Sliding Windows

How do the individual models work? Here we'll focus on a sliding windows classifier

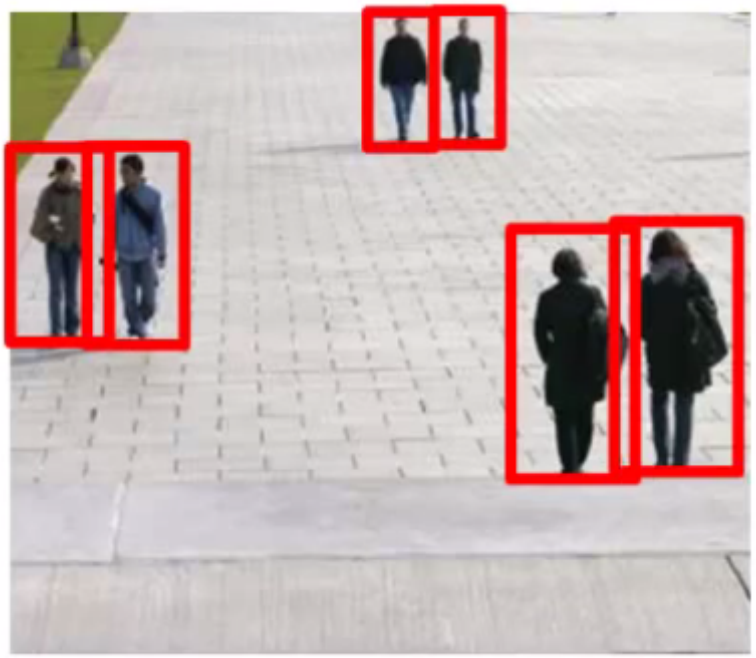
Text detection example

The first stage of the filter was the **text detection** where we look at an image and we try to find the regions of text that appear in this image.

- Text detection is an inusual problem in computer vision
  - Because depending on the length of the text we're trying to find, these rectangles (which surround text) that we're trying to find can have **different aspect ratio** (aspect ratio being **height : width**).

Pedestrian detection (simpler example)

So in order to talk about detecting things in images, let's start with a simpler example of pedestrian detection. Want to take an image and find pedestrians in the image:

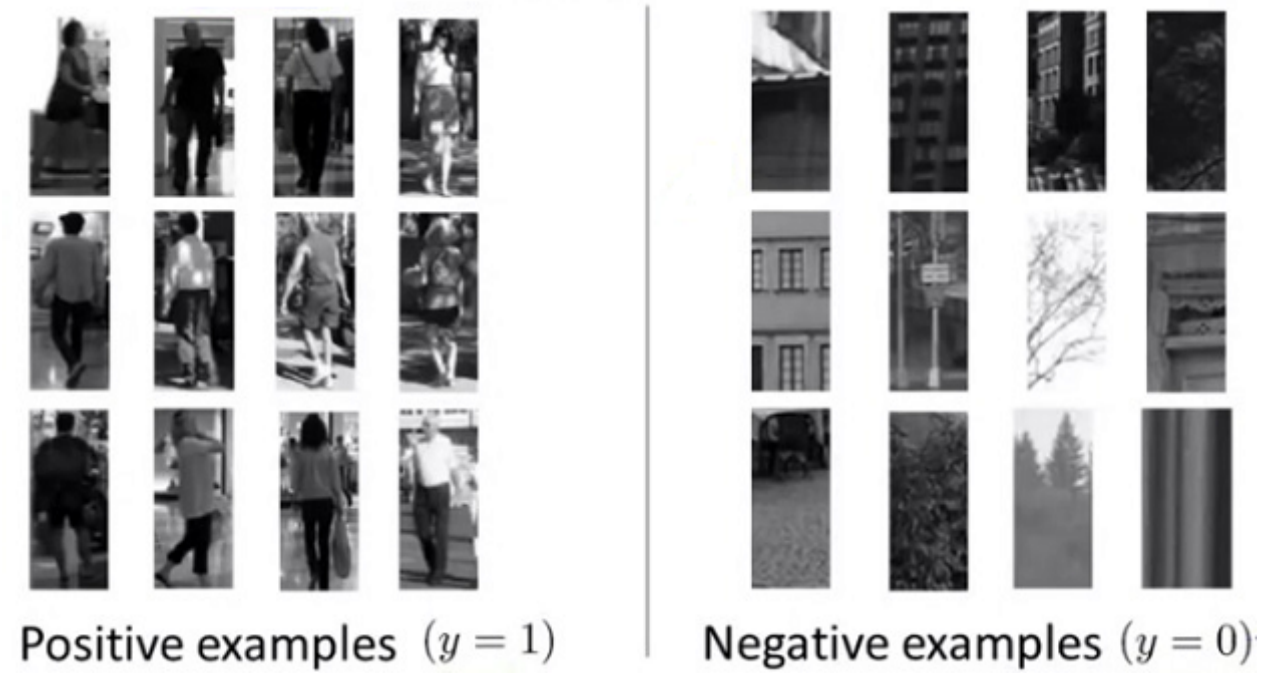


This is a slightly simpler problem because the aspect ration remains pretty constant

