Week 11

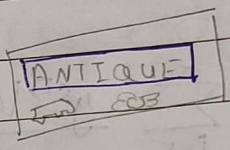
Problem Description and Pipeline

Photo OCR problem.

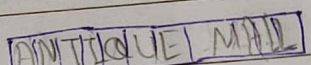
Photo OCR → Photo optical Character Recognition.

☆ Photo OCR pipeline.

1). Text detection.



2). Character Segmentation



3). Character classification. A→A N→N T→T

In 3rd step it may happen Cleaning, which should be cleaning
↳ We are not considering this 4th step for a while.

A system like this is called machine learning pipeline.

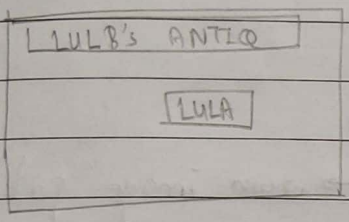
Image → Text detection → Character Segmentation → Character recognition.

Q) Machine learning pipeline refers to
→ A system with many stages/components, several of which may use ml.

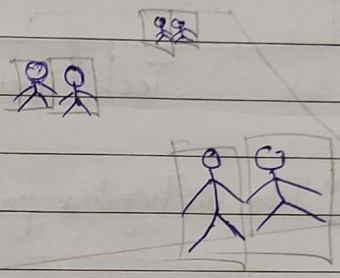
--@ Sliding windows — — — —

#. How invidual component of pipeline works.
. What is sliding windows classifier.

Text detection



Pedestrian detection.

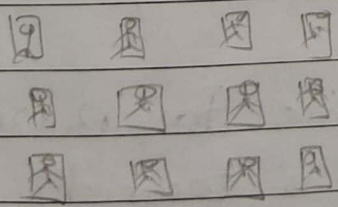


- Aspect ratio is different
- Aspect ratio is same (approx)
- Hight & width of every rectangle may be diffrent.

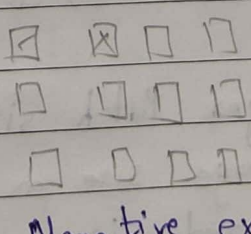
★ Supervised learning for pedestrian detection.

x = pixels in 82 x 36 image patch.

Dataset



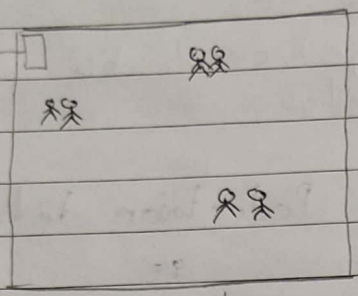
Positive examples (y=1)



Negative examples (y=0)

Image Patch → container of pixels.

* Sliding window detection



Test image.

S1) → Take rectangular patch of this image. → sum image patch (check for pedestrain)

Slide rectangle & do again. In this way slide through whole test image.

↓

by amount?

It is a parameter →

step-size / stride

(performance is best at 1px (expensive) usually 4-8 px)

S2) Now increase patch size.

Because previous image would detect pedestrain of particular size (image patch size) only

while increasing patch size →

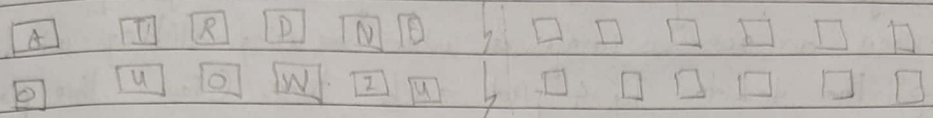
Take larger Patch size, and resize to 82×36 (patch image size on which model is trained)

This resized image will be passed to classifier to decide presence of pedestrain in that patch.

S3) Repeat S1), S2), S3)

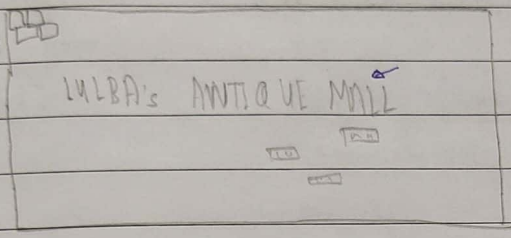
So, at end you get all pedestrians spotted.

☆ Text detection:



Positive examples (y=1) Negative examples (y=0)

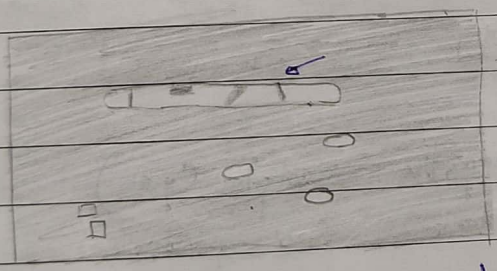
Running sliding window on lots of little patches like this



Test Image.

