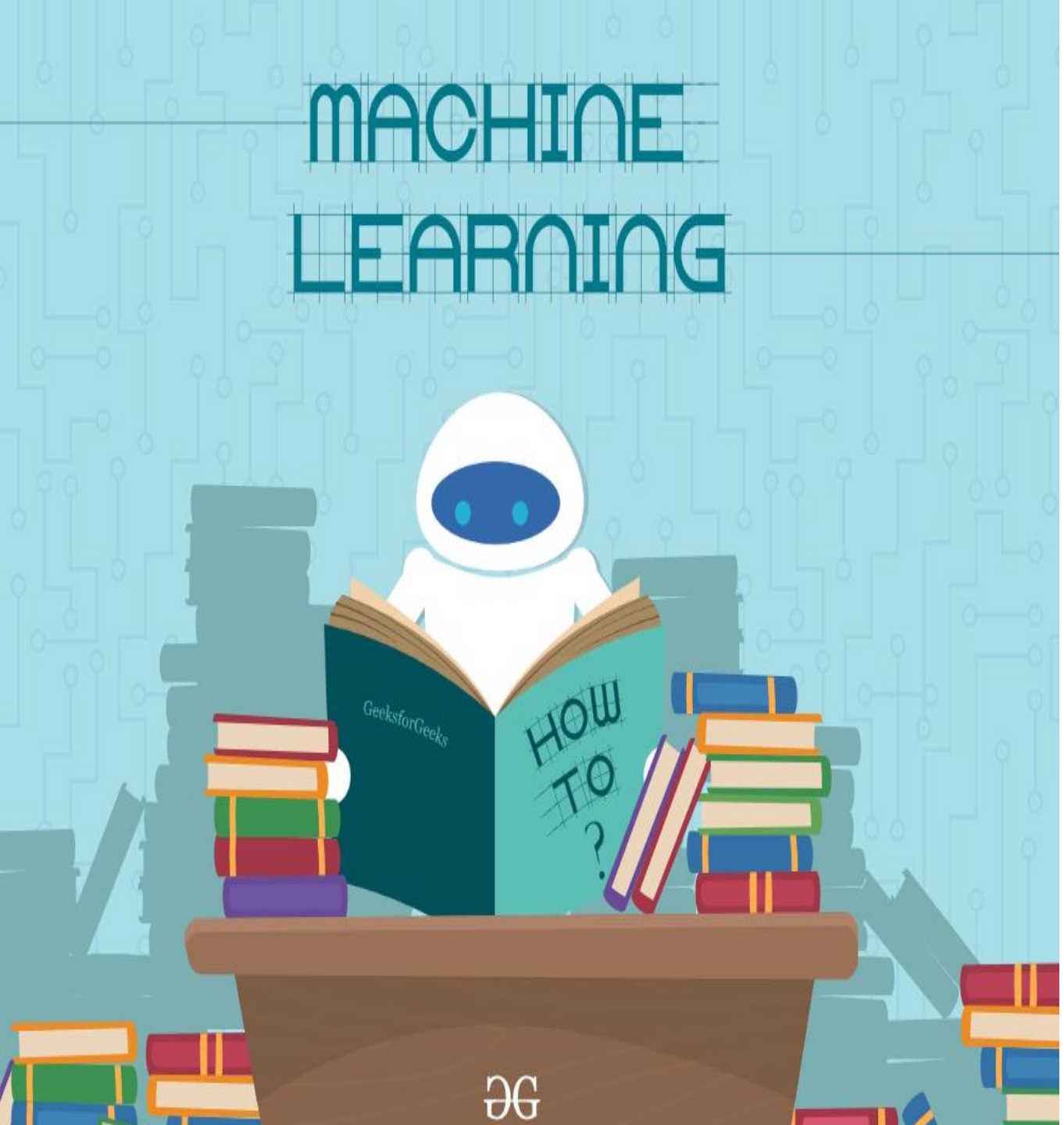


>TCC
THE CODERS' CLUB

Application Example: Photo OCR

WEEK 11

Problem Description and Pipeline

What is OCR?

Optical character recognition or optical character reader is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image.

