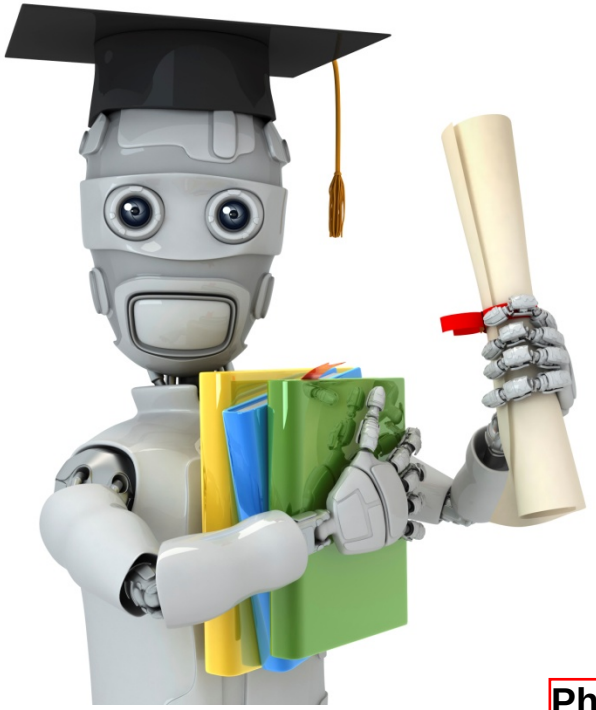


Machine Learning

# Application example: Photo OCR

## Problem description and pipeline

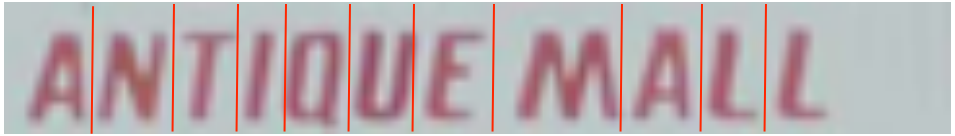
**Photo optical Character recognition**

# The Photo OCR problem

# Photo OCR pipeline

1. Text detection

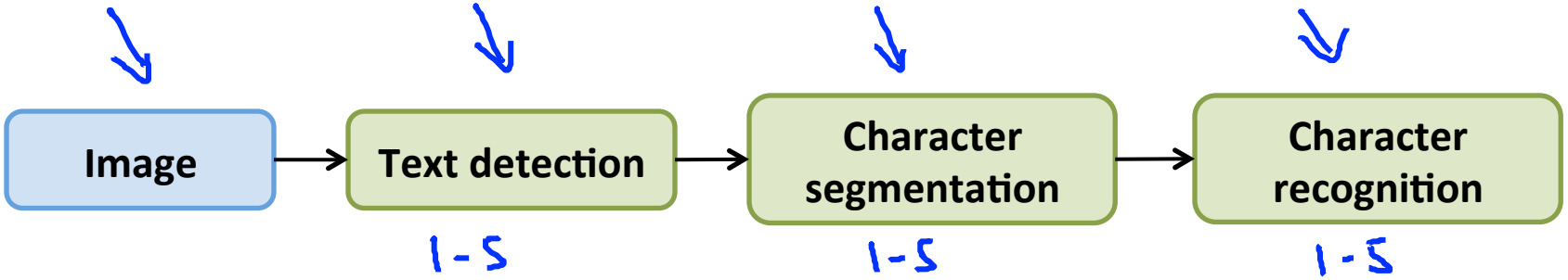

2. Character segmentation




3. Character classification


Clearing → Clearing

# Photo OCR pipeline



# Application example: Photo OCR

# Sliding windows

1. Suppose you are running a sliding window detector to find text in images. Your input images are 1000x1000 pixels. You will run your sliding windows detector at two scales, 10x10 and 20x20 (i.e., you will run your classifier on lots of 10x10 patches to decide if they contain text or not; and also on lots of 20x20 patches), and you will "step" your detector by 2 pixels each time. About how many times will you end up running your classifier on a single 1000x1000 test set image?

