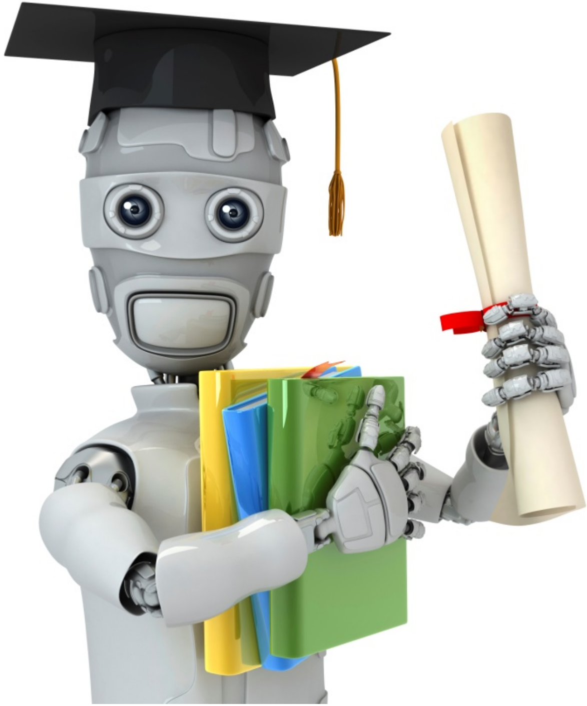


Machine Learning

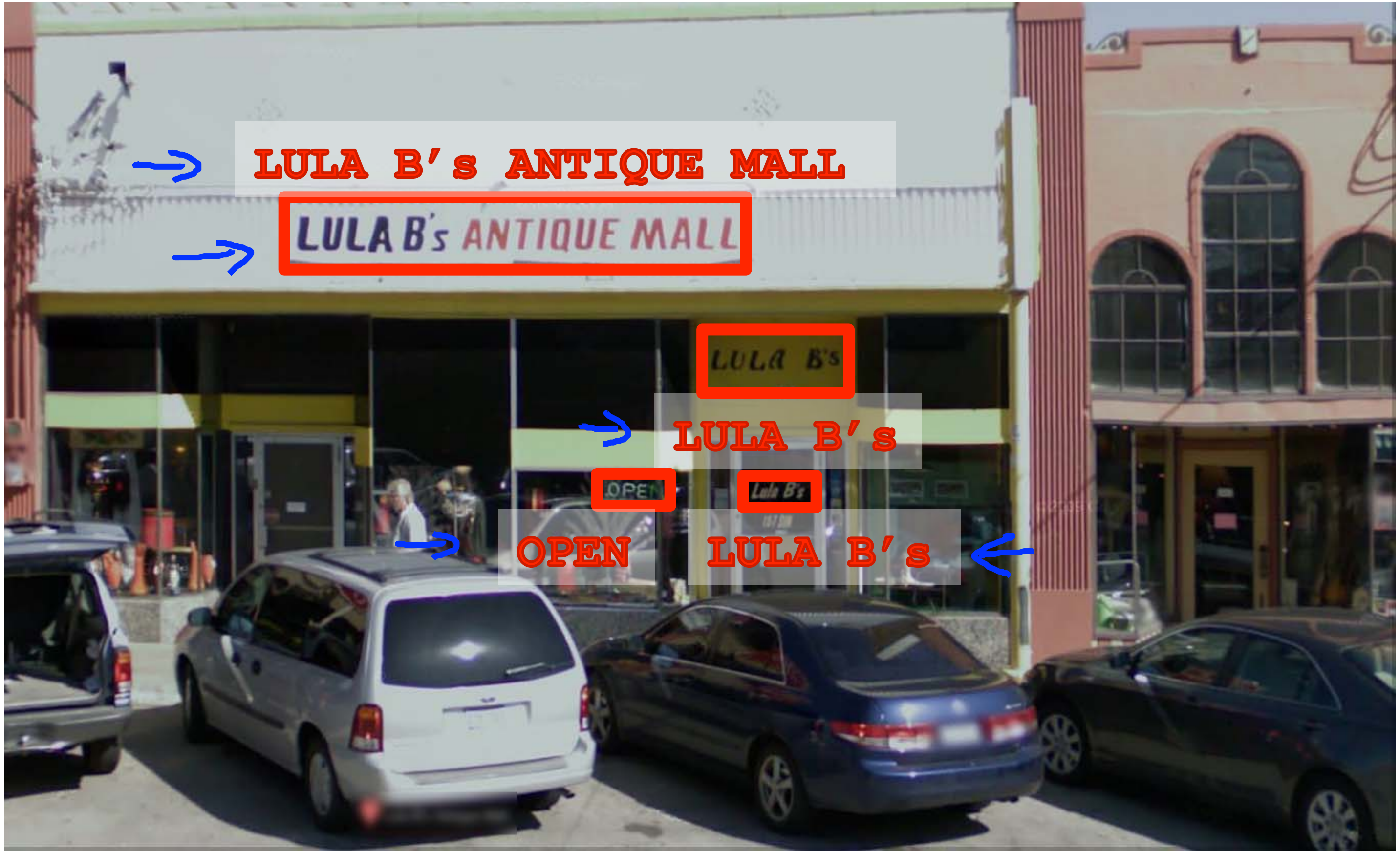
Application example: Photo OCR

Problem description and pipeline

The Photo OCR problem

Photo Optical Character Recognition.

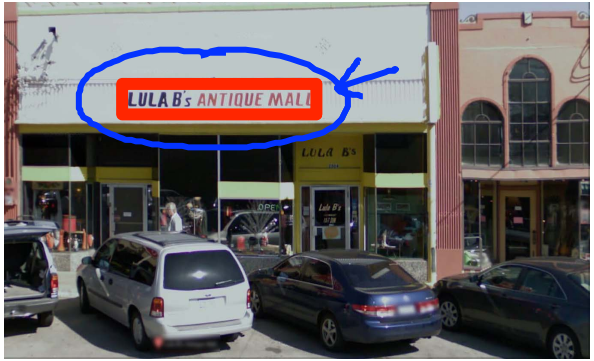
detect where is the text and then read the text in this region



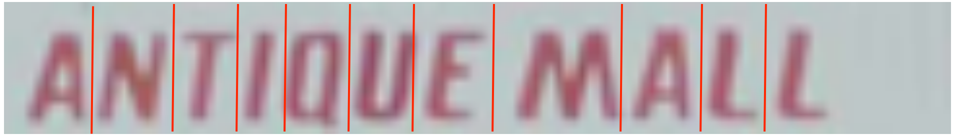
Andrew Ng

Photo OCR pipeline

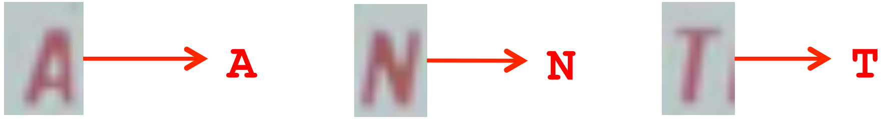
1. Text detection



2. Character segmentation



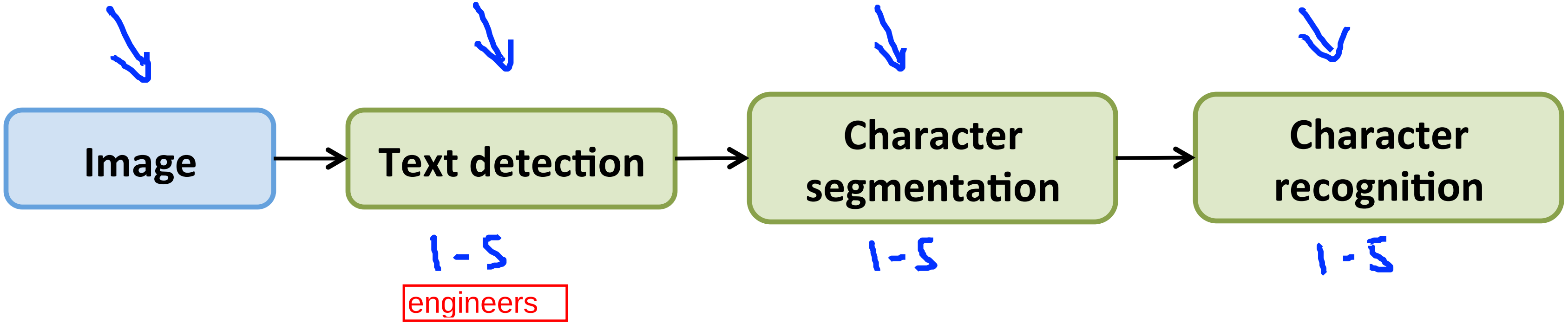
3. Character classification

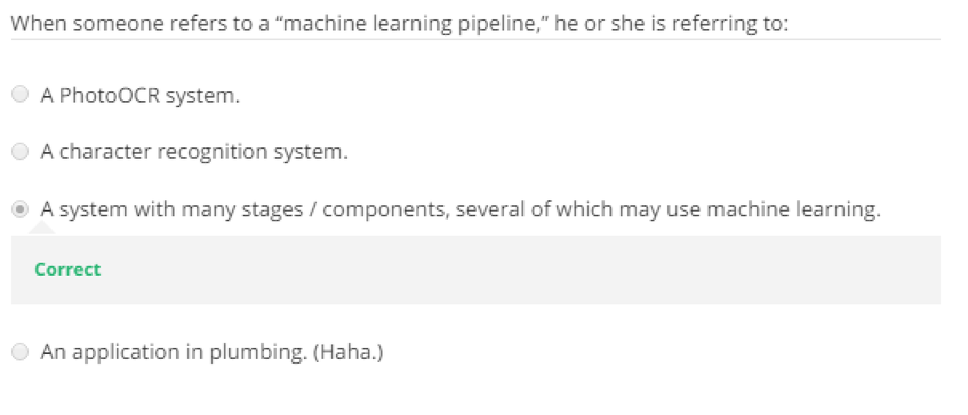


there are some photo OCR systems that do even more complex things, like spelling correction at the end.

