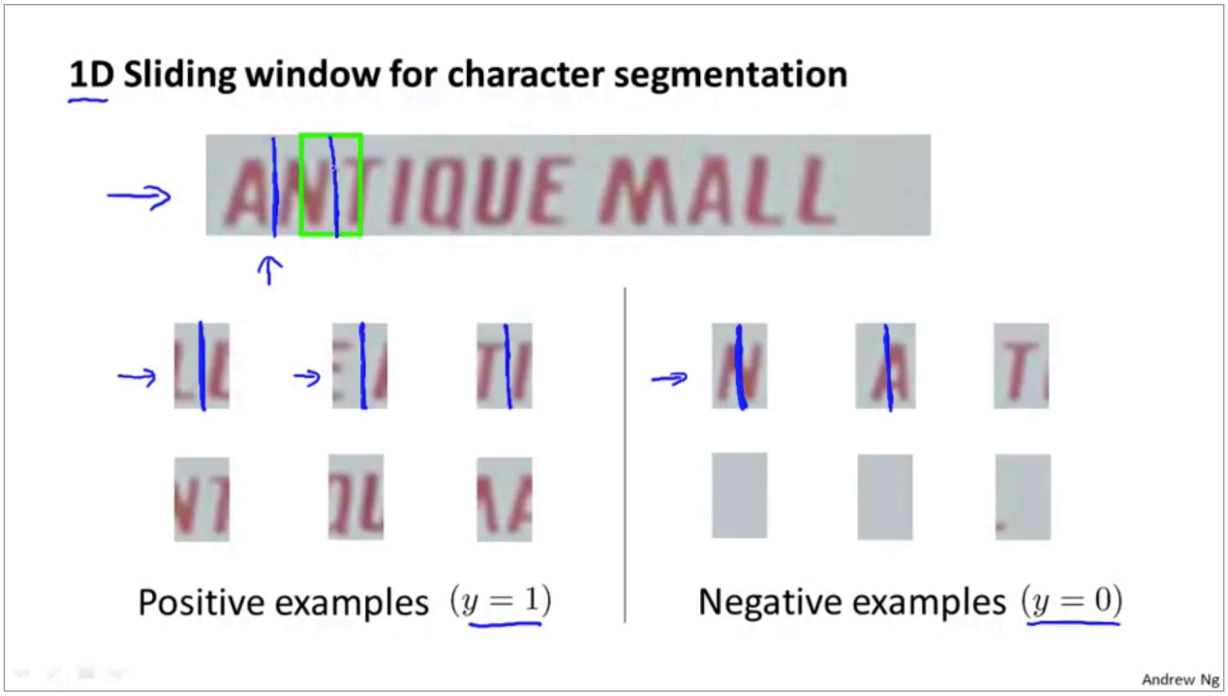


What we would do is start by taking a rectangular patch of this image. Like that shown up here, so that's maybe a 82 X 36 patch of this image, and run that image patch through our classifier to determine whether or not there is a pedestrian in that image patch, and hopefully our classifier will return y equals 0 for that patch, since there is no pedestrian.

You can do this at an even larger scales and run that side of Windows to the end And after this whole process hopefully your algorithm will detect whether theres pedestrian appears in the image.

1. Similar to pedestrian detection you can come up with a label training set with positive examples and negative examples with examples corresponding to regions where text appears.