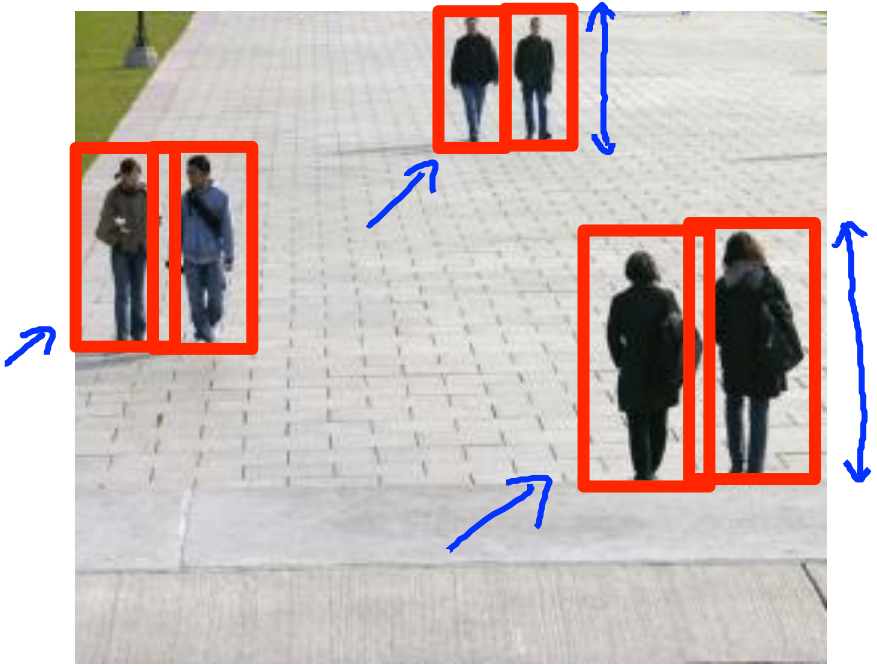
- ○ 1,000,000
- ○ 250,000
- ○ 100,000
- ◉ 500,000

✓ **Correct**

With a stride of 2, you will run your classifier approximately 500 times for each dimension. Since you run the classifier twice (at two scales), you will run it 2 * 500 * 500 = 500,000 times.