Photo OCR Pipeline:

1. Text Detection

2. Character Segmentation

3. Character Classification

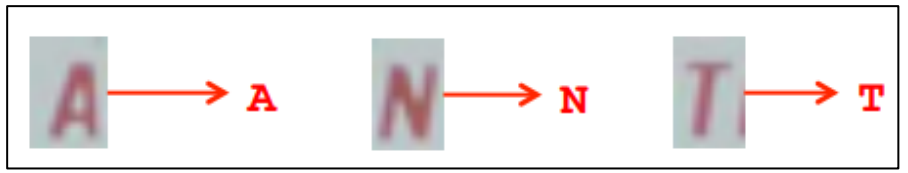


Photo OCR Pipeline:

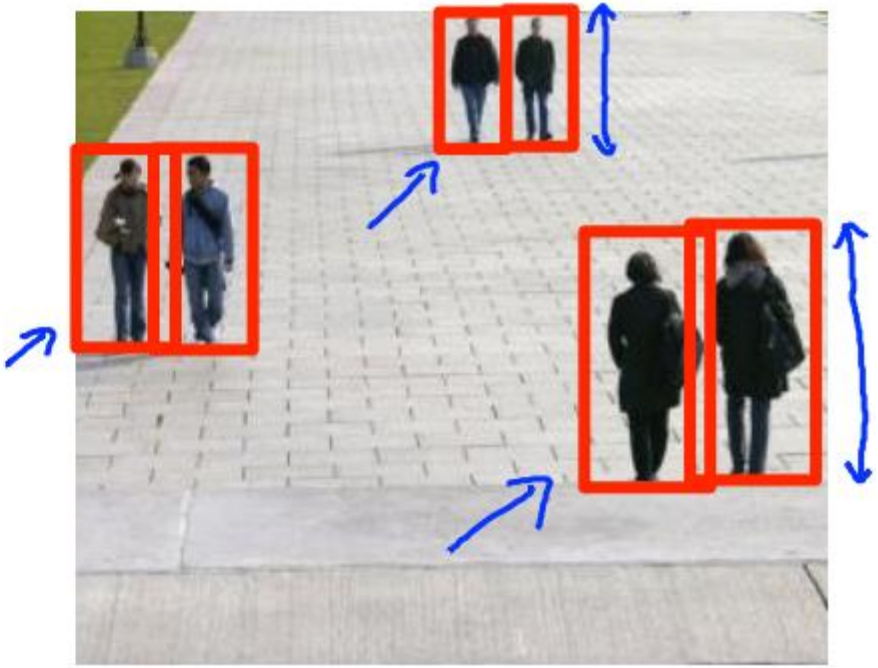
When someone refers to a “machine learning pipeline,” he or she is referring to:

- A PhotoOCR system
- A character recognition system
- A system with many stages / components, several of which may use machine learning.
- An application in plumbing. (LeL)

Sliding Windows

Consider the following two examples:

Text Detection



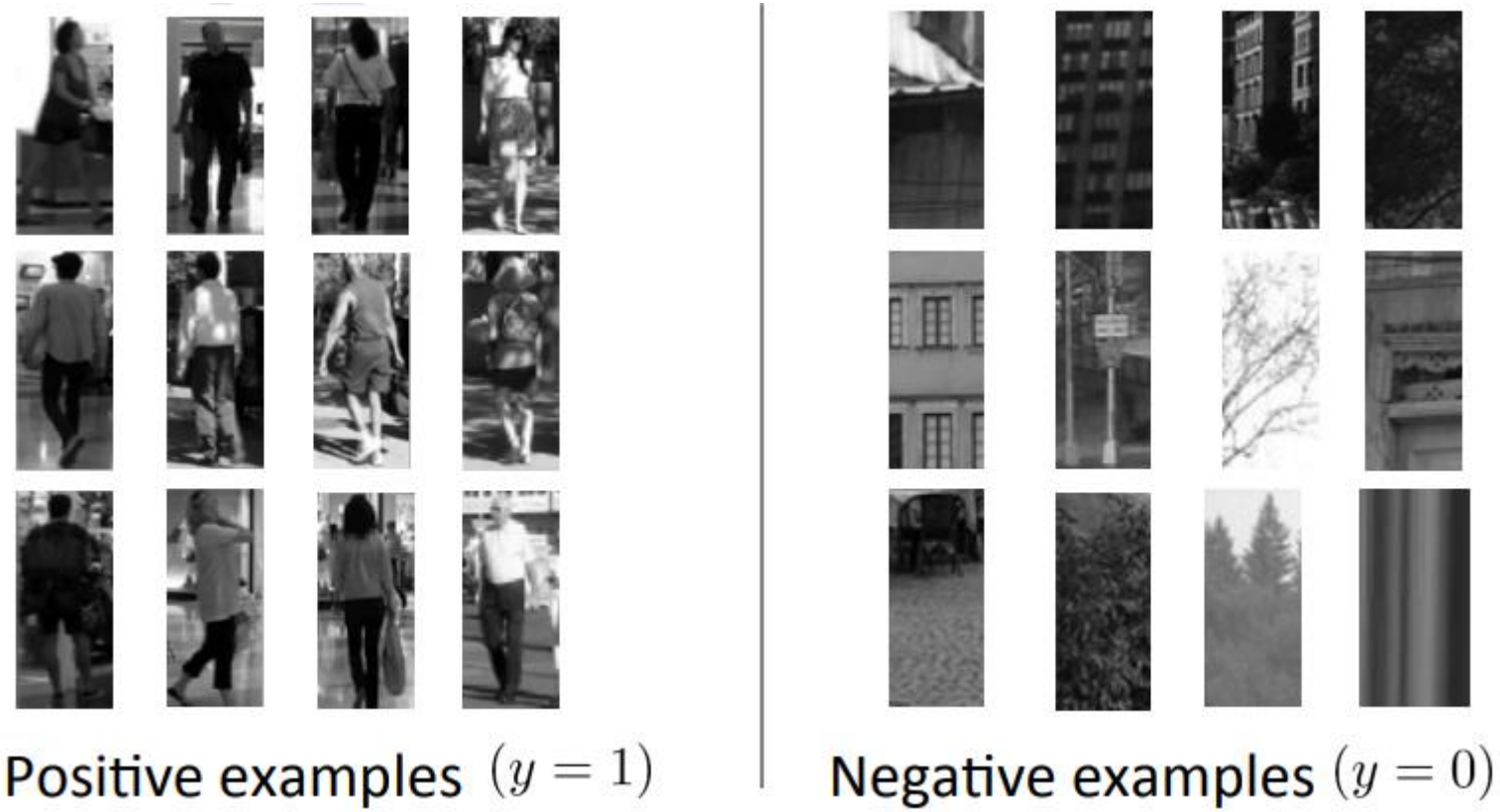
Pedestrian Detection

In both these examples, we see a rectangular bounded box or simply a “window” that detects text and pedestrian respectively.

Supervised Learning for Pedestrian Detection:

Consider a dataset of positive (y=1) and negative (y=0) examples for pedestrian detection.

Each image patch is of 82x36 pixels.



Positive examples $(y = 1)$

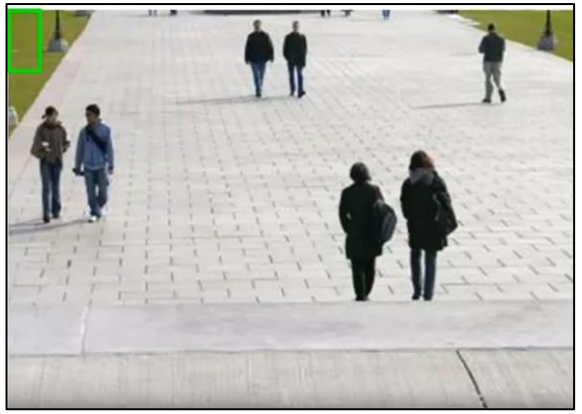
Negative examples $(y = 0)$

Sliding Window Detection:

Now suppose we get a new image (test data) and we want to try to find a pedestrian in the image.

- We start by taking a rectangular patch of this image and run the patch through image classifier to determine whether or not there is a pedestrian in the image.
- We keep sliding the window through the image by some constant pixels and run each window through image classifier.