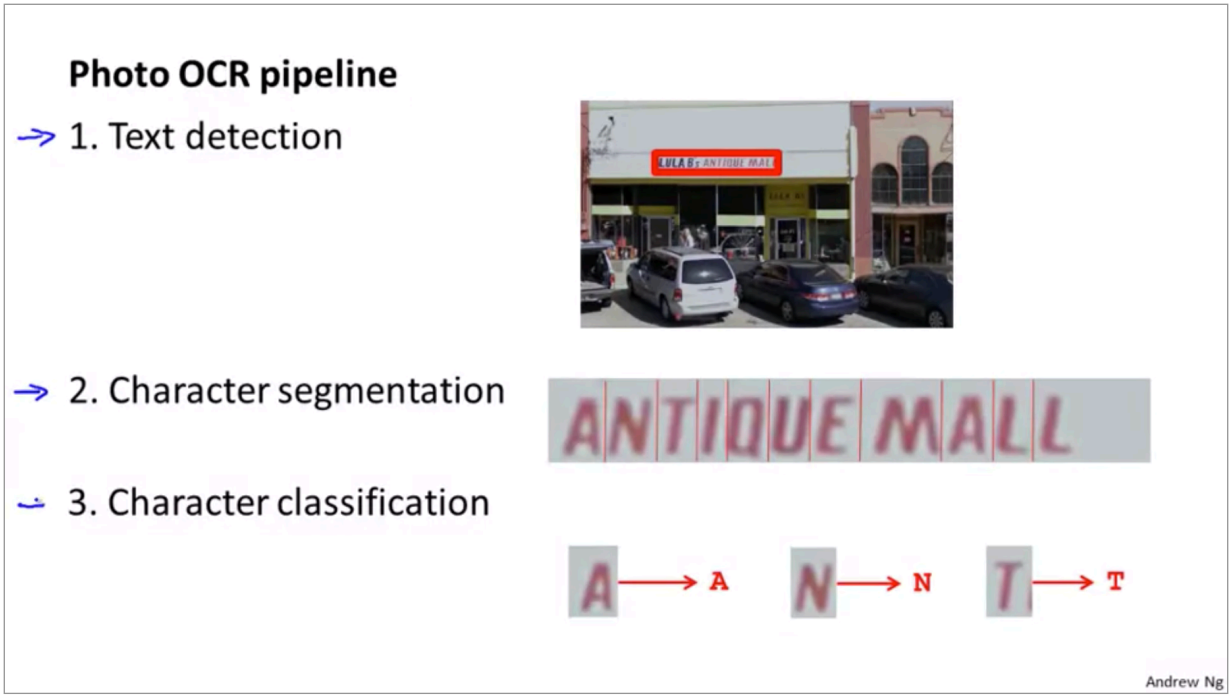
2. We are going to run a sliding windows at just one fixed scale just for purpose of illustration, meaning that I'm going to use just one rectangle size. But lets say I run my little sliding windows classifier on lots of little image patches like this
3. We aren't quite done yet because what we actually want to do is draw rectangles around all the region where this text in the image, so were going to take one more step which is we take the output of the classifier and apply to it what is called an expansion operator. If we use a simple heuristic to rule out rectangles whose aspect ratios look funny because we know that boxes around text should be much wider than they are tall.



Now, you recall that the second stage of pipeline was character segmentation, so given an image like that shown on top, how do we segment out the individual characters in this image. So for initial positive examples. This first cross example, this image patch looks like the middle of it is indeed the middle has splits between two characters and the second example again this looks like a positive example.

Having trained such a classifier, we can then run this on this sort of text that our text detection system has pulled out. We're going to slide the window only in one straight line from left to right, theres no