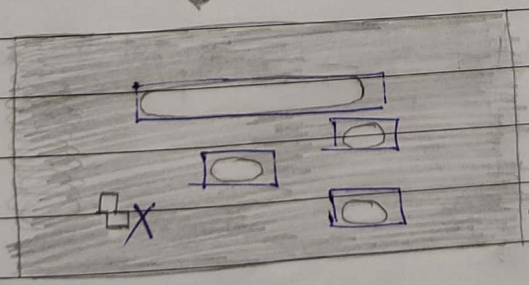
returns image

We use sliding window at just one fixed scale (just for illustration purpose) (one rectangle size)



• White it classifies has found text.
• Shades of grey (probability o/p by classifier)

Expansion operator → Expands white region. [colour some neighbouring pixels as white]



← • Draw ▭ around favourable white space. (aspect ratio must be ▭)

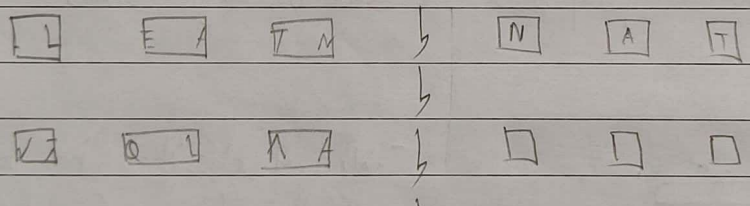
Text must be like ▭ & not like ▯ (aspect ratio)
so, we do not consider (▯) such type of white blobs.
We ouke out vertical text box.

Q1) Suppose you are running a text detector using 200x200 image patch. You run the classifier on 200x200 image. & when using sliding window, you "step" the detector by 4 pixel 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 closest one)

- 100
- 400
- about 2,500 times.
- 40,000

52) character segmentation

- 1D sliding window for character segmentation. look for split b/w two characters.

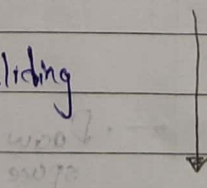


Positive examples (y=1)
There is split

Negative examples (y=0)
There is no split.

ANTIQUE MALL We got this from previous step 5).

Now perform 1D sliding (only 1 row here)



ANTIQUE MALL

-- (Refer Photo OCR pipeline) figure (7 2PTO)

Getting Lots of Data & Artificial Data. — — —

Artificial Data Synthesis. (Specific problem applicable)

Take low biag algo & big data → Peorbomance ↑.

ADS

Creating new Data from Scratch	We already have small training set Amplify training set (small ts → large ts)

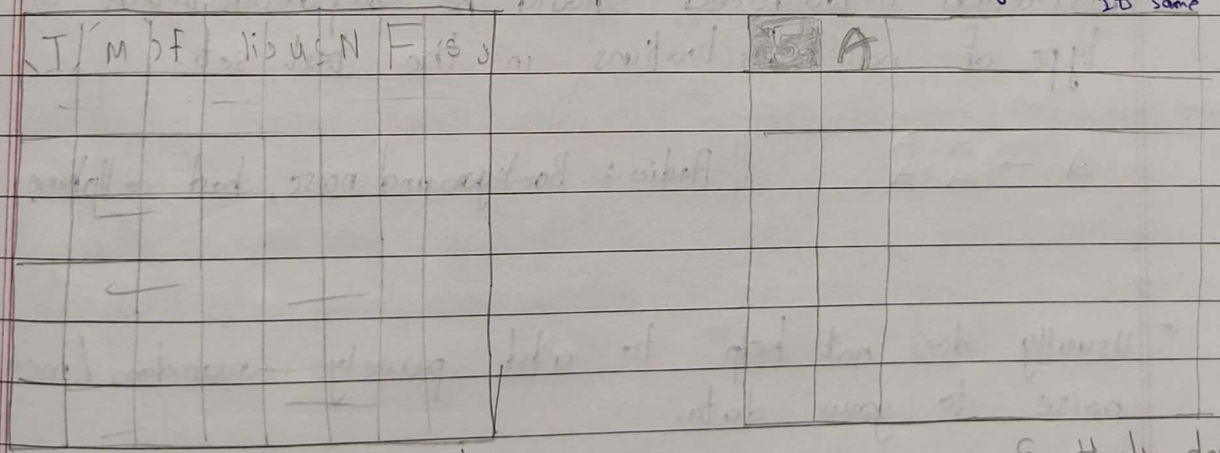
#Character Recognition

A N T I a A

A N T I a A

Artificial Data Synthesis for photo OCR :-

(Coloured image is not useful, its same as grayscale)

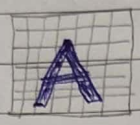


Real data

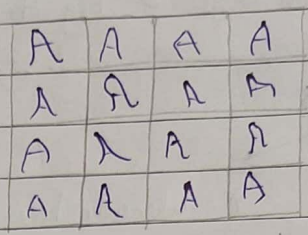
Synthetic data.

Using font liabrary, take fonts & create dataset by giving some random background.

Synthesizing data by introducing distortion: Text recognition.