C1eaning → Cleaning

Photo OCR pipeline





When someone refers to a “machine learning pipeline,” he or she is referring to:

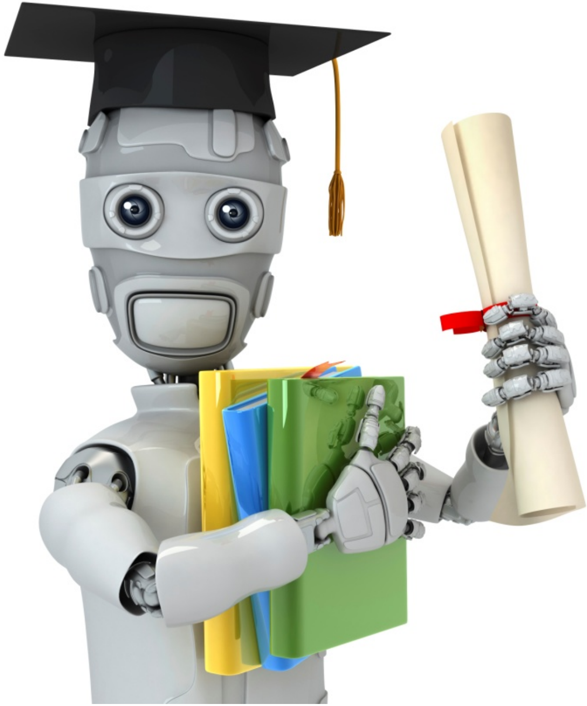
○ A PhotoOCR system.

○ A character recognition system.

● A system with many stages / components, several of which may use machine learning.

Correct

○ An application in plumbing. (Haha.)

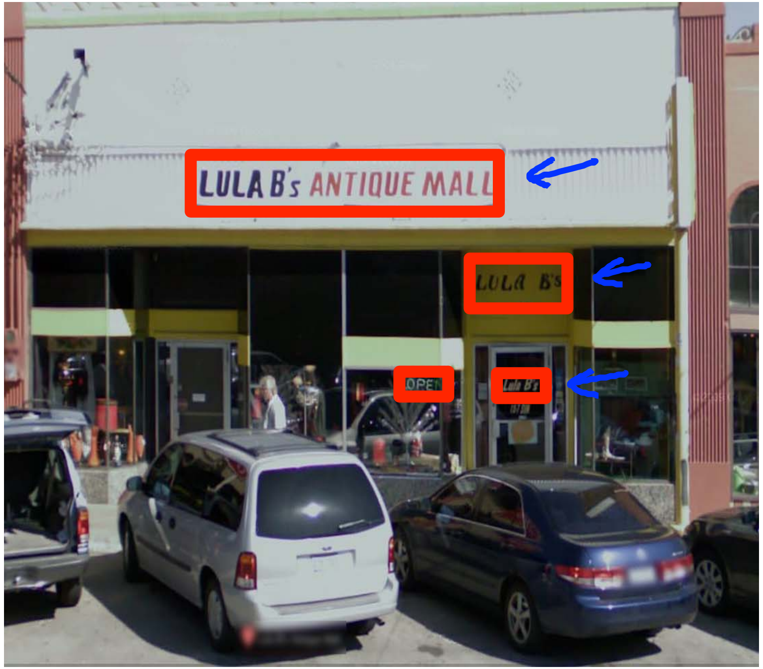


Machine Learning

Application example: Photo OCR

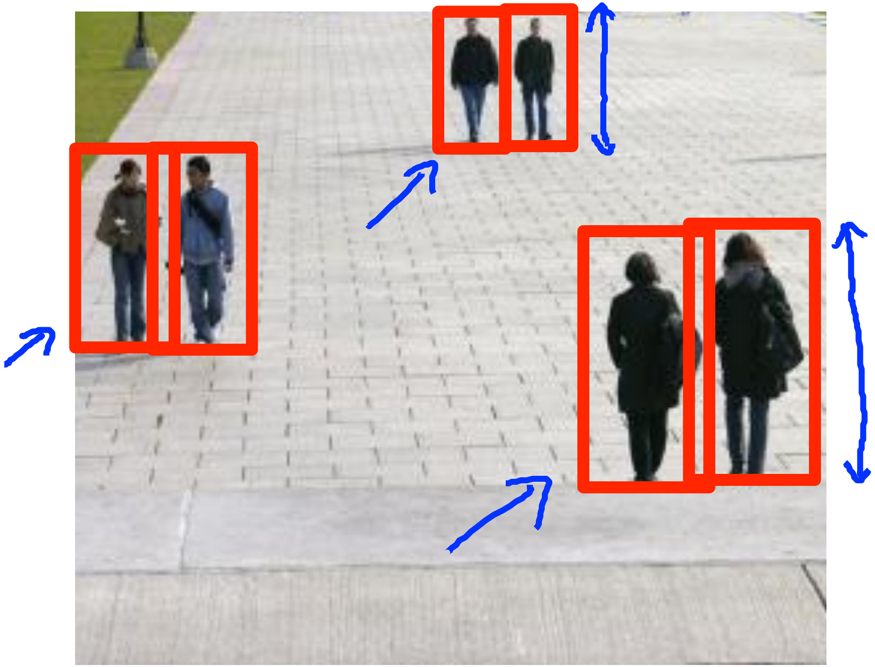
Sliding windows

Text detection



depending on the length of the text you're trying to find, these rectangles that you're trying to find can have different aspect ratios

Pedestrian detection



can be easier because the aspect ratio is about the same

Andrew Ng

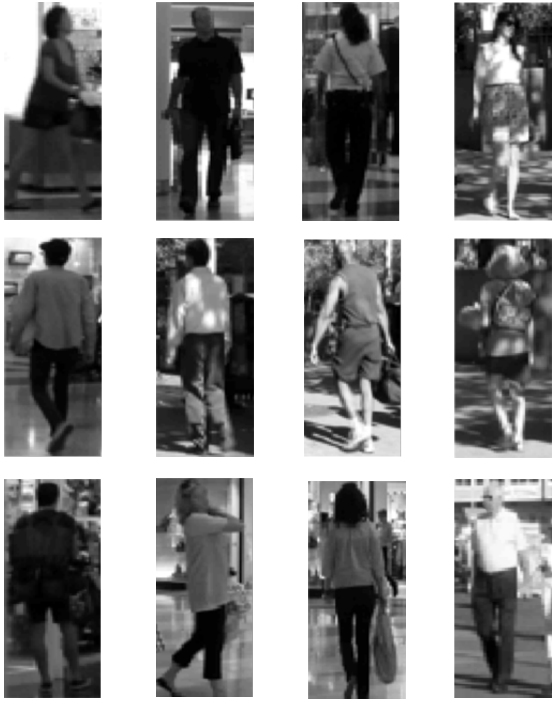
Supervised learning for pedestrian detection