Supervised learning for pedestrian detection

- Building our detection system
- x = pixels in 82 x 36 image patches
  - This is a typical aspect ratio for a standing human

We'll Collect training set of positive and negative examples:

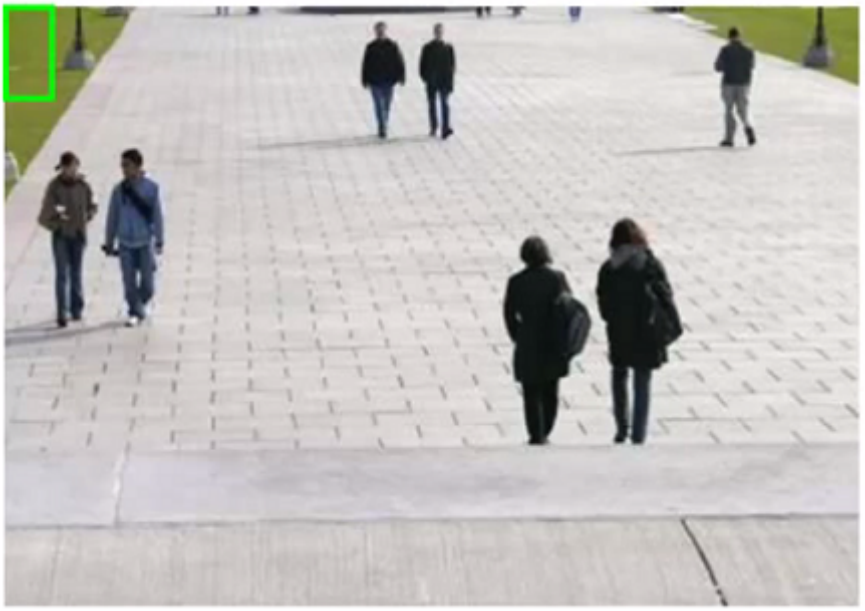


In a more typical pedestrian detection application, we may have anywhere from a 1,000 training examples up to maybe 10,000 training examples, or even more if we can get even larger training sets.

- What we can do is a **train a neural network** (or some other learning algorithm) to take an image and classify that image as pedestrian or not
  - So this gives us a way of **applying supervised learning** in order to take an image patch **can determine whether or not a pedestrian appears in that image patch**.

Now let's sat we get a new image - **how do we find pedestrians in it?**

- **What we can do is starting by taking a rectangular 82 x 36 patch in the image**



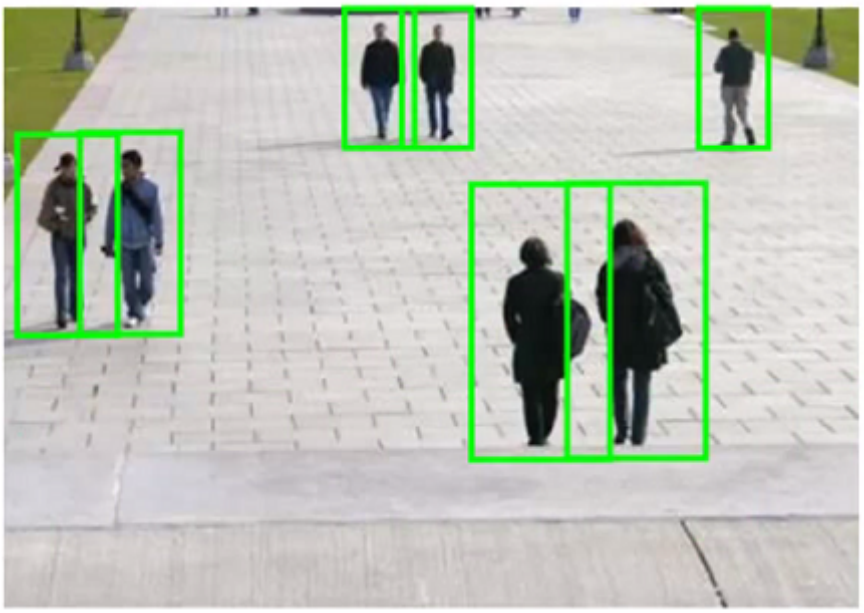
- **We'll run the patch through classifier - hopefully in this example it will return y = 0**
- Next slide the rectangle over to the right a little bit and re-run, then slide again, and so on...
  - The amount we slide each rectangle over is a parameter called the **step-size** or **stride**

    - Could use 1 pixel (**More commonly 5-8 pixels used**)
    - Best, but **computationally expensive**
- So, keep stepping rectangle along all the way to the right
  - Eventually get to the end
- **Then move back to the left hand side but step down a bit too**
  - We'll repeat that until **we've covered the whole image**

Now, that was a pretty small rectangle, that would **only detect pedestrians of one specific size**. What we do next is start to look at **larger image patches**:

- So now we can take a larger image patch (of the same aspect ratio)
- Each time we process the image patch, **we're resizing the larger patch to a smaller image**, then running that smaller image through the classifier

Hopefully, by changing the patch size and rastering repeatedly across the image, we eventually recognize all the pedestrians in the picture!.