Actual dataset



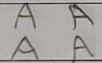
By creating distortion.

Synthesizing data by introducing distortion: Speech recognition.

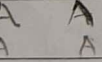
original audio → Audio on bad cellphone connection.
→ Noisy background: Crowd.
→ Noisy background: Machinery.

★ Synthesizing data by introducing distortions.

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

A →  || Audio: Background noise, bad cellphone connection.

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

A → 

x_i = intensity (brightness) of pixel i

$x_i \longleftarrow x_i$ + random noise

Purely random meaningless noise is less likely to be useful.
While adding distortions, think what other meaningful distortions you might add that will cause you to generate additional training examples that are least somewhat representative of the sorts of image you expect to see in your test sets.

Suppose you learning are training a LR with m examples by minimizing. (Linear reg?)

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

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?

→ ✓ No, and in fact you will end up with the same parameters θ as before you duplicate the data.

★ 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/ no of hidden units in neural network until you have a low bias classifier.

2) "How much work would it be to get 10× as much data as we currently have?"
- Artificial data Synthesis
- Collect / label it yourself.
- "Crowd source" (Amazon Mechanical Turk)

--(a) Ceiling Analysis : What part of the Pipeline to work on next.

Ceiling Analysis : Strong advice of what parts of the pipeline might be the best use of your time to work on

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

Image → Text detection → Character Segmentation → Character recognition.

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

	Component	Accuracy
1)	overall system	72%
2)	Text detection	89%
3)	Char ~~recog~~ seg^n	90%
4)	Char recog^n	100%

In development process, to make decisions on what to do for developing the system is going to very helpful to have single valued number ~~ele~~ et evaluation metrix.

Pick character level accuracy:- Given test set image, what is fraction of alphabets or characters in test image that we recognize correctly.
→ or any single valued elevation metrix.

Image → Text detection → Character segmentation → Character recognition.

→ Give correct labels for the text detection part of pipeline Manually. So that I get perfect text detection system on my test set.

→ Run data through rest of pipeline.
(Through char seg. & char. recg?)

Text detection 89%. (Increased overall) ← elevation metric now overall

Image → Text detection → Character segmentation → Character recognition.

both manually supplied.

character segmentation 90%.

Image → Text detection → character segmentation → character recognition.

Everything manually supplied.

character recognition 100% (obviously)

1) 2) (17%) worthy to work on 2)

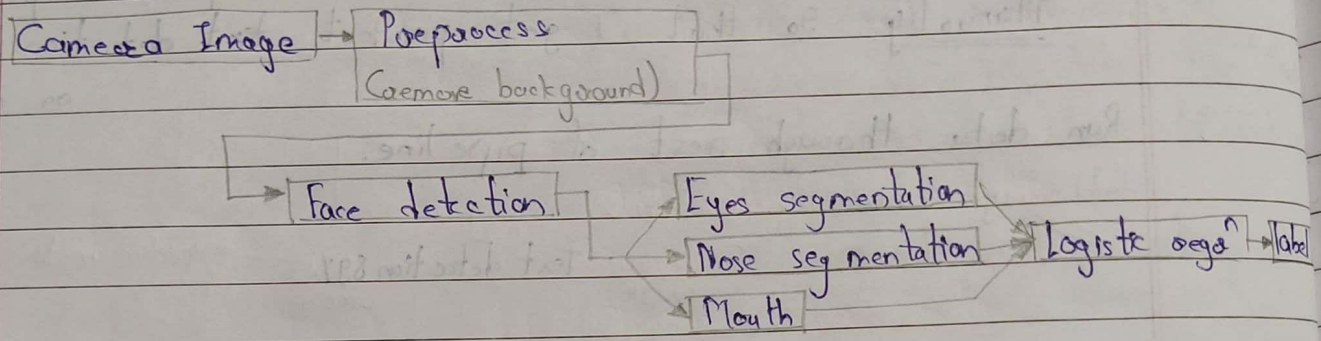
2) 3) (1%) not worthy to work.

3) 4) (10%) worthy

We have upside potential of improving each component (Anylasis provides)

★ Another ceiling analysis example.

Face recognition from images (Artificial example)



Component	Accuracy.	
overall system	85%	
		0.1%
Preprocess	85.1%	
		5.9%
Face detection	91%	
		4%
Eyes	95%	1%
Nose	96%	1%
Mouth	97%	
		3%
LR	100%	

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

→ • 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 10× 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.

Instead of going by gut feeling, do ceiling analysis.

-- (a) Summary & Thank you.

Summary: Plain topics.

Supervised learning: Linear regn; logistic regn; neural networks, SVMs.

Unsupervised learning: K-means, PCA; Anomaly detection.

Special applications/special topics: Recommender system, large scale machine learning.

Advice on building a machine learning system.
Bias/variance, regularization; deciding what to work on next: evaluation of learning algorithm, learning curves, error analysis, ceiling analysis.