1,000

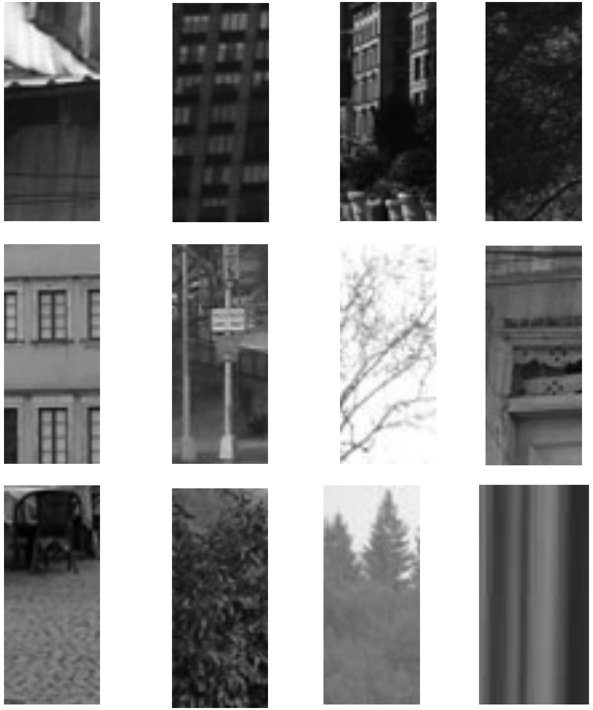
10,000

⋮

$x =$ pixels in 82x36 image patches



Positive examples $(y = 1)$

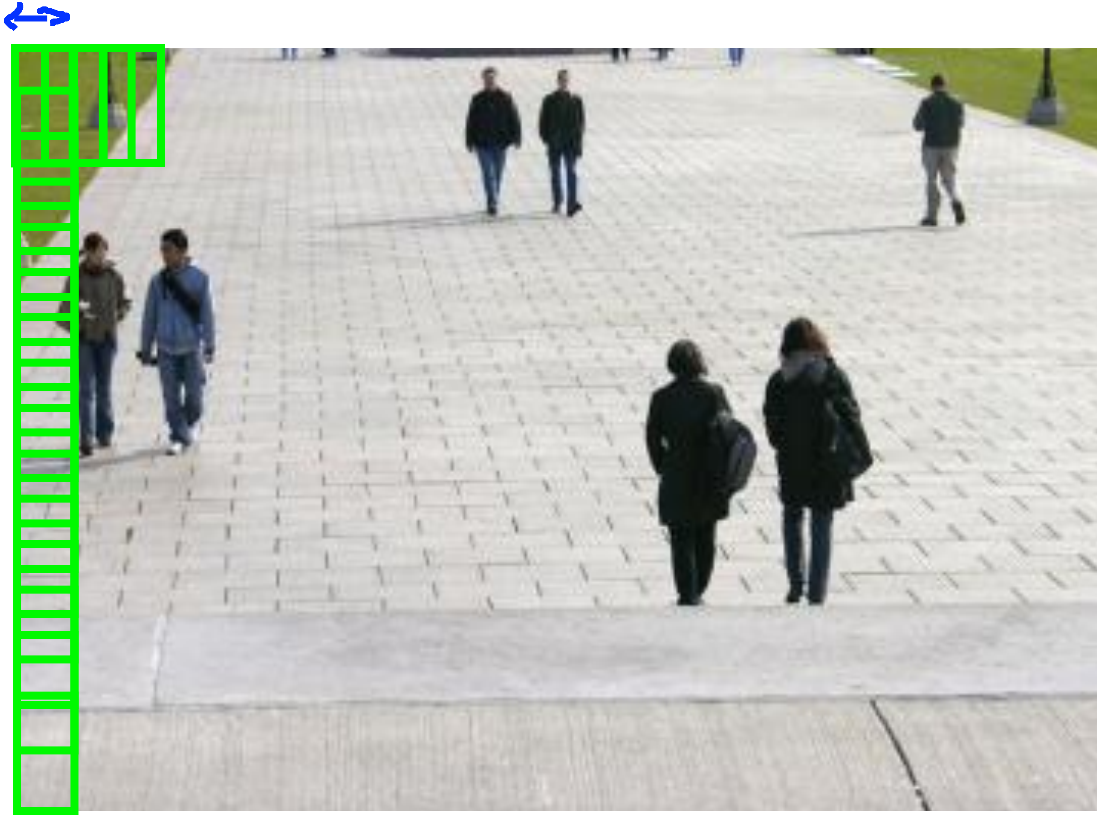


Negative examples $(y = 0)$

Sliding window detection

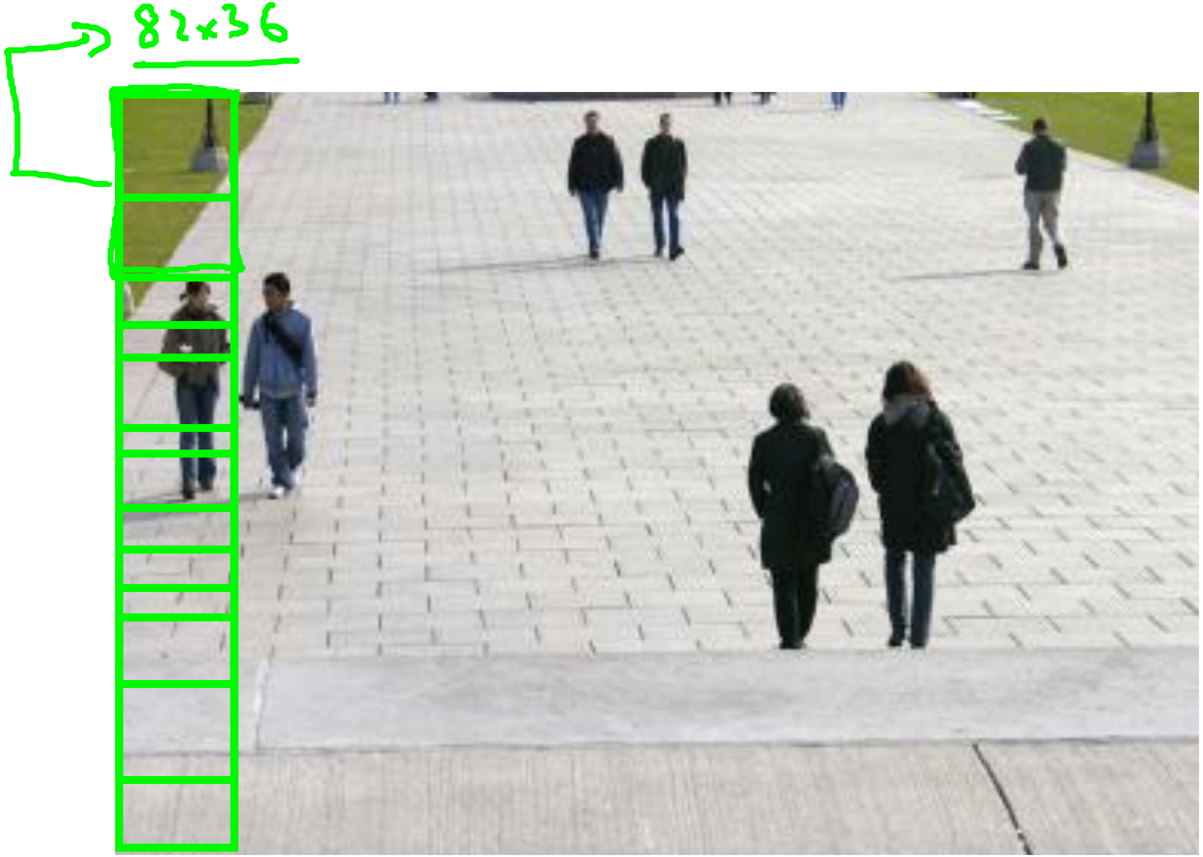
taking a rectangular patch of this image, so that's maybe a 82 X 36 patch of this image, and run that image patch through our classifier to determine whether or not there is a pedestrian in that image patch, and hopefully our classifier will return y equals 0 for that patch, since there is no pedestrian.

step-size /stride

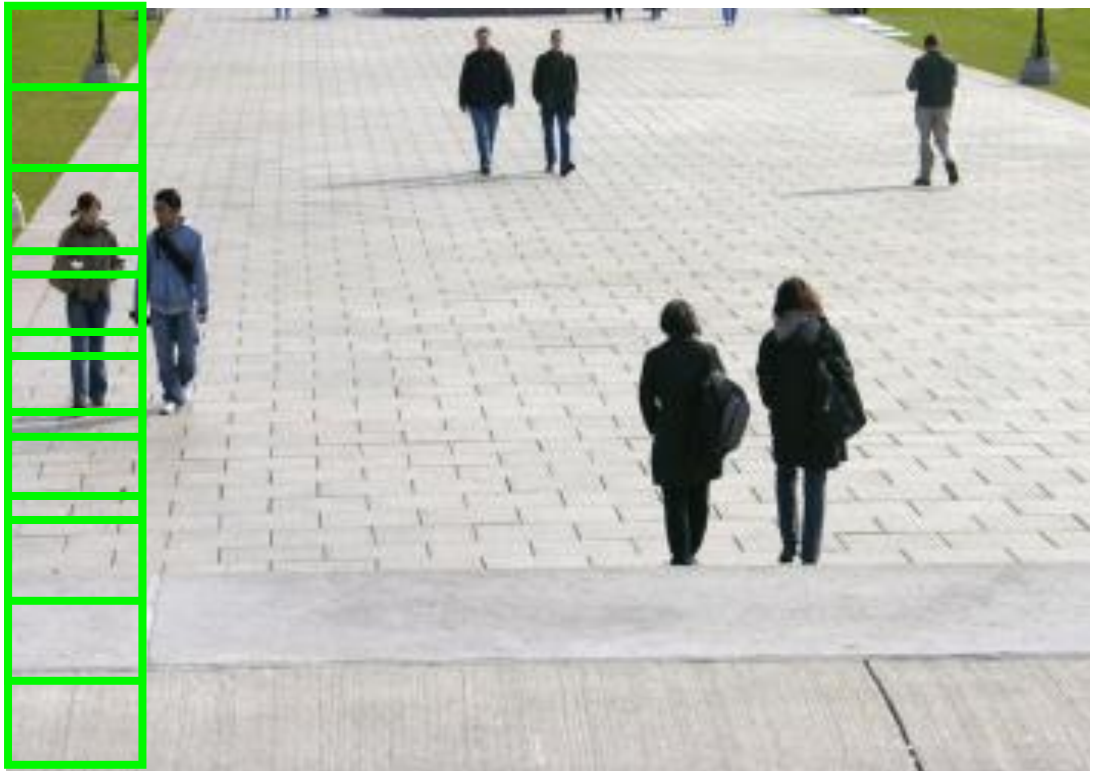


Sliding window detection

What we do next is start to look at larger image patches.