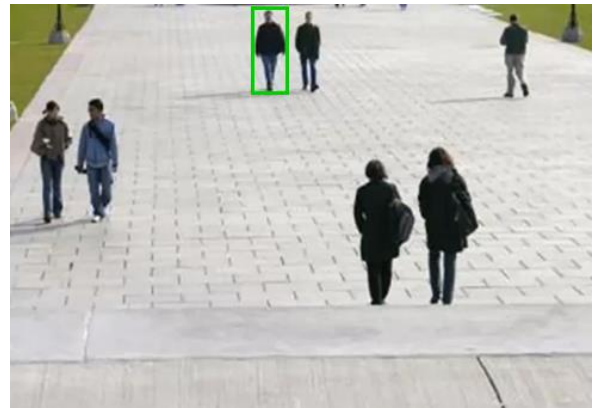
Sliding Window Detection:



Window 1 (y=0)

Window 2 (y=0)

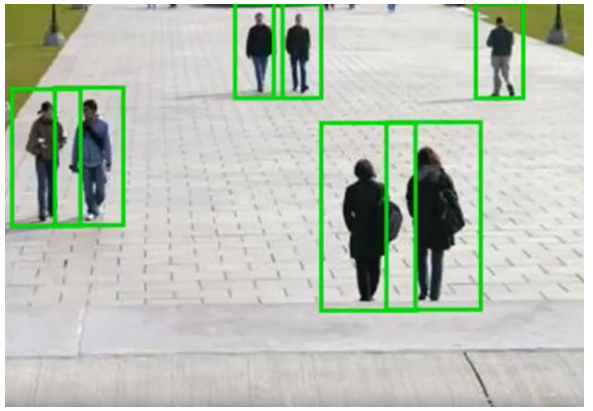
...



Window 10 (y=1)

Window 15 (y=0)

...



All pedestrians detected

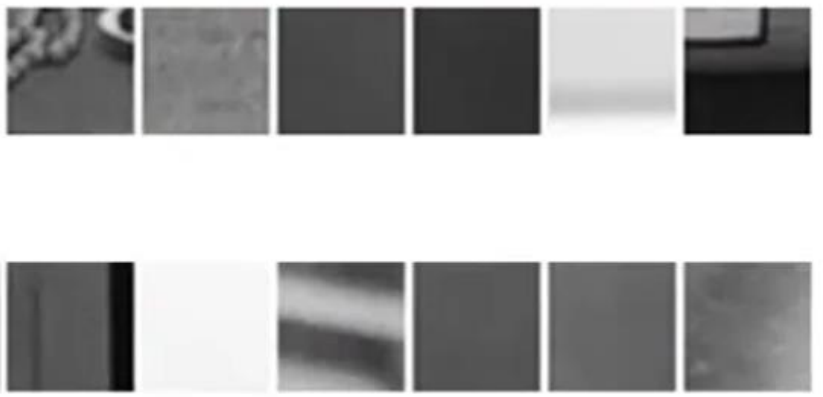
To understand more about Sliding Window:
https://www.youtube.com/watch?v=rUi7WbKVMig
https://datalya.com/blog/machine-learning/object-detection-with-sliding-window-algorithm

Text Detection:

We follow a similar approach to text detection problem as well.

Training set of positive and negative examples of text.

Positive examples $(y = 1)$



Negative examples $(y = 0)$

Consider an image as an example.

Sliding Window

- We run the sliding window classifiers on lots of little image patches on the above image and end up getting a black and white image.
- The white regions show where our text is found.
- We apply an “expansion operator” to it so that rectangles are drawn around all the regions where the text is found.

Expansion Operator

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
- About 40,000 times

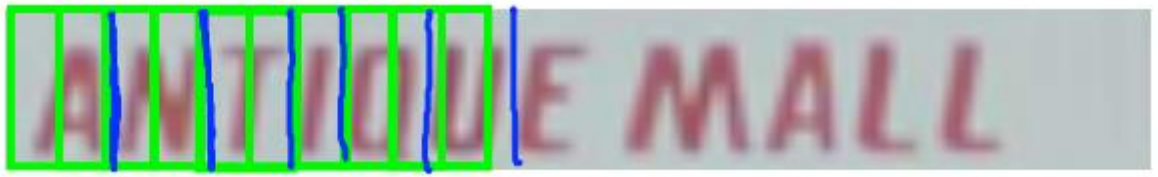
Solution:

Image dimension = 200, step size = 4

200/4 = 50

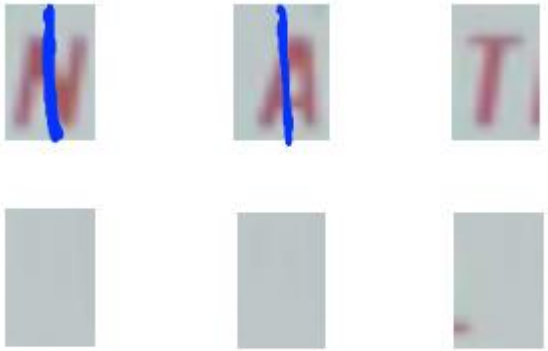
50 x 50 = 2,500 times

1D Sliding Window for Character Segmentation:



Positive examples $(y = 1)$

(Represent a midpoint between two distinct characters)

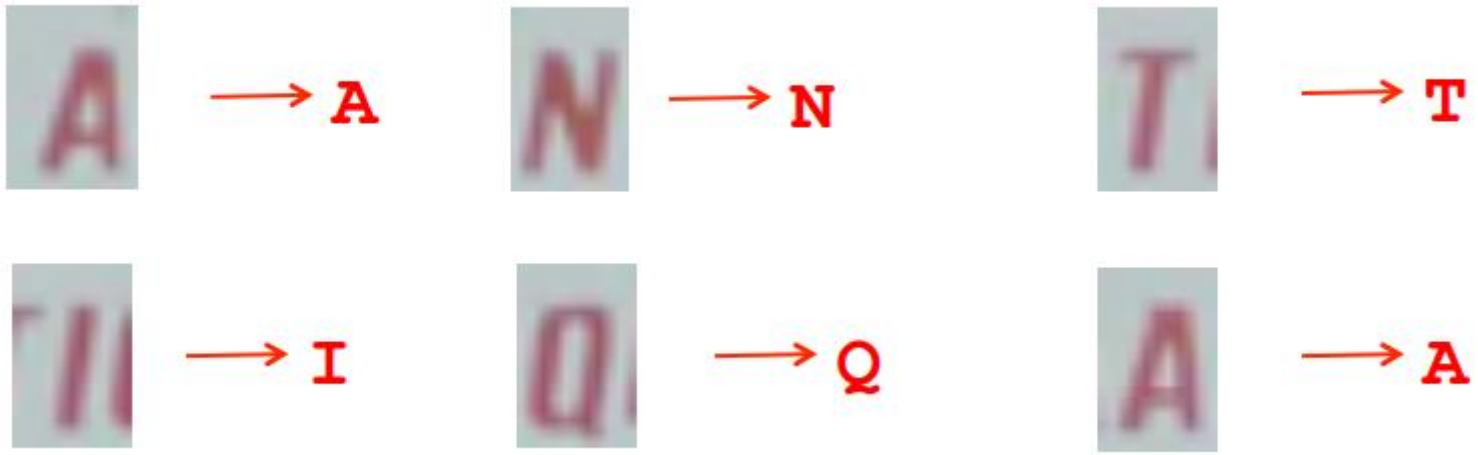


Negative examples $(y = 0)$

(No midpoint between two distinct characters)

Getting Lots of Data: Artificial Data Synthesis

Character Recognition:



- To recognize a character from an image, we need to train the dataset with a lot of variations (e.g. letters represented in different font styles, size, color, etc.).
- Data can be artificially synthesized to form a training set with a good mix of variation that will likely help the algorithm to learn and identify characters.