## Text detection


## Pedestrian detection



# Supervised learning for pedestrian detection

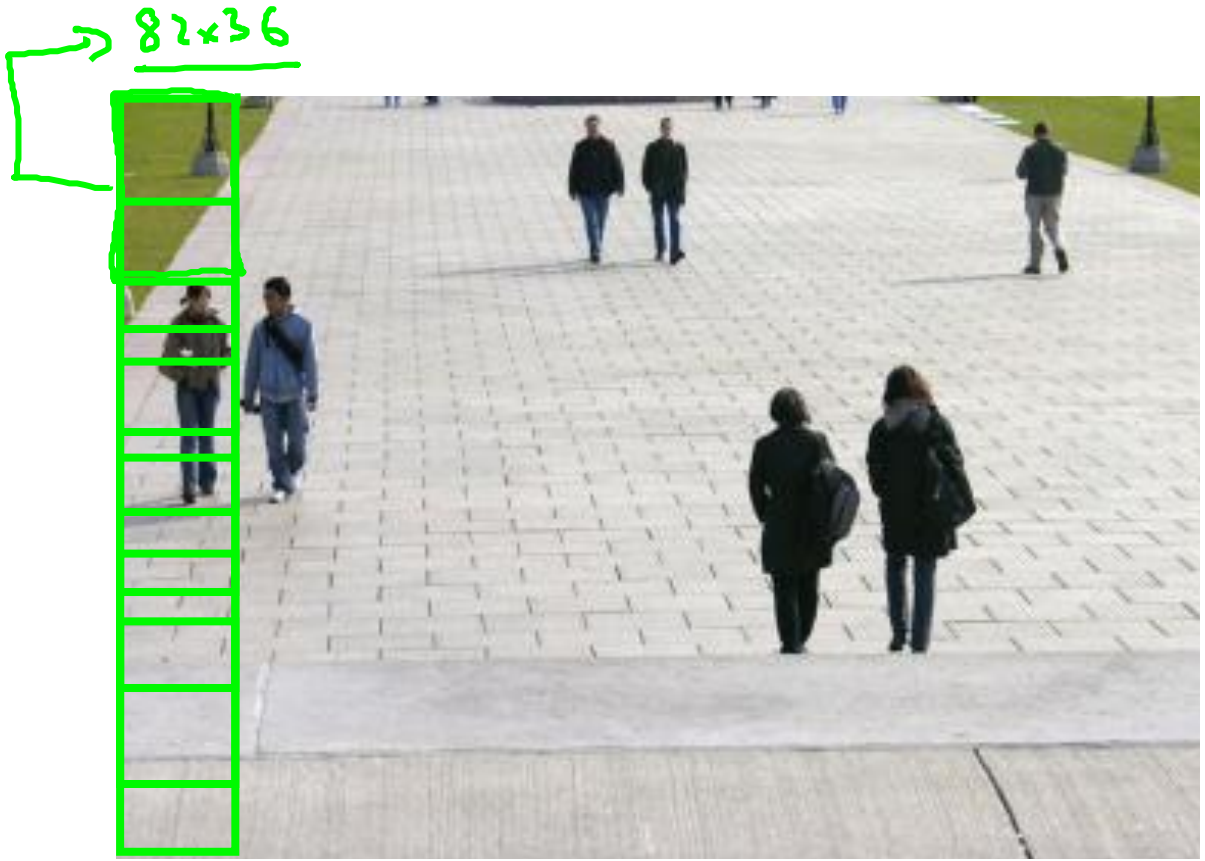
$x =$ pixels in 82x36 image patches

Positive examples $(y = 1)$

Negative examples $(y = 0)$

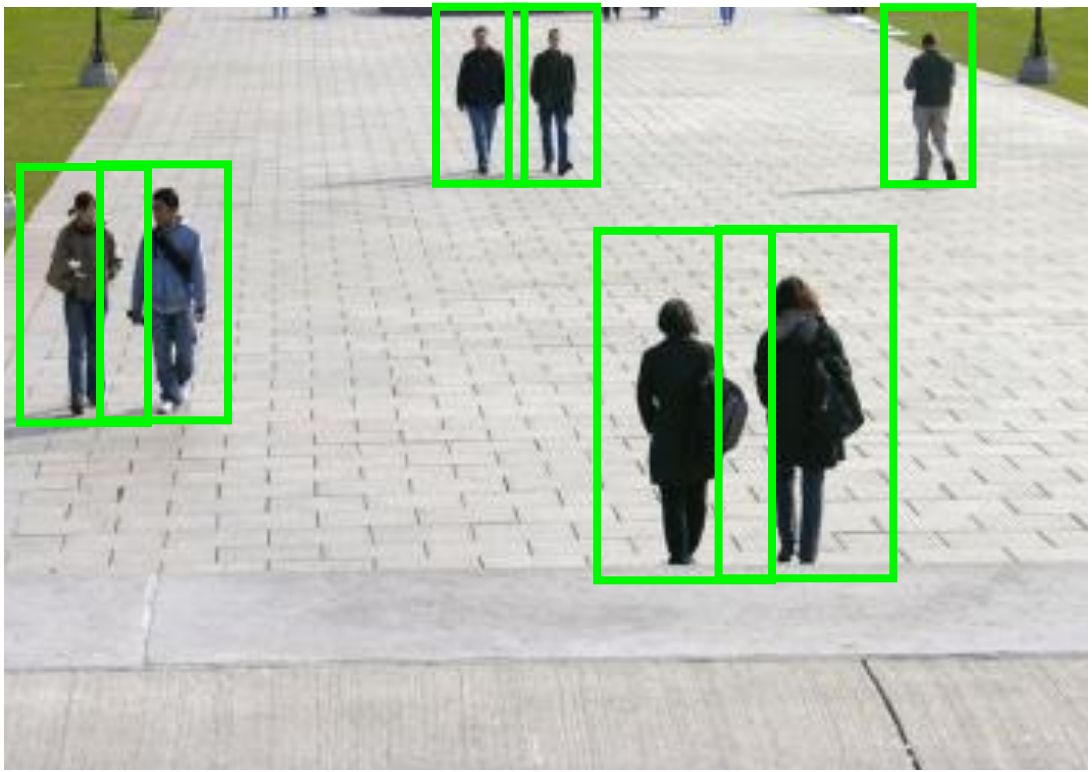
# Sliding window detection

# Sliding window detection



# Sliding window detection

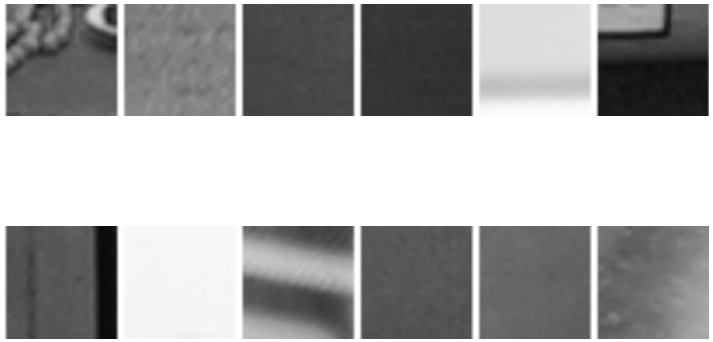