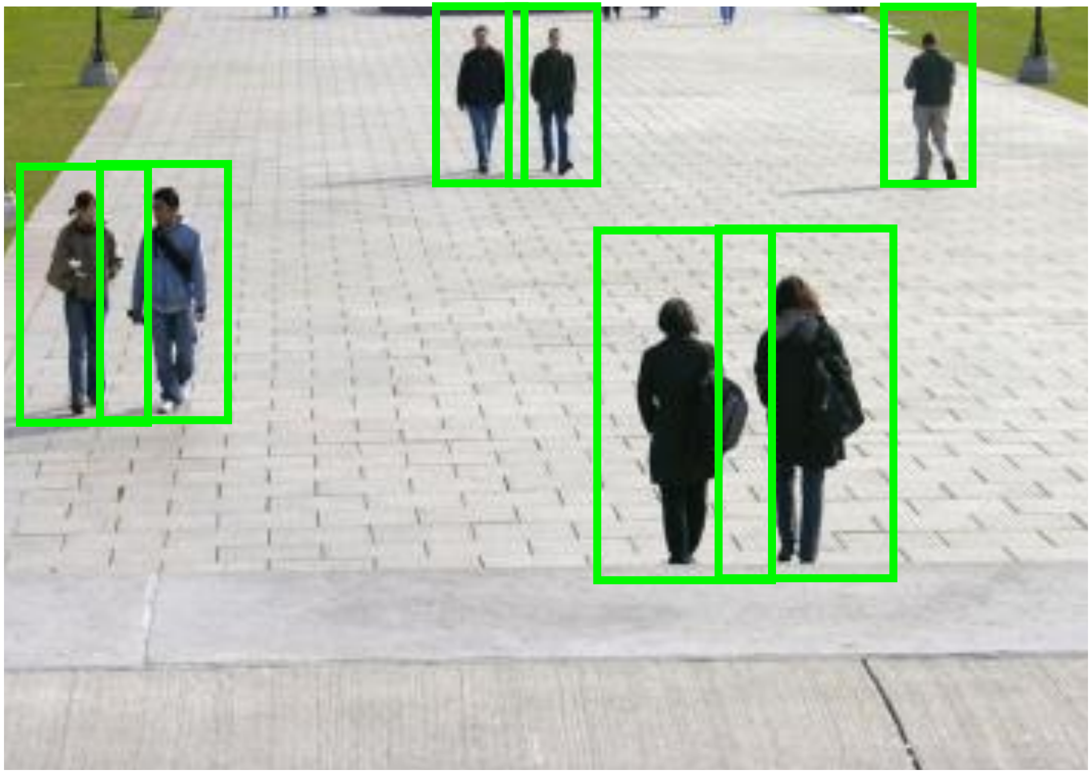


Sliding window detection

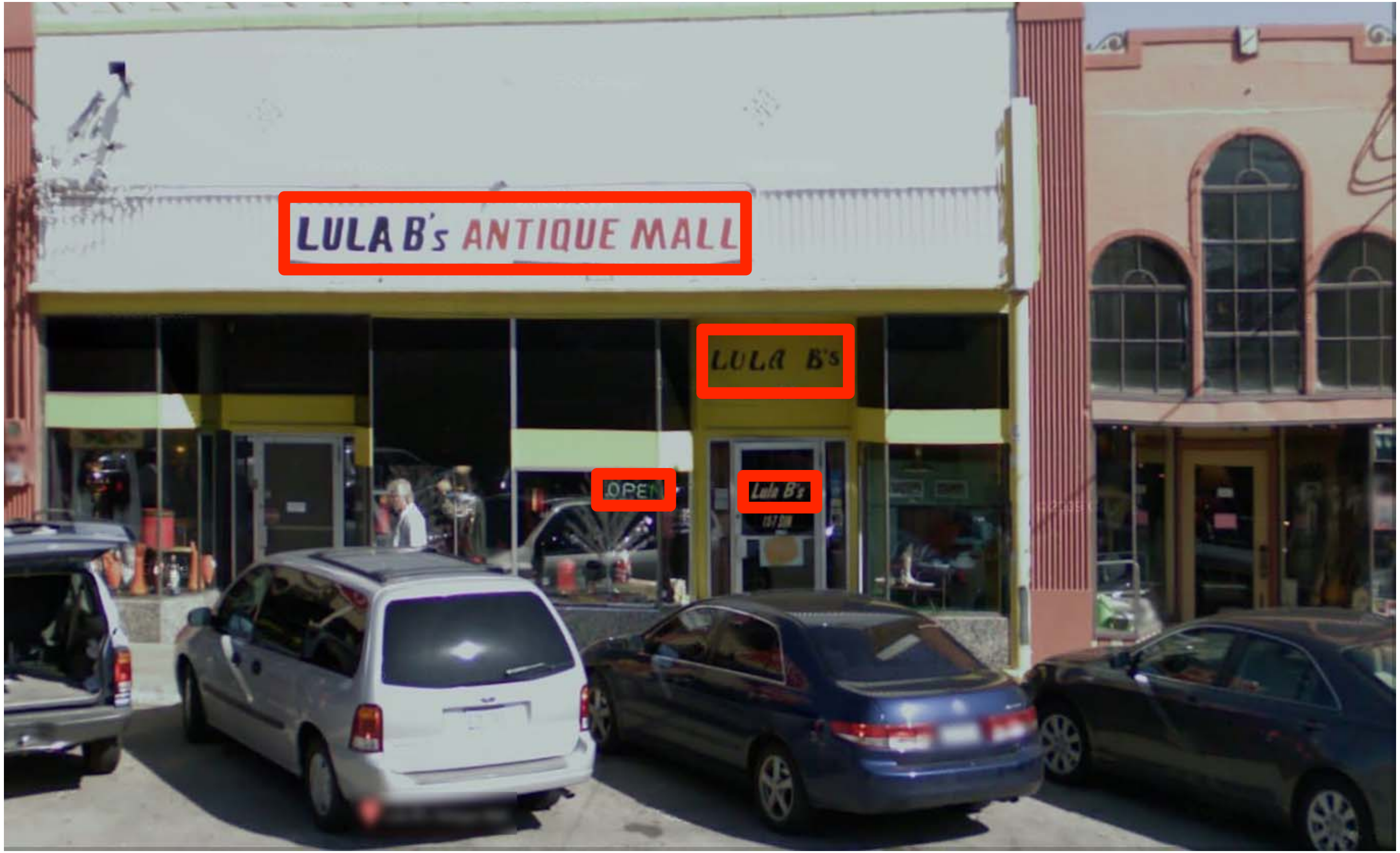


Andrew Ng

Sliding window detection

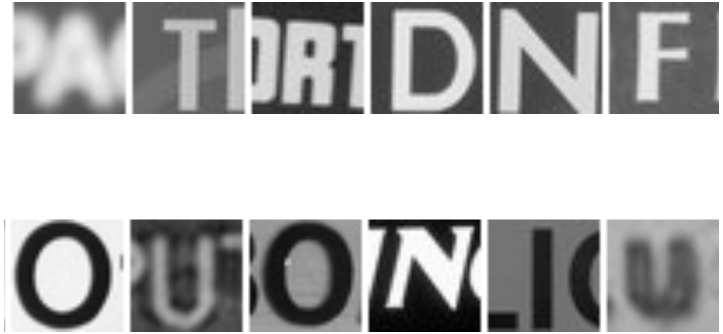


Text detection

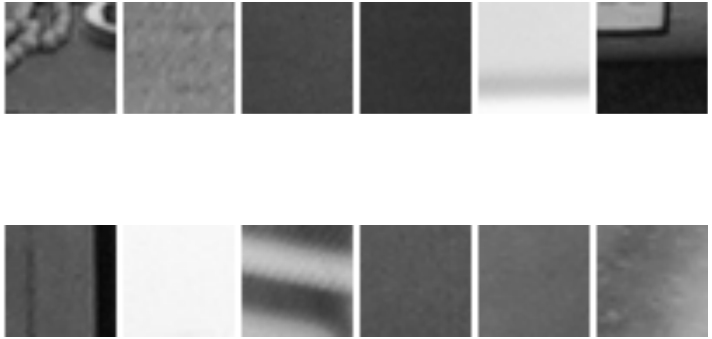


Text detection

positive examples are going to be patches of images where there is text. And negative examples is going to be patches of images where there isn't text.



Positive examples $(\underline{y = 1})$

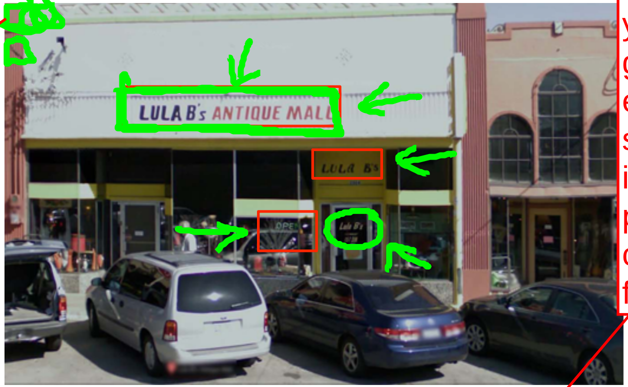


Negative examples $(\underline{y = 0})$

what we want to do is draw rectangles around all the region where this text in the image

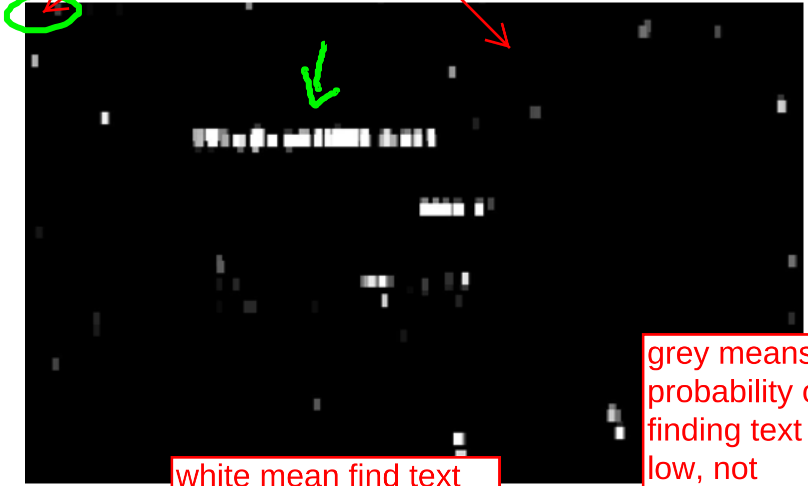
Text detection

smaller window



take one more step : expansion operator: it take the image, and it takes each of the white blobs, and expands that white region. if you look at the image on the right, what we're going to create the image on the right is, for every pixel we are going to ask, is it within some distance of a white pixel in the left image. if a specific pixel is within, say, five pixels in the leftmost image, then we'll also color that pixel white in the rightmost image. finally, draw box around the write region.