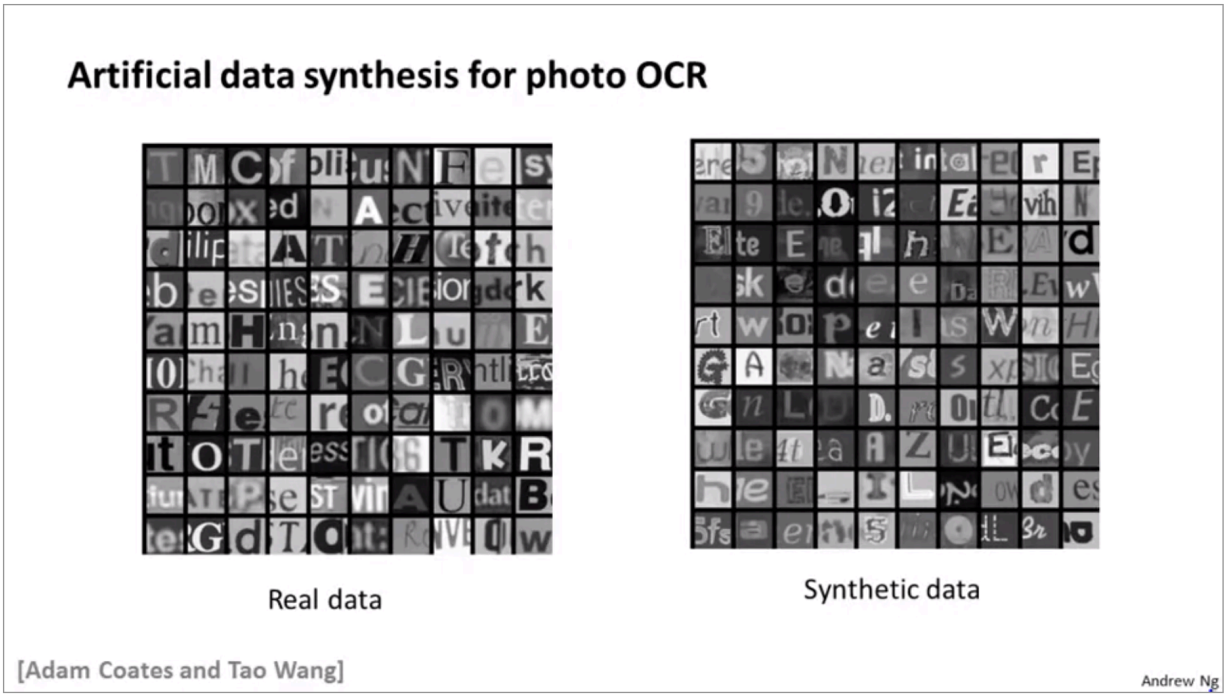
different rows here.



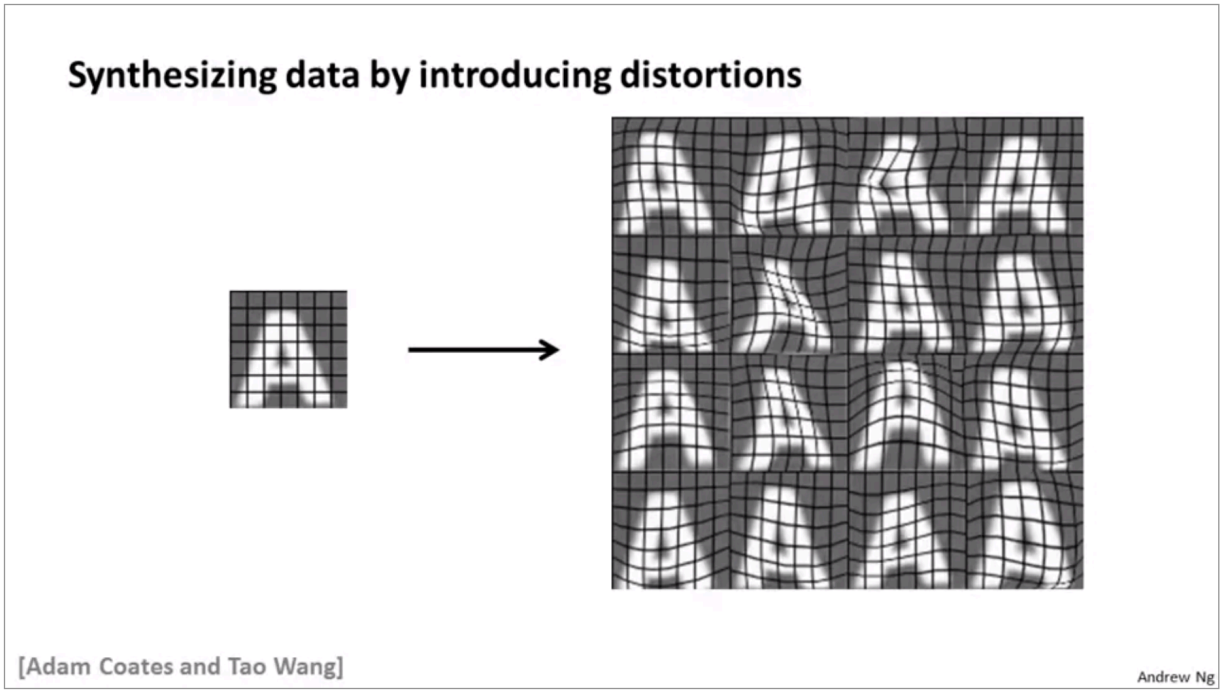
Getting lots of data: Artificial Data Synthesis