# Sliding window detection



# Text detection

# Text detection

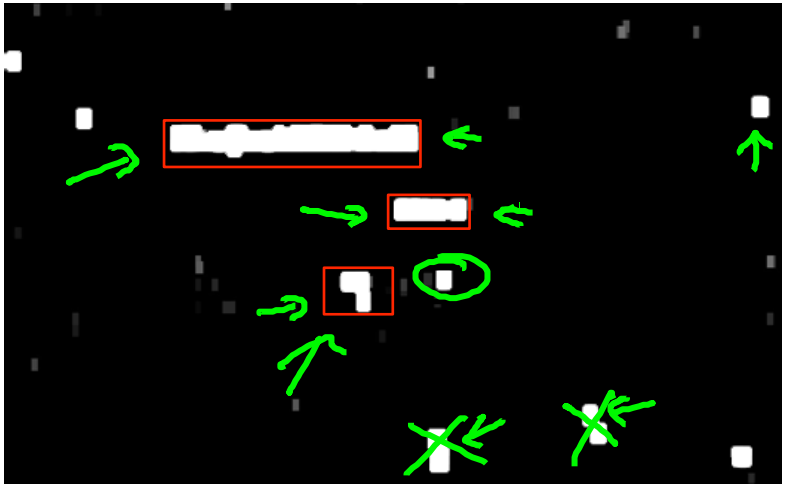
Positive examples $(y = 1)$



Negative examples $(y = 0)$

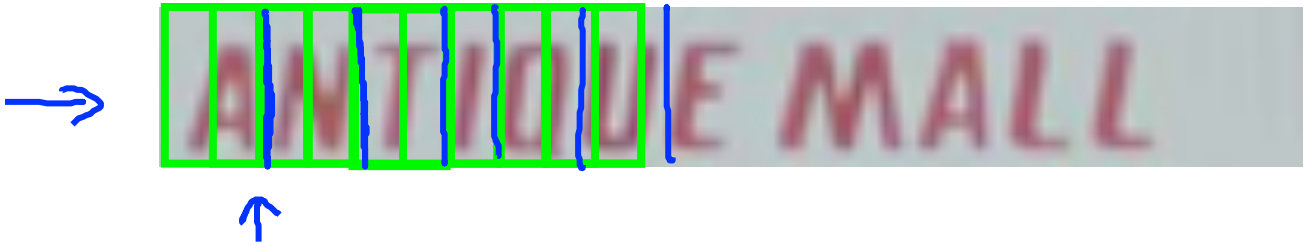
# Text detection

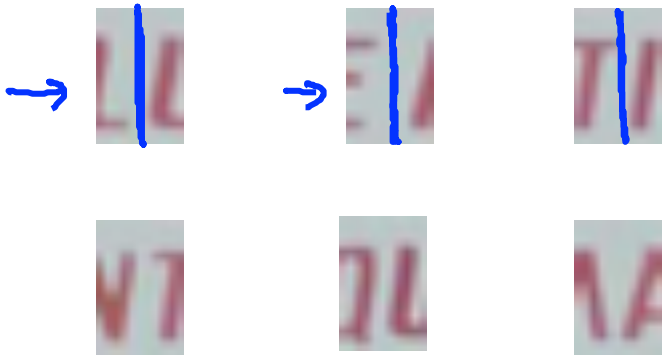




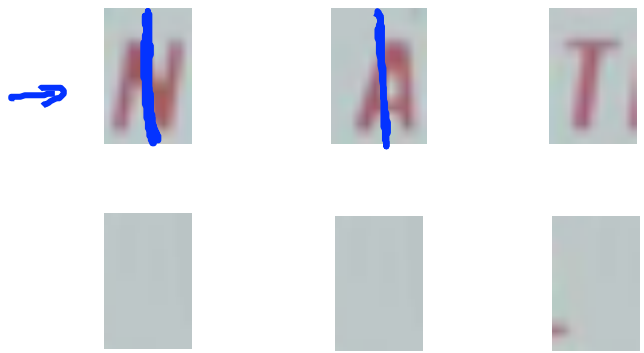
[David Wu]