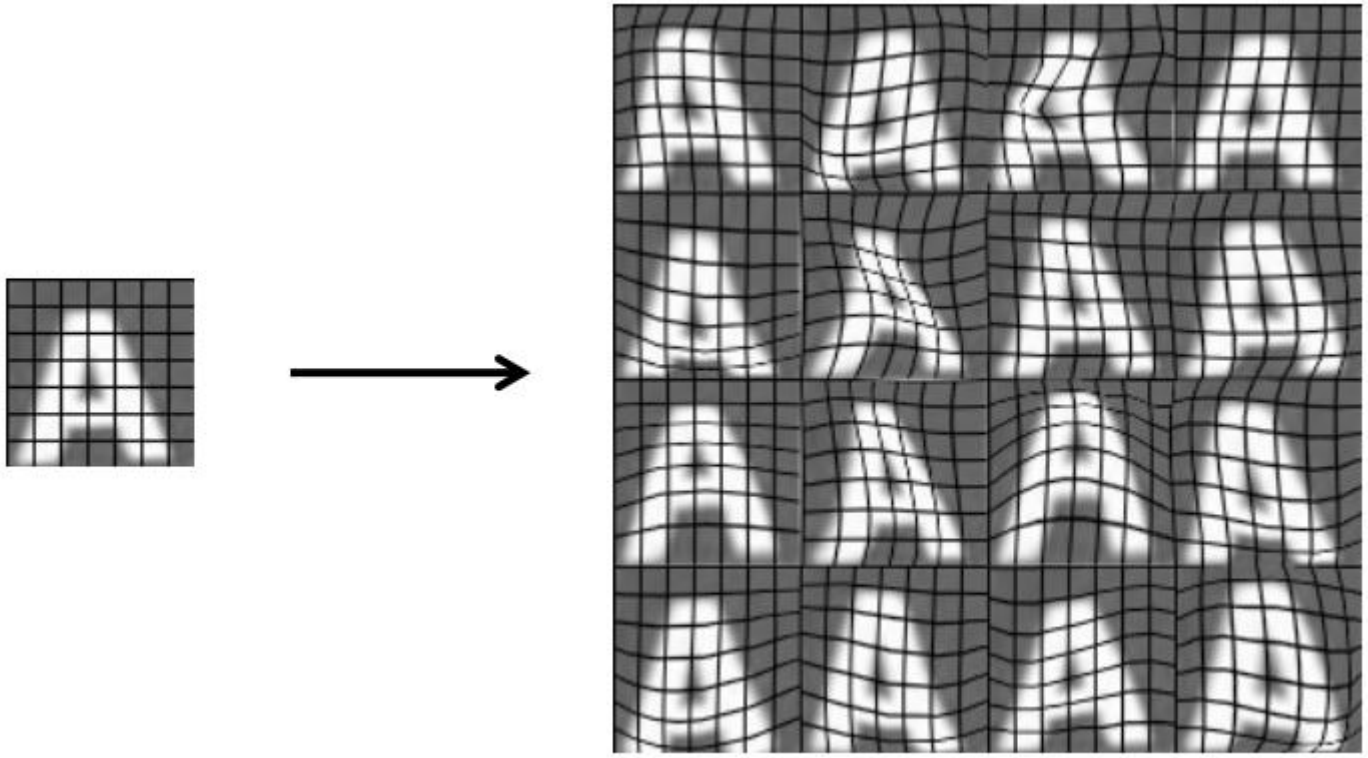
- If we go out and collect the training dataset, it would look something like the data below:
- To come up with a much larger training set, one thing we can do is take characters from different fonts and paste them against different random backgrounds.

Real Data

Extract Characters

Synthetic Data

Synthesizing data by introducing distortions:



By adding distortions to the original data, variation in the synthesized data would increase and it would be better for the algorithm to learn and adapt to newer examples.

Synthesizing data by introducing distortions: Speech Recognition

Original audio:

Audio on bad cellphone connection

Noisy background: Crowd

Noisy background: Machinery

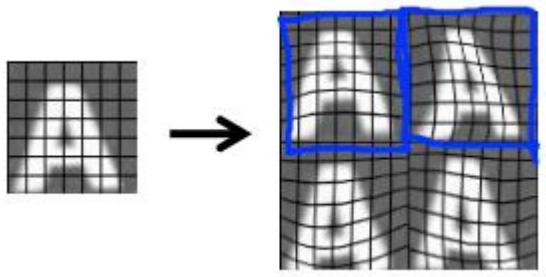
[Source: www.pdsounds.org]

Synthesizing data by introducing distortions:

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

Example:

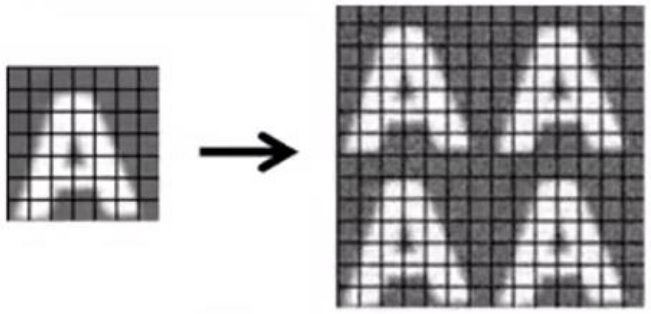
For text recognition:



For speech recognition:
Background noise, bad cellphone connection

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

Example:



x_i = intensity (brightness) of pixel i

$x_i \leftarrow x_i +$ random noise

Discussion on getting more data:

- 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.
- “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)

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.

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
- About 28 days
- About 200 days

Solution:

1 training example in 10 seconds ≡ 6 training examples in 60 seconds (1 minute)

Therefore, 6 x 60 = 360 training examples in 1 hour.

⇒ 360 x 8 = 2,880 training examples in 1 day

Therefore, for 10,000 examples, it would take 10000/2880 ≈ 3.5 days

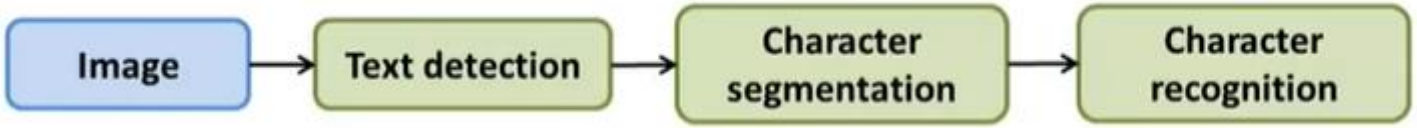
Ceiling Analysis

What part of the pipeline to work on next?

Ceiling Analysis:

Estimating the errors due to each component is called “Ceiling Analysis”.