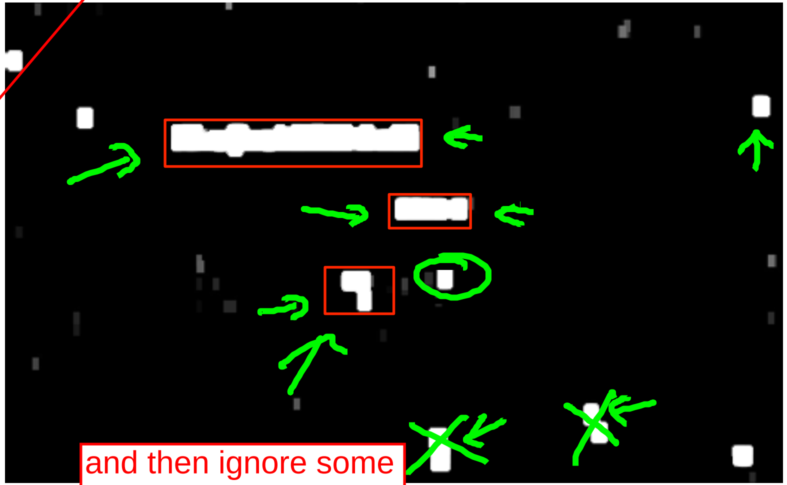
black means not find the text



grey means probability of finding text is low, not confident

white mean find text (high probability)

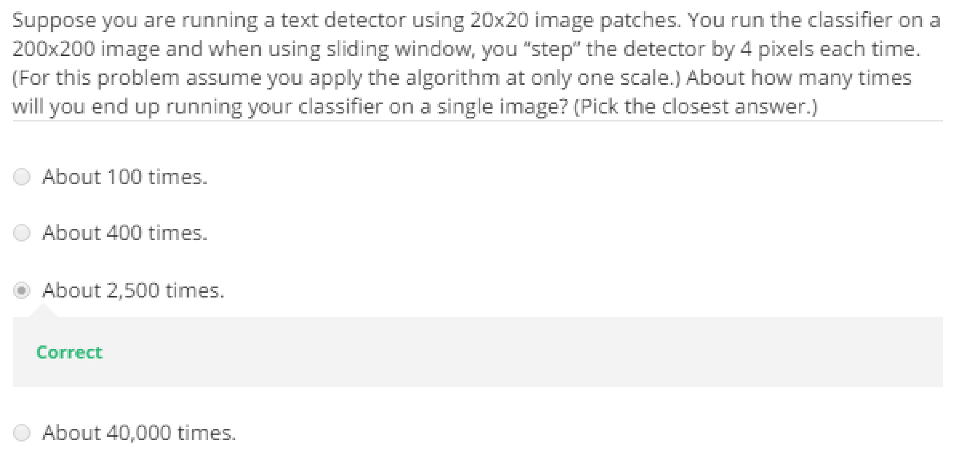
2



and then ignore some thin and tall box

[David Wu]

Andrew Ng



Suppose you are running a text detector using 20x20 image patches. You run the classifier on a 200x200 image and when using sliding window, you "step" the detector by 4 pixels each time. (For this problem assume you apply the algorithm at only one scale.) About how many times will you end up running your classifier on a single image? (Pick the closest answer.)

○ About 100 times.

○ About 400 times.

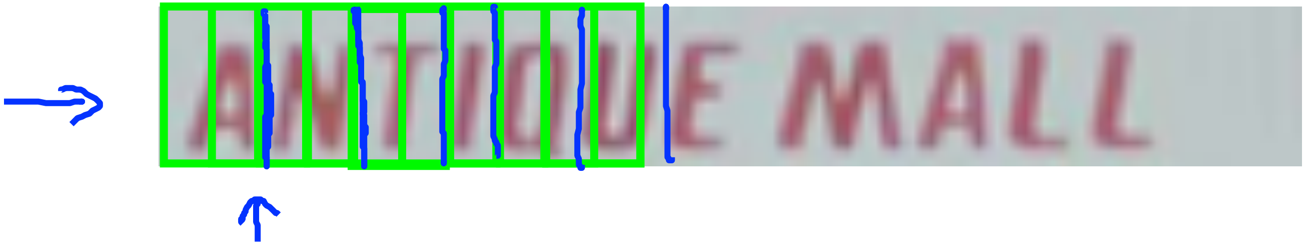
● About 2,500 times.

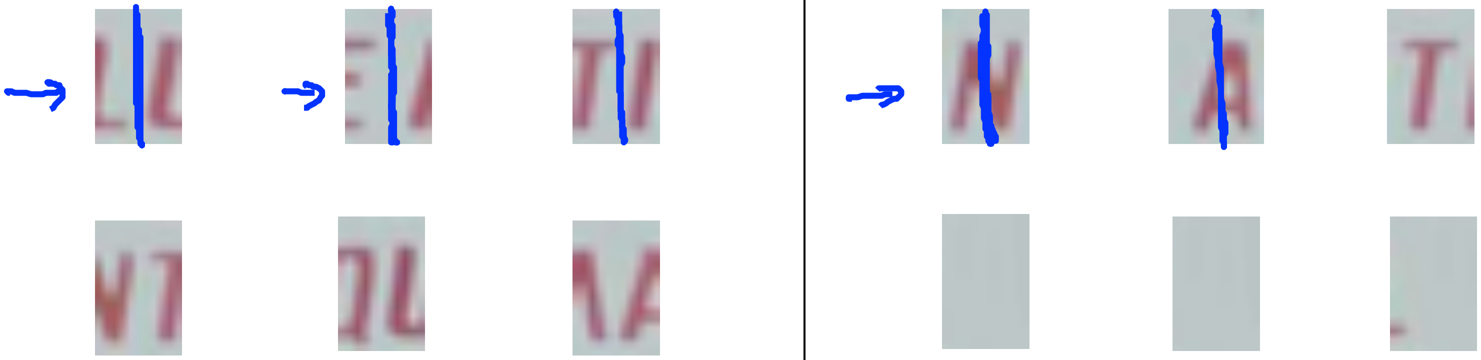
Correct

○ About 40,000 times.

1D Sliding window for character segmentation

we will train a classifier, maybe using new network, maybe using a different learning algorithm, to try to classify between the positive and negative examples.





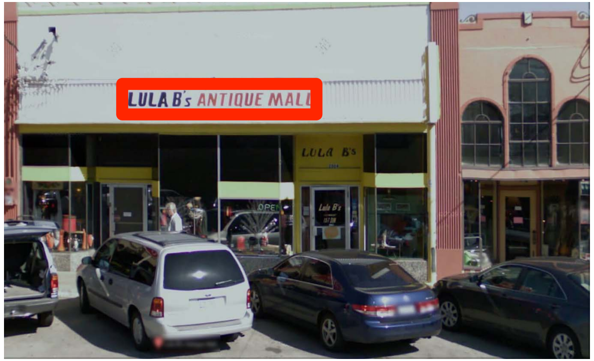
Positive examples $(y = 1)$

Negative examples $(y = 0)$

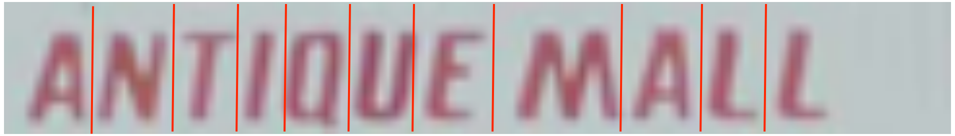
Andrew Ng

Photo OCR pipeline

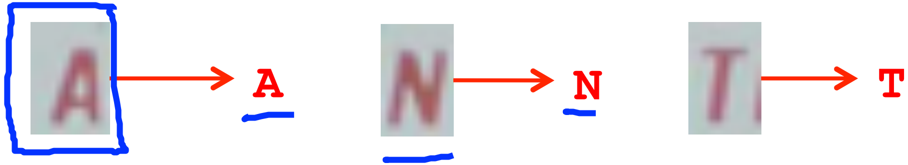
→ 1. Text detection



→ 2. Character segmentation



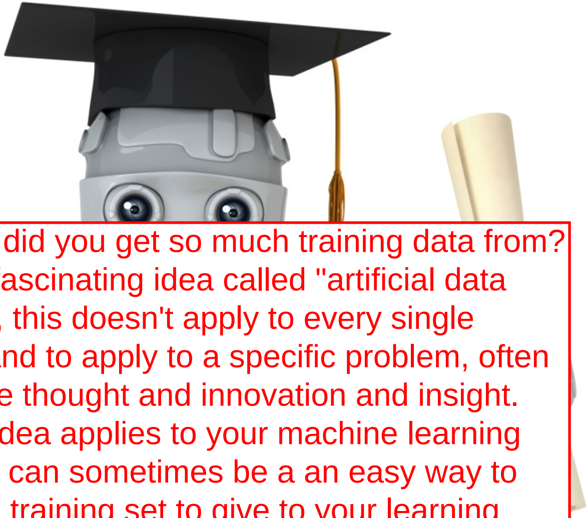
→ 3. Character classification



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.

Application example: Photo OCR

Getting lots of data: Artificial data synthesis

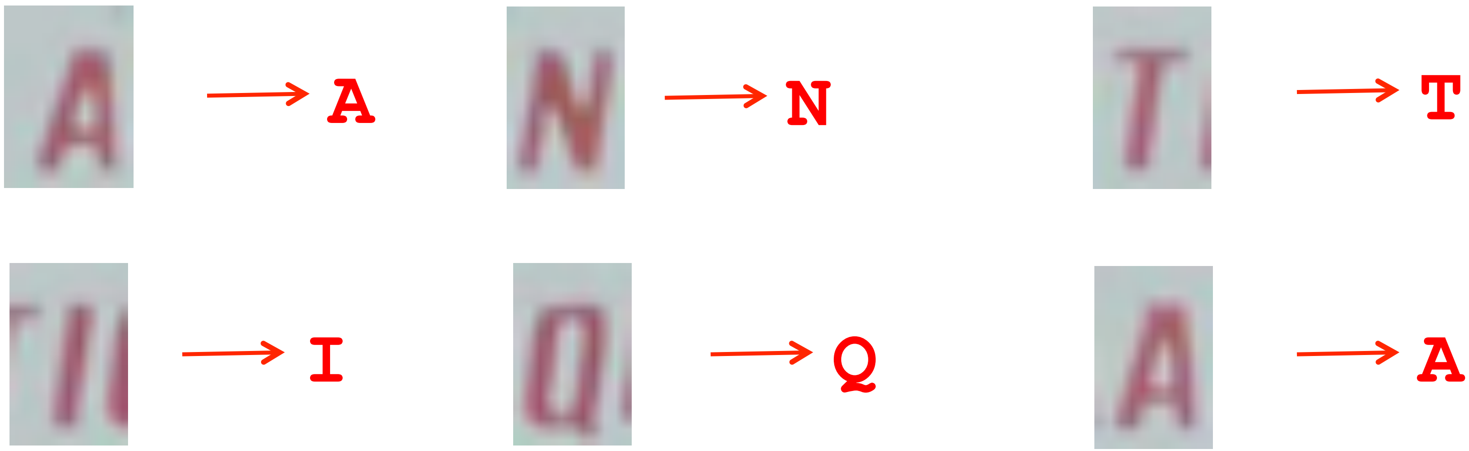


But where did you get so much training data from? There's a fascinating idea called "artificial data synthesis", this doesn't apply to every single problem, and to apply to a specific problem, often takes some thought and innovation and insight. But if this idea applies to your machine learning problem, it can sometimes be a an easy way to get a huge training set to give to your learning algorithm.