# 1D Sliding window for character segmentation





Positive examples $(y = 1)$

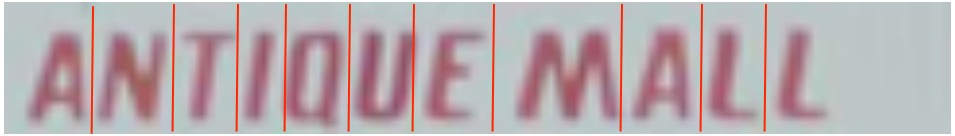


Negative examples $(y = 0)$

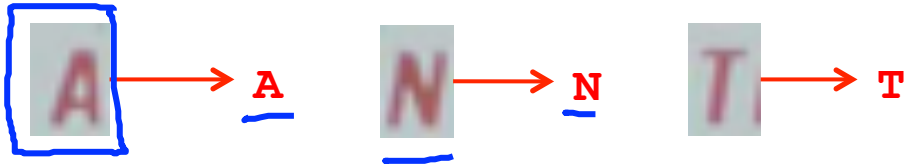
# Photo OCR pipeline

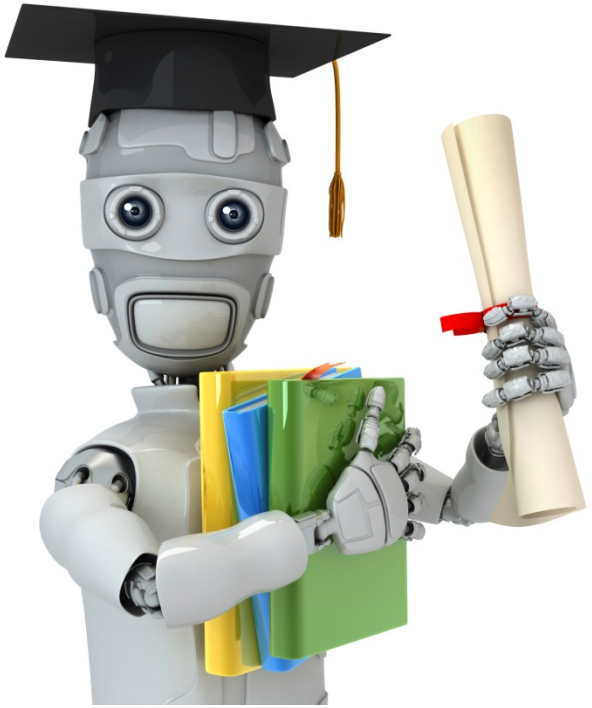
1. Text detection


2. Character segmentation




3. Character classification



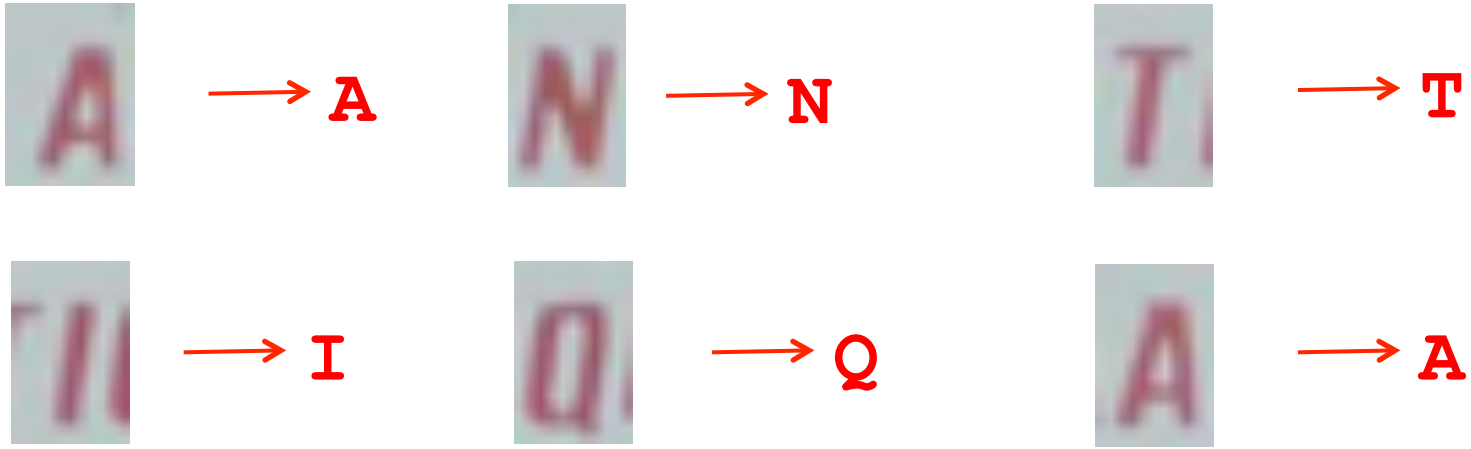


Machine Learning