One of the most reliable ways to get a high performance machine learning system is to take a low bias learning algorithm and to train it on a massive training set. But where did you get so much training data from? Turns out that the machine earnings there's a fascinating idea called artificial data synthesis. this doesn't apply to every single problem, and to apply to a specific problem, often takes some thought and innovation and insight.

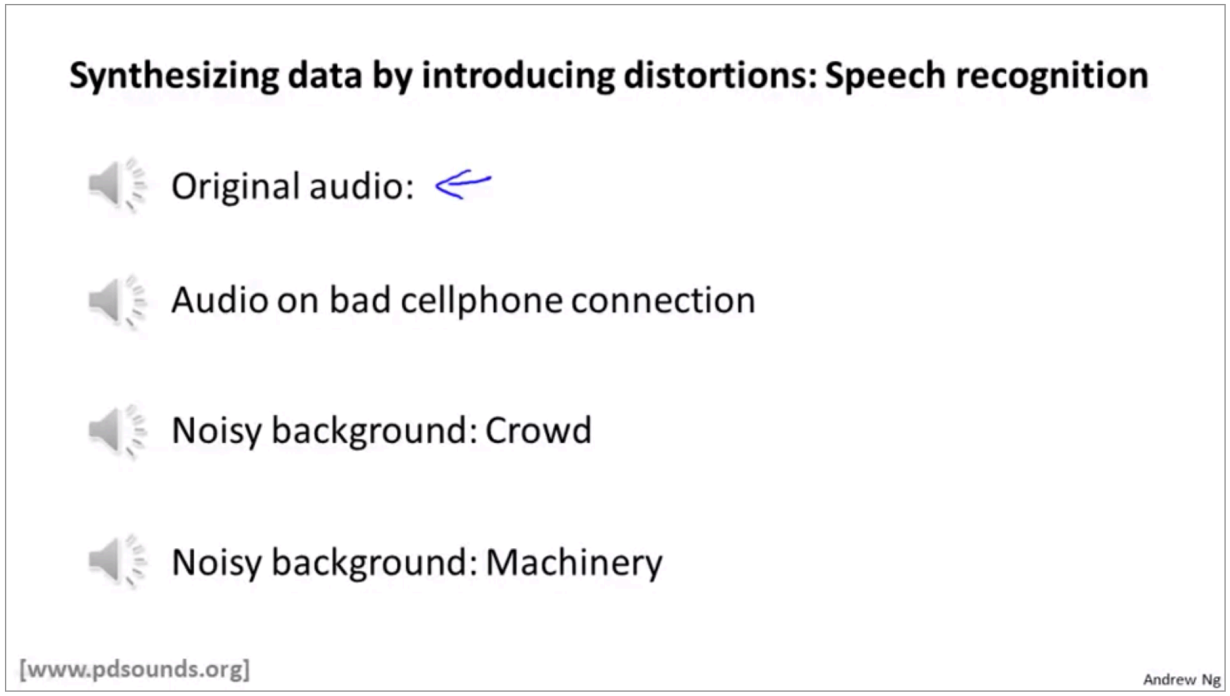
The idea of artificial data synthesis comprises of two variations, main the first is if we are essentially creating data from scratch. And the second is if we already have it's small label training set and we somehow have amplify that training set or use a small training set to turn that into a larger training set.



Modern computers often have a huge font library and if you use a word processing software, you might have all of these fonts and many, many more Already stored inside. So if you want more training examples, one thing you can do is just take characters from different fonts and paste these characters against different random backgrounds. So after some amount of work, you know this, and it is a little bit of work to synthesize realistic looking data. But after some amount of work, you can get a synthetic training set like that.