The idea of artificial data synthesis comprises of two variations (1) if we are essentially creating data from scratch. (2) 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.

use this as example

Character recognition



A

N

T

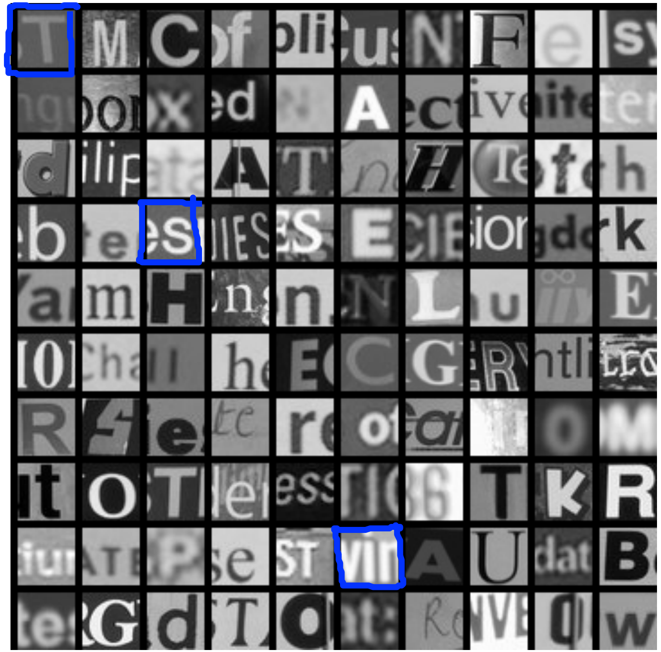
I

Q

A

Artificial data synthesis for photo OCR

I'm going to treat these images as grey scale images, rather than color images. It turns out that using color doesn't seem to help that much for this particular problem.



Real data

Abcdefg

Abcdefg

Abcdefg

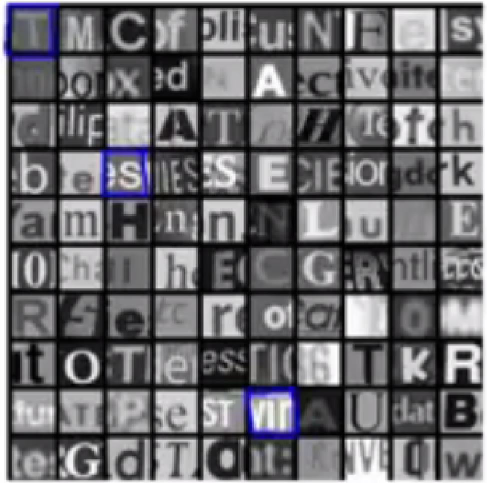
Abcdefg

Abcdefg

[Adam Coates and Tao Wang]

Artificial data synthesis for photo OCR

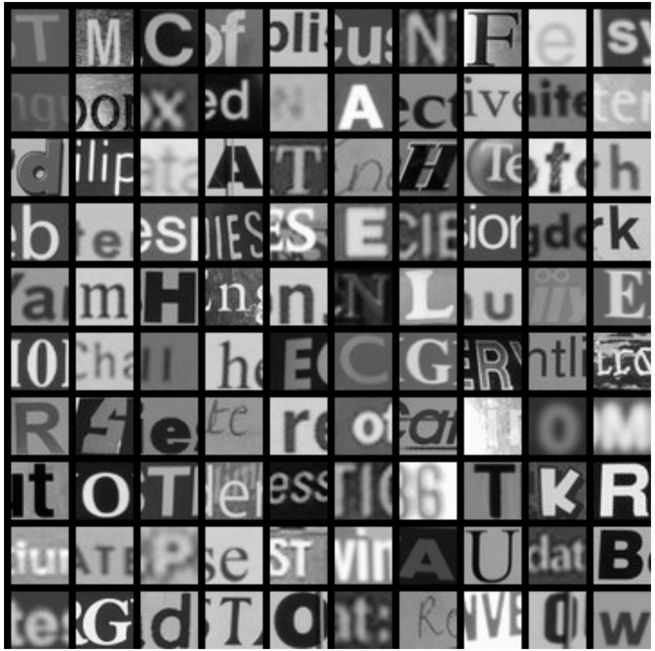
use the text that has been recognized and apply different fonts



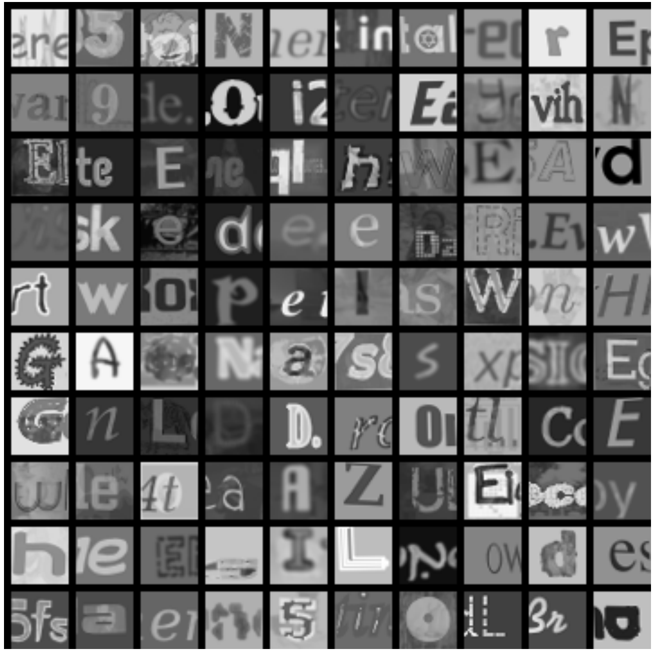
Real data

Abcdefg
Abcdefg
Abcdefg
Abcdefg
Abcdefg

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 you might take this ---- and paste that c against a random background. If you do that you now have a training example of an image of the character C. So after some amount of work, you can get a synthetic training set like that.



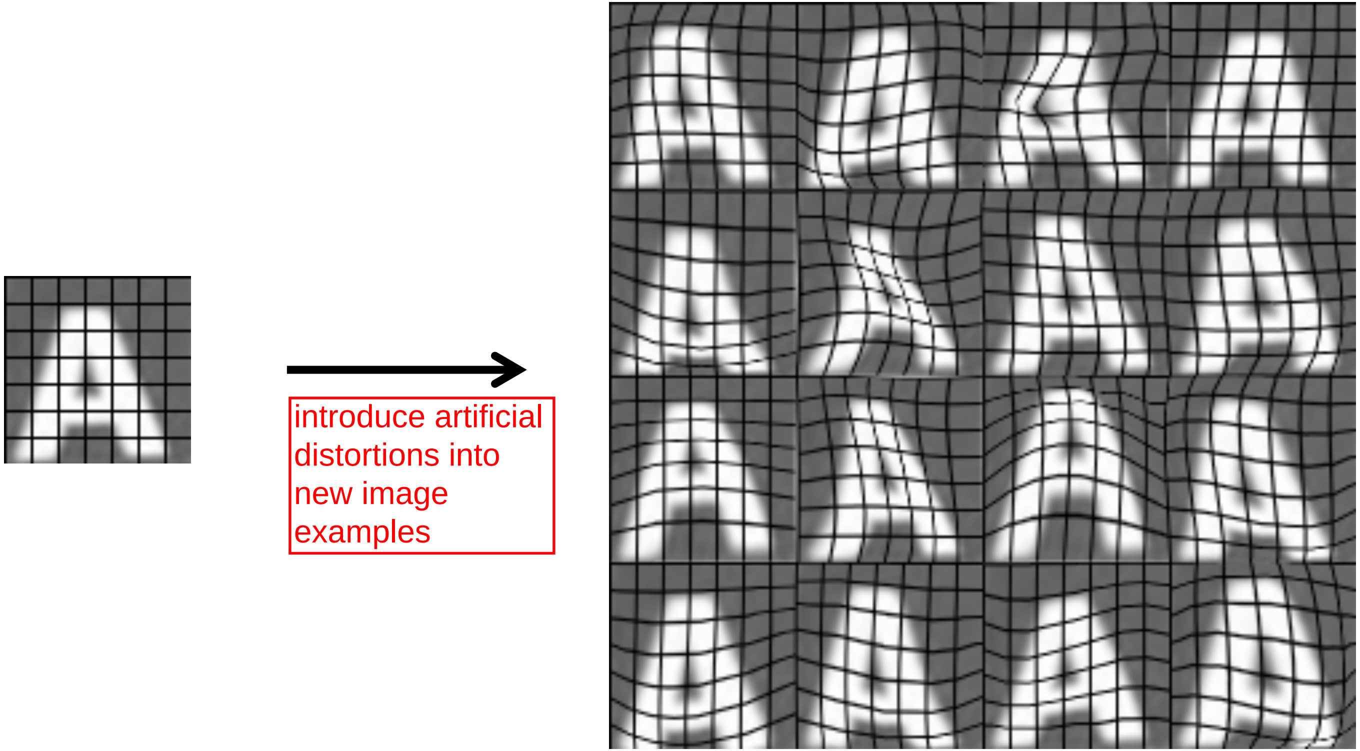
Real data



Synthetic data

[Adam Coates and Tao Wang]