# Application example: Photo OCR

## Getting lots of data: Artificial data synthesis

# Character recognition



# Artificial data synthesis for photo OCR

Real data

What is the ***"Aspect ratio"***?

Abcdefg

Abcdefg

Abcdefg

Abcdefg

Abcdefg

[Adam Coates and Tao Wang]

# Artificial data synthesis for photo OCR

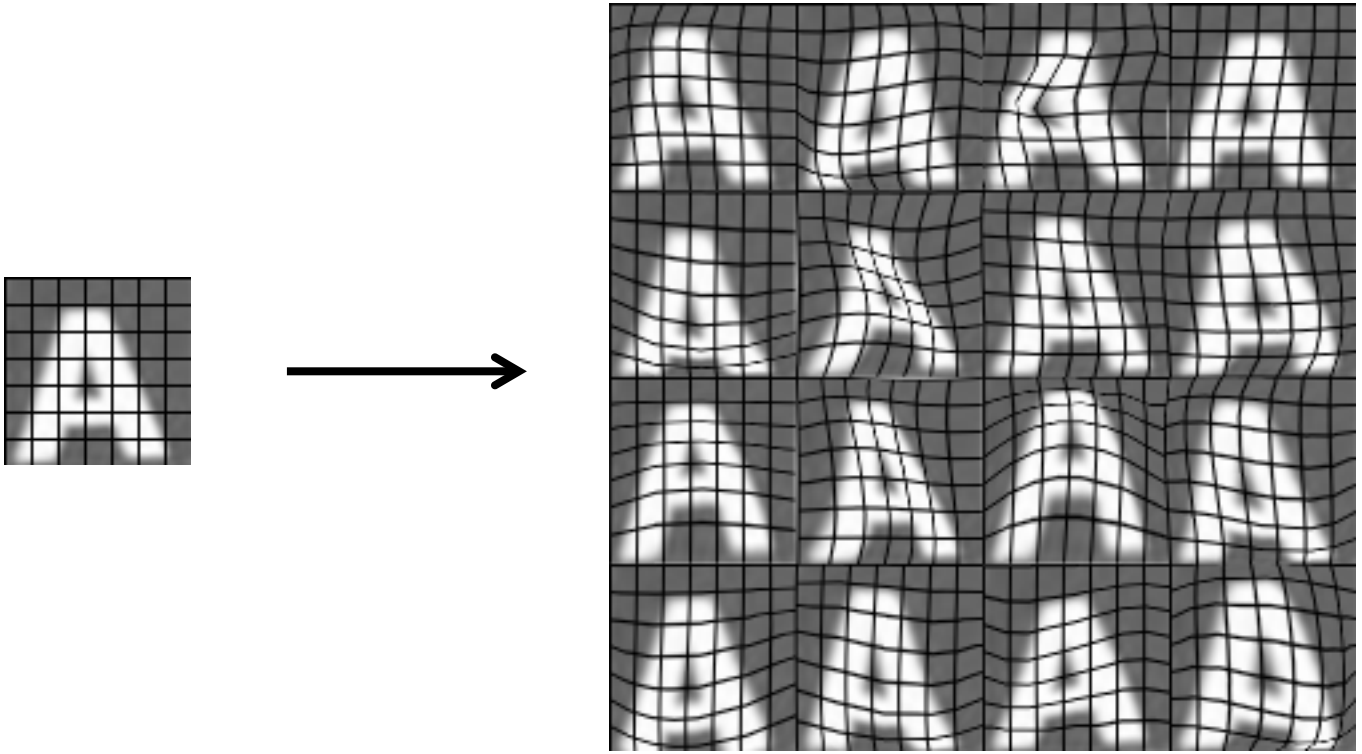
Real data

Synthetic data

[Adam Coates and Tao Wang]