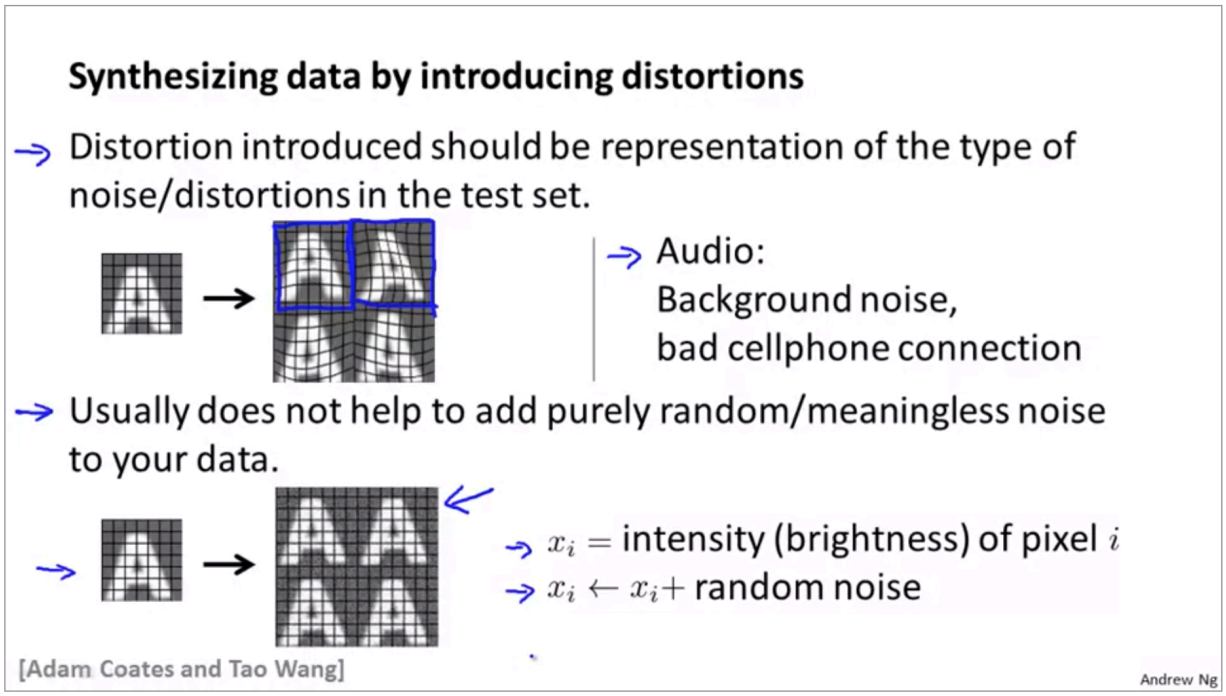


The other main approach to artificial data synthesis is that we take a real example and you create additional data, so as to amplify your training set. So what you can do is then take this alphabet here, take this image and introduce artificial distortions into the image so they can take the image a and turn that into 16 new examples.

