Week 11

Application example: photo OCR. → problem description and pipeline

Photo OCR pipeline → sliding windows

1. text detection. (1-5 engineers)
2. character segmentation. (1-5 engs)
3. character classification (1-5 etc)
4. (spelling correction)

Sliding Windows

Supervised learning for pedestrian detection

x = pixels in 82 x 36 image patches

positive examples ($y=1$) | negative examples ($y=0$)

→1) run patch through classifier

2) slide rectangle over a bit, (shifting amount (step-size / stride))

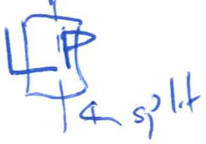
3) increase image patch size → resize down to 82×36

→ take output of classifier → expand it; if a character is detected within the vicinity of another character → can conclude that it is text

↳ use heuristics to look for "long rectangles"

Character segmentation → look for splits

spits = $y=1$ | no split $y=0$



Week 11

Application example: photo OCR: getting lots of data: artificial data synthesis

'one of the best things to do in ML.
take a low bias algorithm, → train on a massive data set.

synthesize data
- by making more examples using real fonts
- distort existing examples to create more data

can do the same with audio say.

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

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

Discussion on getting more data

1) Make sure you have a low bias classifier before expending 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. → how many hours will it take to collect 10x as many examples
  m = 1000 → 10,000
  10 seconds/example
- "crowd source"
  → e.g amazon mechanical turk.

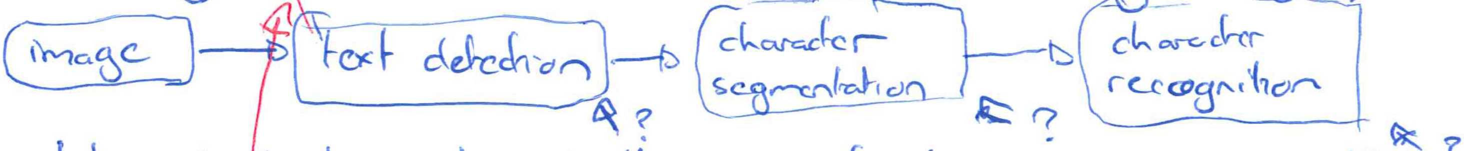
Week 11

Application example: photo OCR.

: Ceiling analysis, what part of the pipeline to work on next?

→ simulate 100% accuracy

estimating the errors due to each component (ceiling analysis)



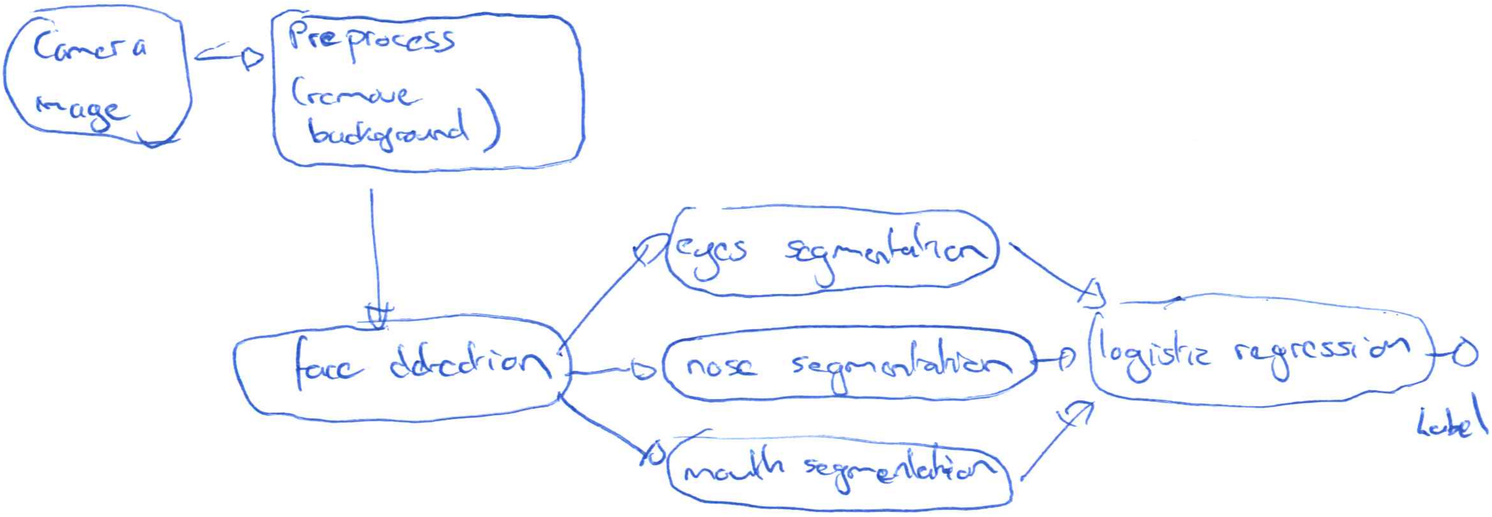
what part of the pipeline should you spend the most time trying to improve? → where should you allocate scarce resources?

Component	Accuracy	
Overall system	72%	
text detection	89%	17%
character seg.	90%	[illegible]%
character recog.	100%	10%

provide it the correct text detection output → text detection

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