Bóp méo

# Synthesizing data by introducing distortions



[Adam Coates and Tao Wang]

# Synthesizing data by introducing distortions: Speech recognition

Original audio:

Audio on bad cellphone connection

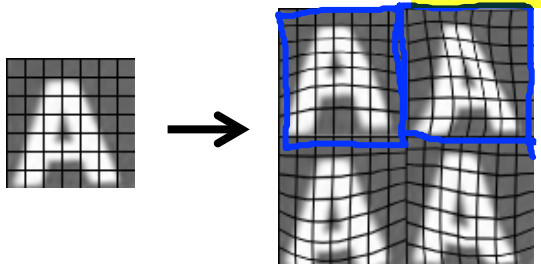
Noisy background: Crowd

Noisy background: Machinery

[www.pdsounds.org]

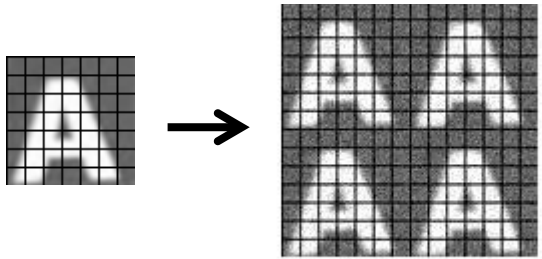
# Synthesizing data by introducing distortions

- 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.



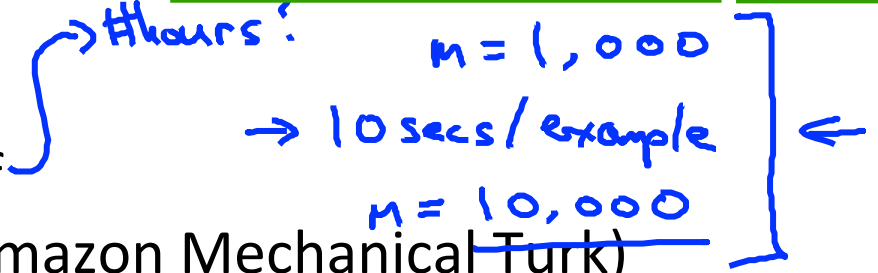
- $x_i$ = intensity (brightness) of pixel $i$
- $x_i \leftarrow x_i +$ random noise

meaningless

[Adam Coates and Tao Wang]

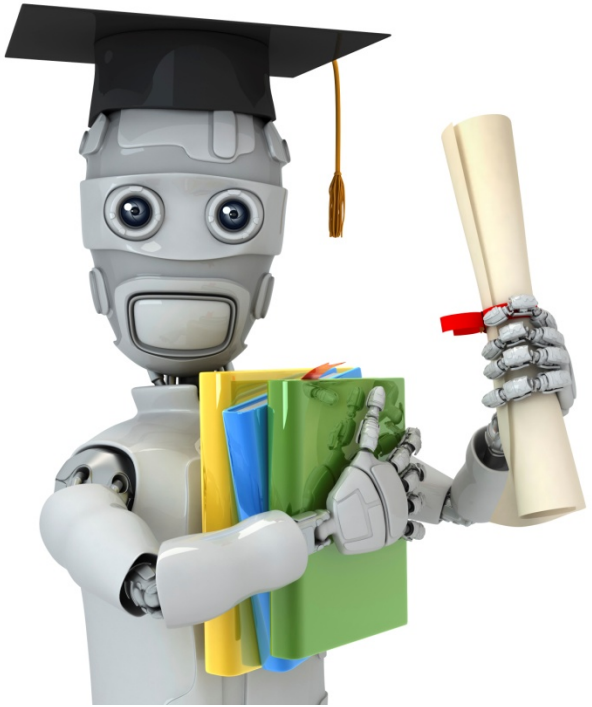
## Discussion on getting more data

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)