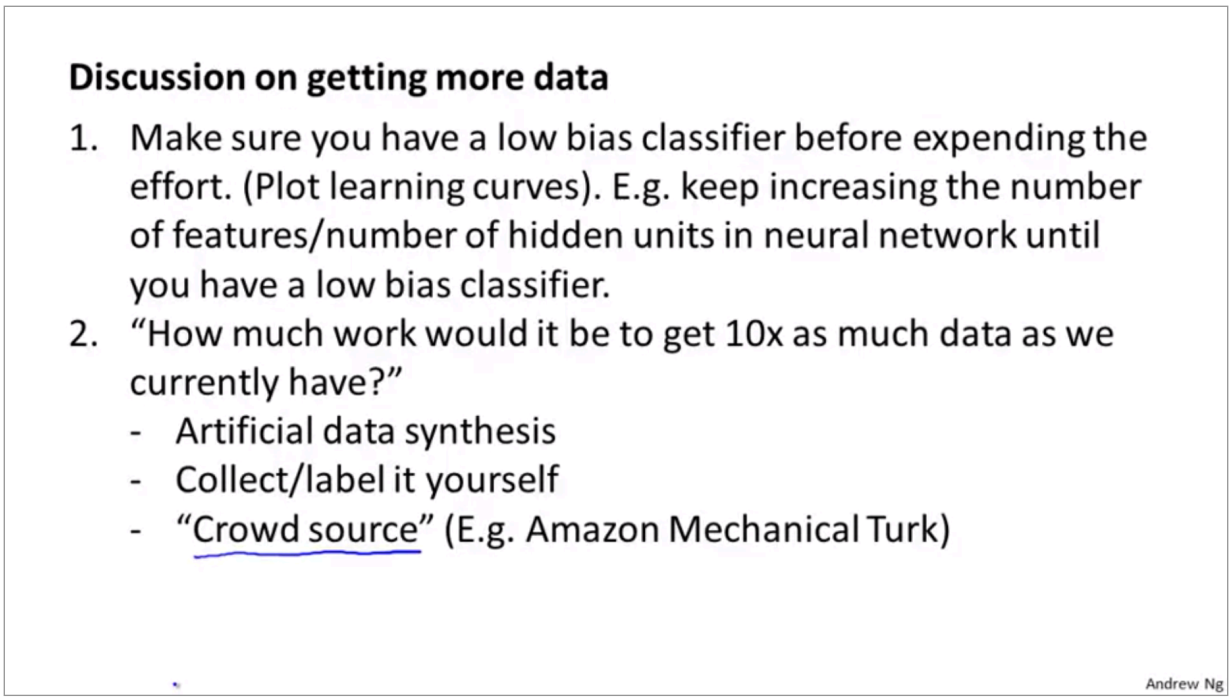


But for a different learning machine application, there may be

different the distortions that might make more sense. Let me just show one example from the totally different domain of speech recognition. So let's say you have one labeled training example, of someone saying a few specific words. So, how can we amplify the data set? Well, one thing we do is introduce additional audio distortions into the data set. So here I'm going to add background sounds to simulate a bad cell phone connection.



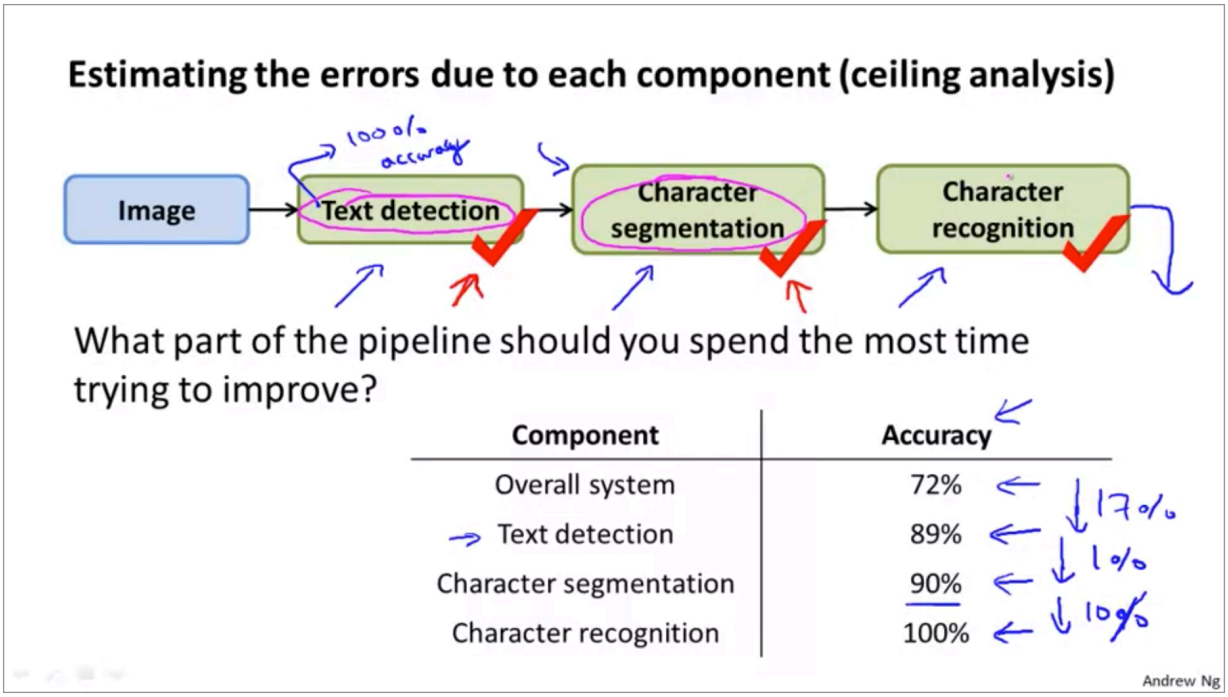
If you're trying to decide what sorts of distortions to add, you know, do think about what other meaningful distortions you might add that will cause you to generate additional training examples that are at least somewhat representative of the sorts of images you expect to see in your test sets.