Synthesizing data by introducing distortions



introduce artificial distortions into new image examples

[Adam Coates and Tao Wang]

Synthesizing data by introducing distortions: Speech recognition

Original audio: if you have labeled example

Audio on bad cellphone connection

add noise to synthesize the data

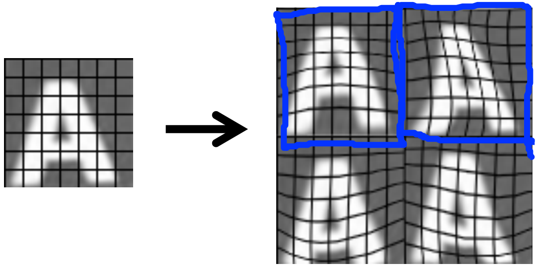
Noisy background: Crowd

Noisy background: Machinery

[www.pdsounds.org]

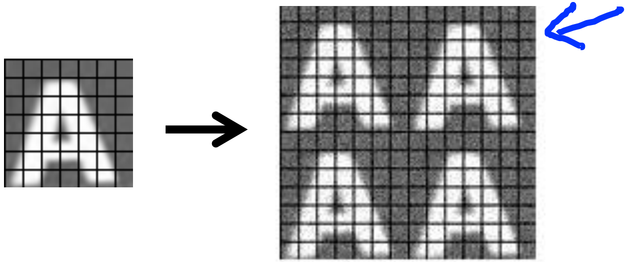
Synthesizing data by introducing distortions

- Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

- Usually does not help to add purely random/meaningless noise to your data.

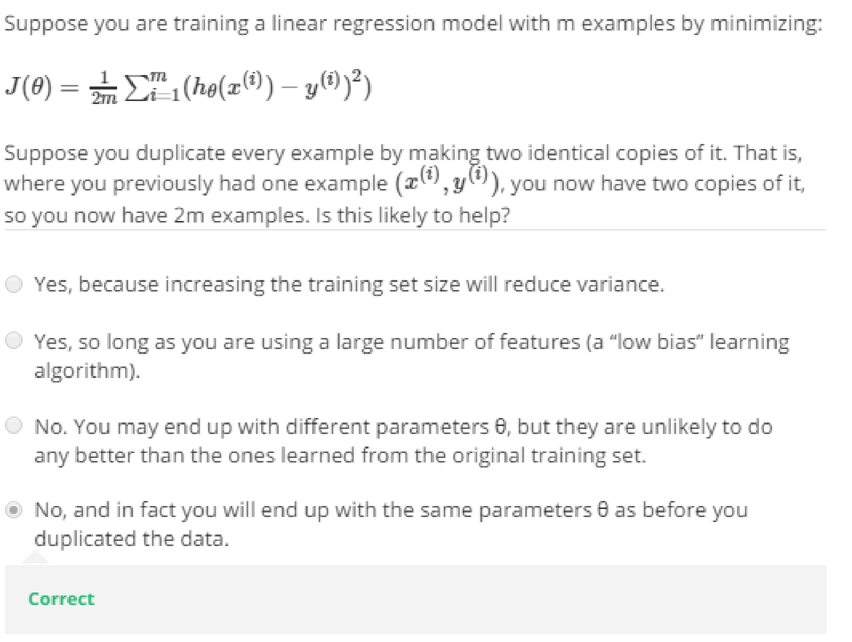


$x_i =$ intensity (brightness) of pixel i

$x_i \leftarrow x_i +$ random noise

[Adam Coates and Tao Wang]

Andrew Ng



Suppose you are training a linear regression model with m examples by minimizing:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2)$$

Suppose you duplicate every example by making two identical copies of it. That is, where you previously had one example $(x^{(i)}, y^{(i)})$, you now have two copies of it, so you now have 2m examples. Is this likely to help?

○ Yes, because increasing the training set size will reduce variance.

○ Yes, so long as you are using a large number of features (a "low bias" learning algorithm).

○ No. You may end up with different parameters θ, but they are unlikely to do any better than the ones learned from the original training set.

● No, and in fact you will end up with the same parameters θ as before you duplicated the data.

Correct

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
   - Artificial data synthesis
   - Collect/label it yourself
   - "Crowd source" (E.g. Amazon Mechanical Turk)

→ #hours?

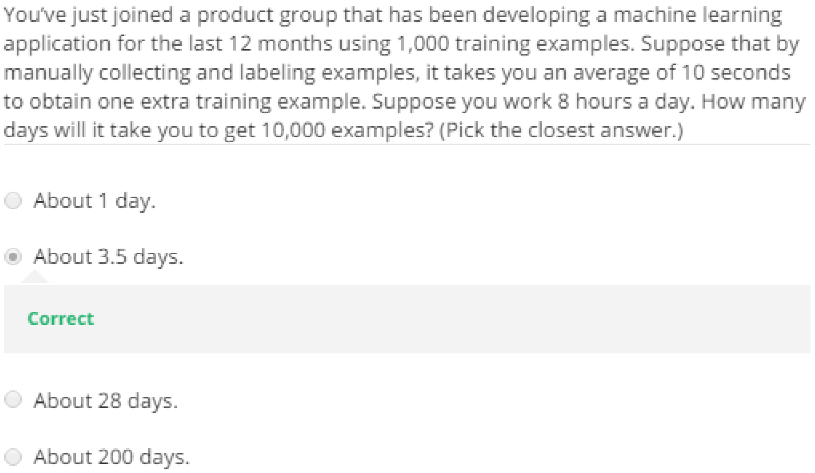
m = 1,000

→ 10 secs/example ←

m = 10,000

if you don't have a low bias classifier, one other thing that's worth trying is to keep increasing the number of features that your classifier has, increasing the number of hidden units in your network until you actually have a low bias falsifier

a few websites or a few services that allow you to hire people on the web to inexpensively label large training sets for you.



You've just joined a product group that has been developing a machine learning application for the last 12 months using 1,000 training examples. Suppose that by manually collecting and labeling examples, it takes you an average of 10 seconds to obtain one extra training example. Suppose you work 8 hours a day. How many days will it take you to get 10,000 examples? (Pick the closest answer.)

○ About 1 day.

● About 3.5 days.

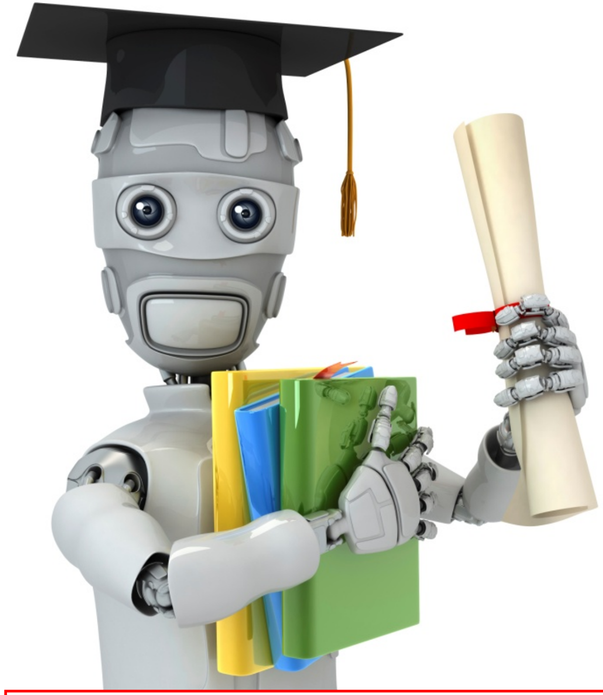
Correct

○ About 28 days.

○ About 200 days.

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
    - Artificial data synthesis
    - Collect/label it yourself
    - "Crowd source" (E.g. Amazon Mechanical Turk)



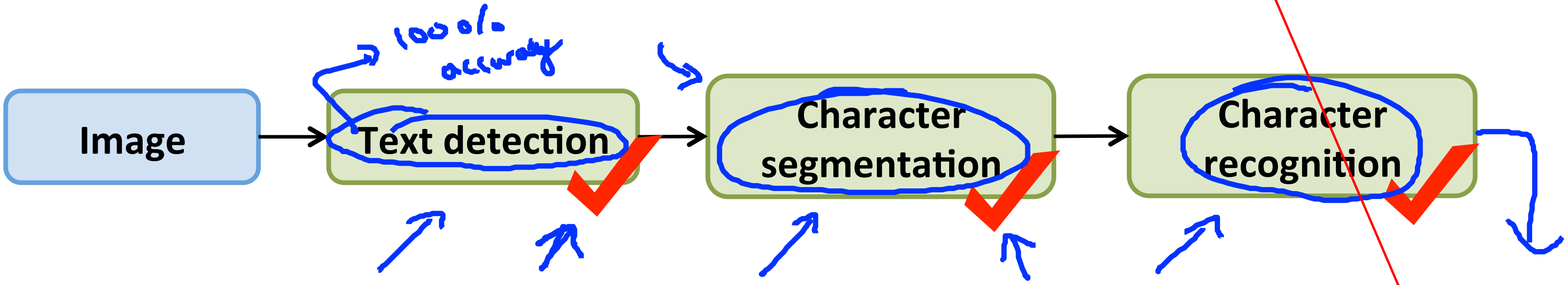
Application example: Photo OCR

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

what you really want to avoid is that you and your colleagues spend a lot of time working on some component. Only to realize after weeks or months of time spent, that all that worked just doesn't make a huge difference on the performance of the final system. In this video what I'd like to do is to talk about something called ceiling analysis. When your team work on the pipeline machine on your system, this can sometimes give you a very strong guidance on what parts of the pipeline might be the best use of your time to work on.

how much could you possibly gain if one of these components became absolutely perfect?

Estimating the errors due to each component (ceiling analysis)



What part of the pipeline should you spend the most time trying to improve?

overall system has

Component	Accuracy
Overall system	72%
Text detection	89%
acter segmentation	90%
Character recognition	100%

increase by

↓ 17%

↓ 1%

↓ 10%

overall system accuracy before we do any improvement

with perfect text detection (manually give the system where it is), the overall performance go up to 89%.