## 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)

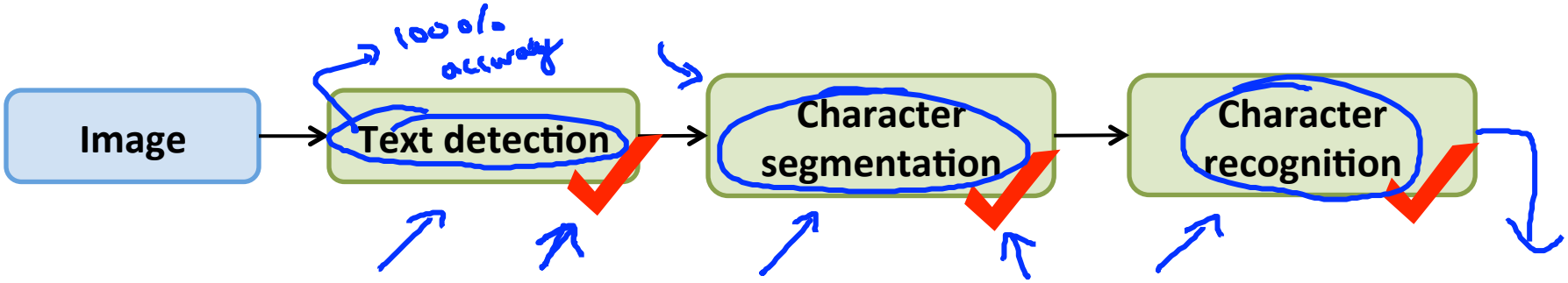


Machine Learning

# Application example: Photo OCR

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

# Estimating the errors due to each component (ceiling analysis)

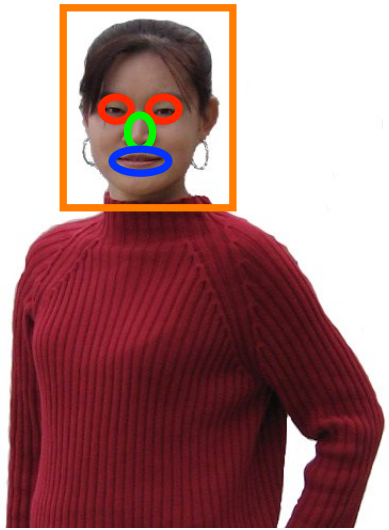



What part of the pipeline should you spend the most time trying to improve?

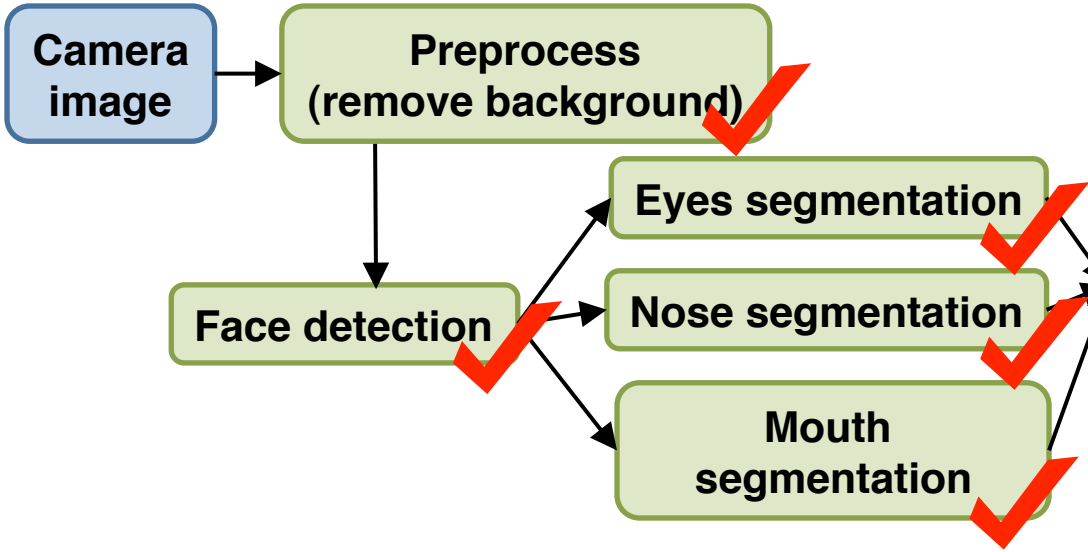
| Component | Accuracy |
|---|---|
| Overall system | 72% |
| Text detection | 89% |
| Character segmentation | 90% |
| Character recognition | 100% |

# Another ceiling analysis example

Face recognition from images
(Artificial example)



# Another ceiling analysis example




| Component | Accuracy |
|---|---|
| Overall system | 85% |
| Preprocess (remove background) | 85.1% |
| Face detection | 91% |
| Eyes segmentation | 95% |
| Nose segmentation | 96% |
| Mouth segmentation | 97% |
| Logistic regression | 100% |

0.1%
5.9%
4%
1%
1%
3%