You will be a hero and whatever product development, whatever team you're working on, because this can be a great way to get much better performance.

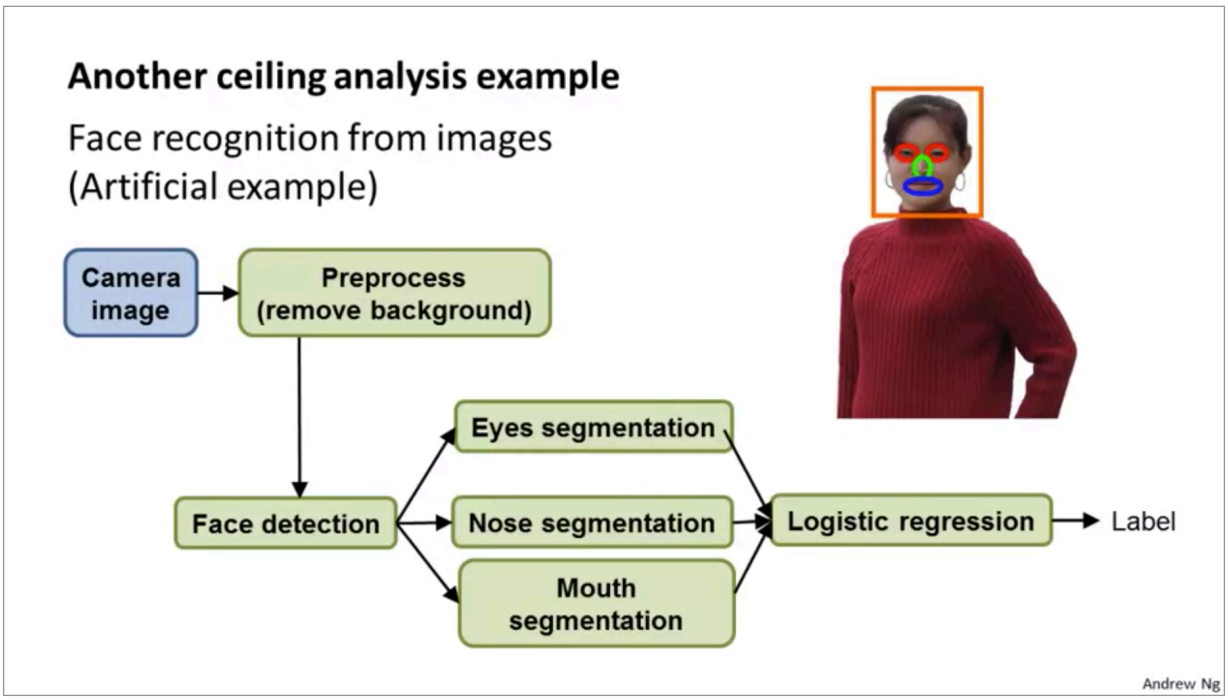
Ceiling analysis: What part of the pipeline to work on next