Example: Text Detection

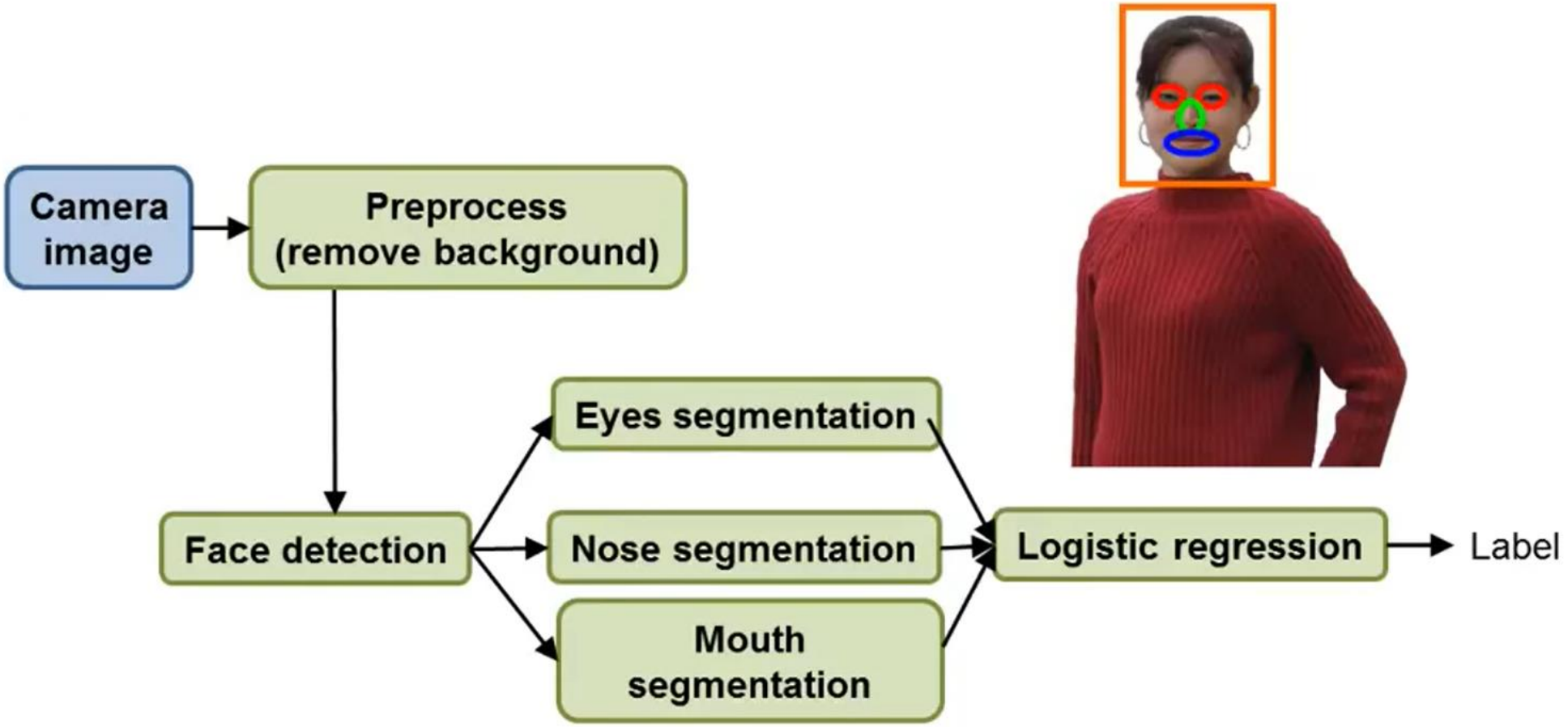


What part of the pipeline should you spend the most time trying to improve?

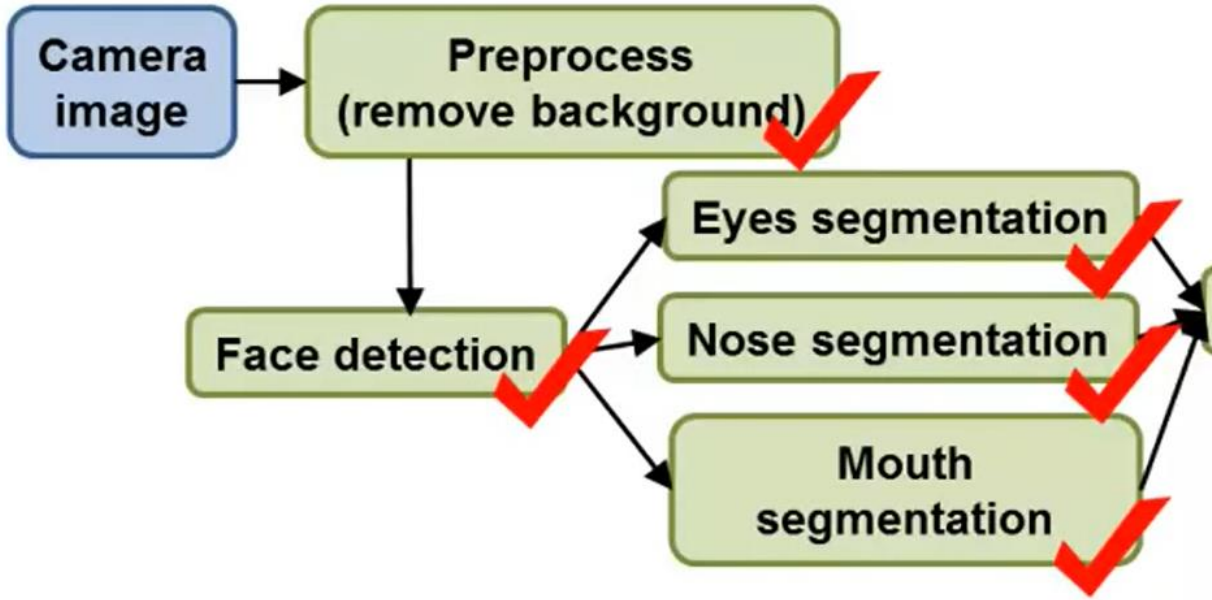
Component	Accuracy
Overall system	72%
Text detection	89%
Character segmentation	90%
Character recognition	100%

Another ceiling analysis example:

Facial 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%

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

Summary: Main Topics

- **Supervised Learning:**
  - Linear Regression, Logistic Regression, Neural Networks, Support Vector Machines

- **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