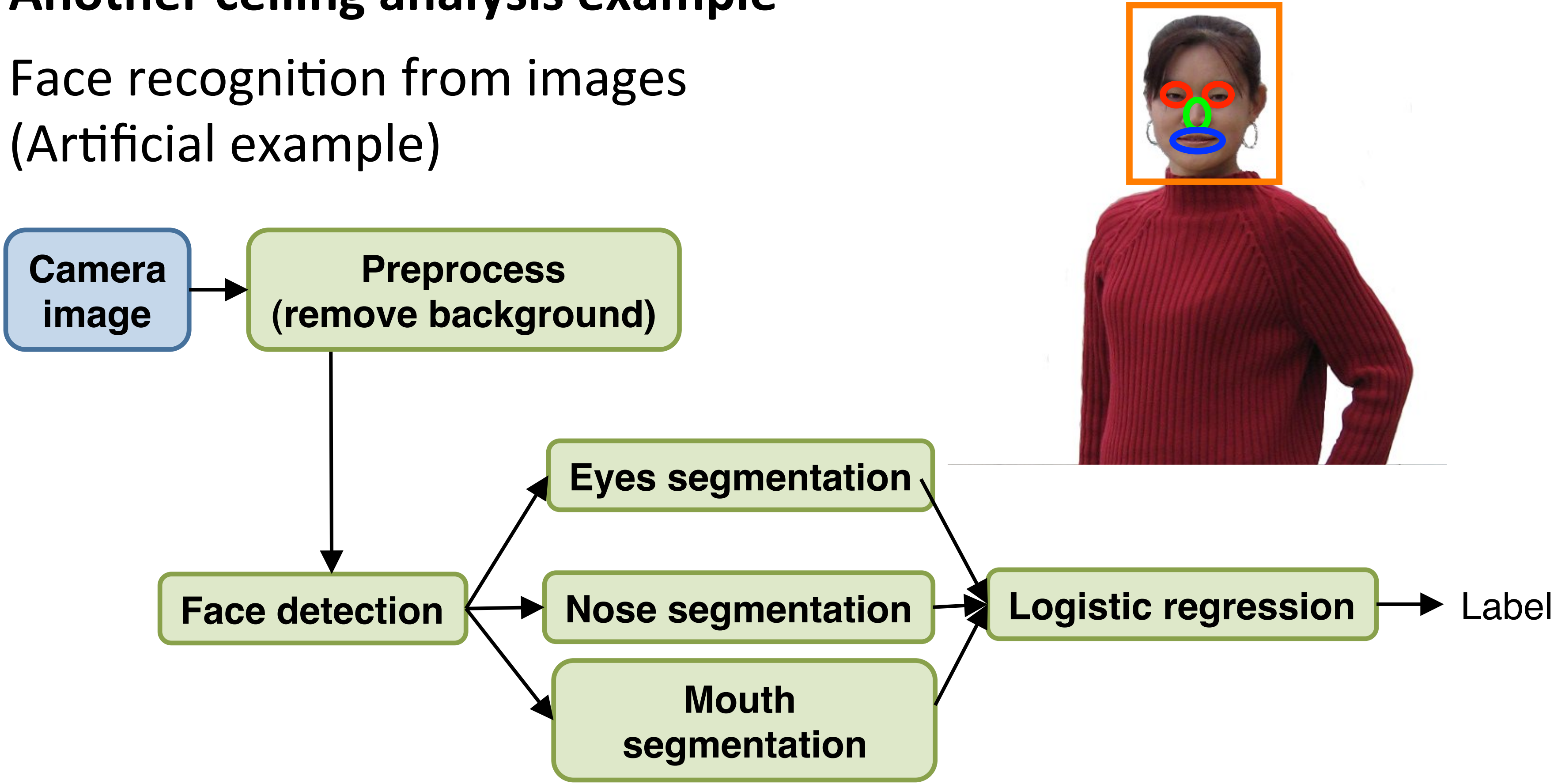
if manually give the segmentation, the overall performance go up to 90%.

you want to put more ppl here to gain 17% improvement

Andrew Ng

Another ceiling analysis example

Face recognition from images
(Artificial example)



Camera image

Preprocess (remove background)

Face detection

Eyes segmentation

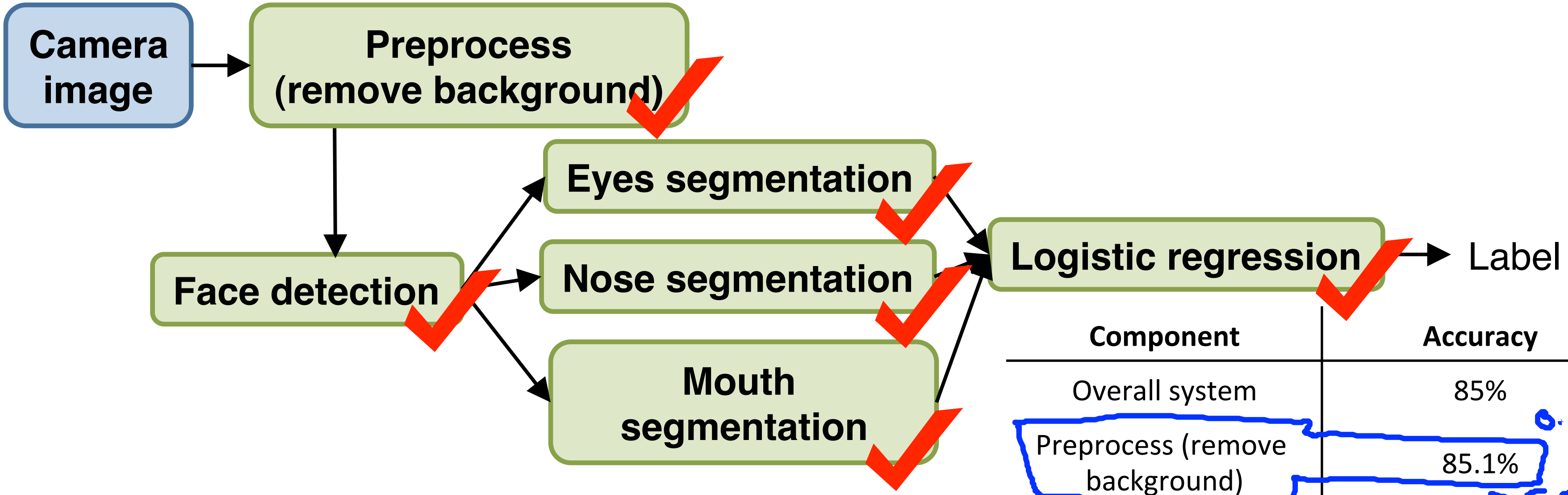
Nose segmentation

Mouth segmentation

Logistic regression

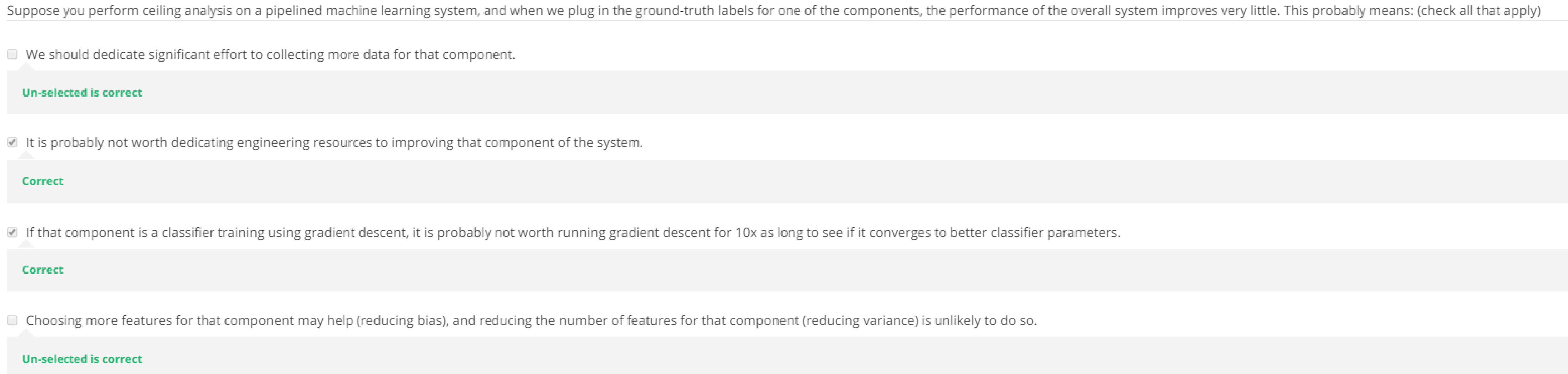
Label

Another ceiling analysis example



Component	Accuracy	
Overall system	85%	
→ Preprocess (remove background)	85.1%	0.1%
→ Face detection	91%	→ 5.9%
→ Eyes segmentation	95%	→ 4%
Nose segmentation	96%	1%
Mouth segmentation	97%	1%
→ Logistic regression	100%	→ 3%

Andrew Ng



Suppose you perform ceiling analysis on a pipelined machine learning system, and when we plug in the ground-truth labels for one of the components, the performance of the overall system improves very little. This probably means: (check all that apply)

□ We should dedicate significant effort to collecting more data for that component.

Un-selected is correct

☑ It is probably not worth dedicating engineering resources to improving that component of the system.

Correct

☑ If that component is a classifier training using gradient descent, it is probably not worth running gradient descent for 10x as long to see if it converges to better classifier parameters.

Correct

□ Choosing more features for that component may help (reducing bias), and reducing the number of features for that component (reducing variance) is unlikely to do so.

Un-selected is correct

Summary: Main topics

Supervised Learning $(x^{(i)}, y^{(i)})$

- Linear regression, logistic regression, neural networks, SVMs

Unsupervised Learning $x^{(i)}$

- 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.