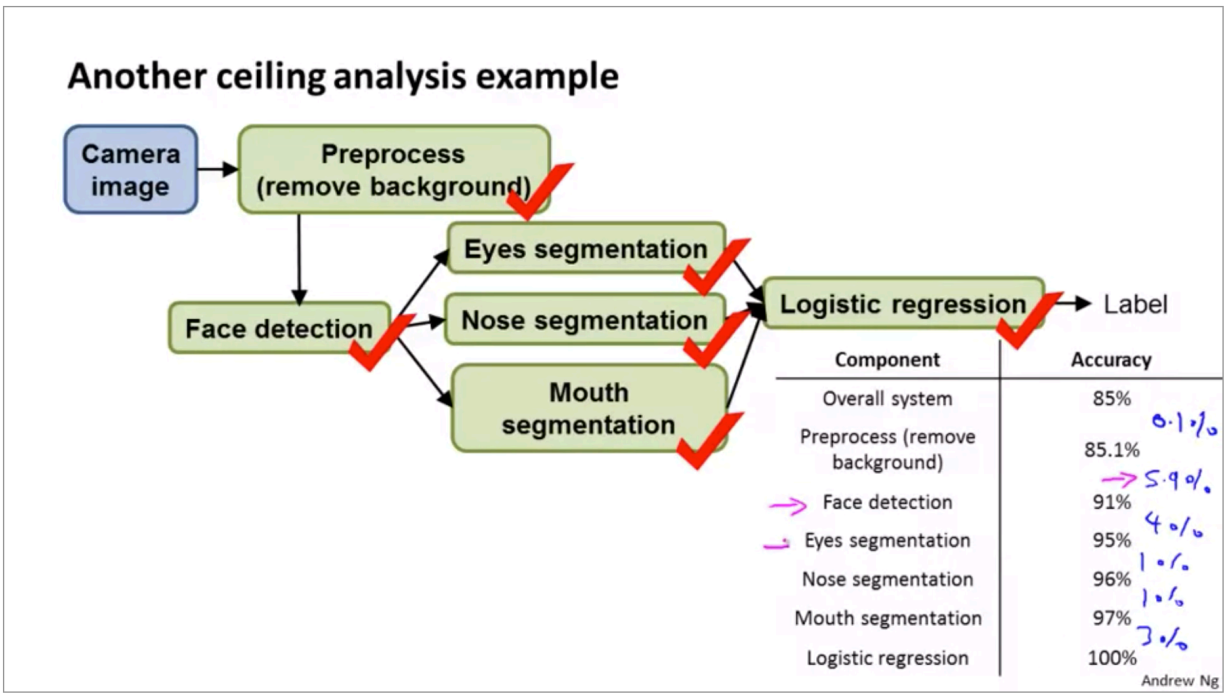
When you're the team working on the pipeline machine on your system, Ceiling analysis can sometimes give you a very strong signal, guidance on what parts of the pipeline might be the best use of your time to work on.



We can now understand what is the upside potential of improving each of these components? So we see that if we get perfect text detection, our performance went up from 72 to 89%. So that's a 17% performance gain. So this means that if we take our current system we spend a lot of time improving text detection, that means that we could potentially improve our system's performance by 17%.

Another way of thinking about this, is that by going through these sort of analysis you're trying to think about what is the upside potential of improving each of these components or how much could you possibly gain if one of these components became absolutely perfect.





The reason I put this is in this in preprocessing background removal is because I actually know of a true story where there was a research team that actually literally had to people spend about a year and a half, spend 18 months working on better background removal. But actually I'm obscuring the details for obvious reasons, but there was a computer vision application where there's a team of two engineers that literally spent about a year and a half working on better

background removal, actually worked out really complicated algorithms and ended up publishing one research paper. But after all that work they found that it just did not make huge difference to the overall performance of the actual application they were working on and if only someone were to do ceiling analysis before hand maybe they could have realized.

Summary and Thank You

Supervised Learning

- Linear regression, logistic regression, neural networks, SVMs

Unsupervised Learning

- K-means, PCA, Anomaly detection

Special applications/special topics

- Recommender systems, large scale machine learning

Advice on building a machine learning system

- Bias/variance, regularization; deciding what to work on next: evaluation of learning algorithms, learning curves, error analysis, ceiling analysis

← Top →