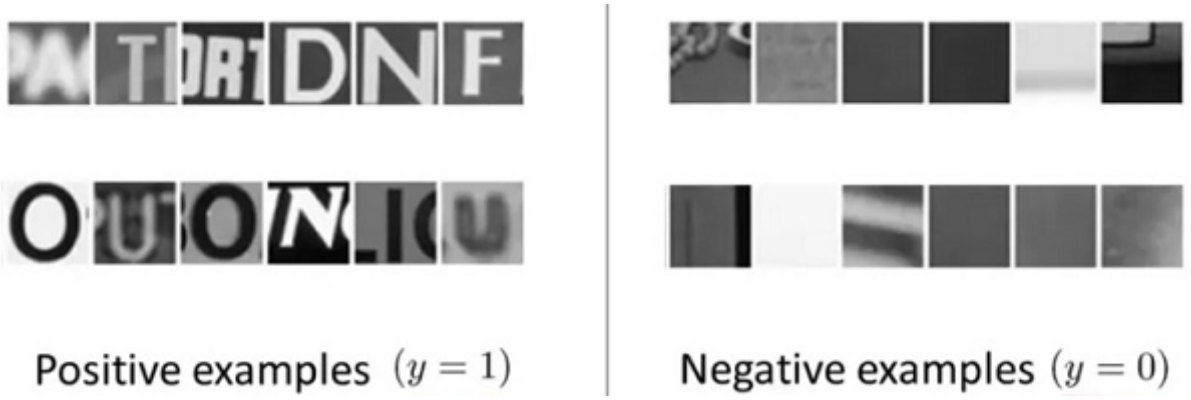


So that's how we train a supervised learning classifier, and then use a sliding windows classifier, or use a sliding windows detector in order to find pedestrians in the image.

Text detection example

Like pedestrian detection, we generate a labeled training set with:

- **Positive examples** (patches of images where there is text)
- **Negative examples** (patches of images where there isn't text)



Having trained the classifier we apply it to an image

So let's run a sliding window classifier at a fixed rectangle size (once we having trained the classifier), if we do that end up with something like this:

White region show where text detection system thinks text is

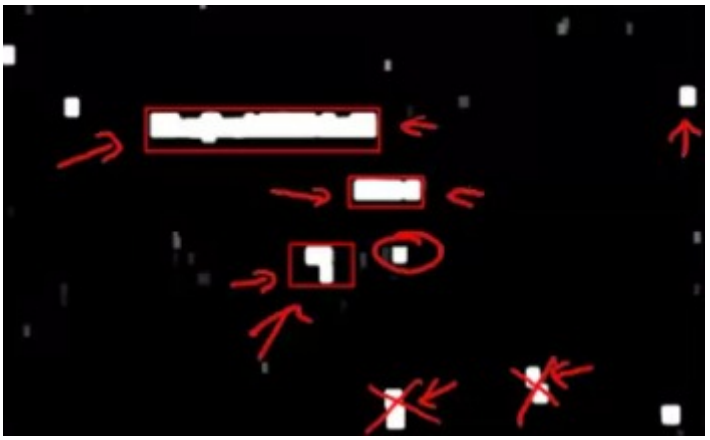
- Different shades of gray correspond to probability associated with how sure the classifier is the section contains text
- Black - **text detection system thinks that there's no text**
- White - **text detection system thinks that there's text**

Concretely, what we have done on this image is actually use white to show where the classifier thinks it has found text and different shades of grey correspond to the probability that was output by the classifier.

So like the shades of grey corresponds to where it thinks, it might have found text but has lower confidence the bright white response to whether the classifier, up with a very high probability, estimated probability of there are text.

For text detection, **we want to draw rectangles around all the regions where there is text in the image:**

- So, we'll take classifier output and apply an **expansion algorithm**
  - Takes each of white regions and **expands it**
- **How do we implement this?**
  - Say, for every pixel, is it within some distance of a white pixel?
  - If yes then **colour it white**



Look at connected white regions in the image above:

- Draw rectangles **around those which make sense as text** (i.e. tall thin boxes don't make sense)
- This example misses a piece of text on the door because **the aspect ratio is wrong or very hard to read.**

So that's text detection using sliding windows and having found the rectangles with the text in it, we can now just cut out the image regions and then use later stages of pipeline to try to meet the texts.

Video Question: 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).

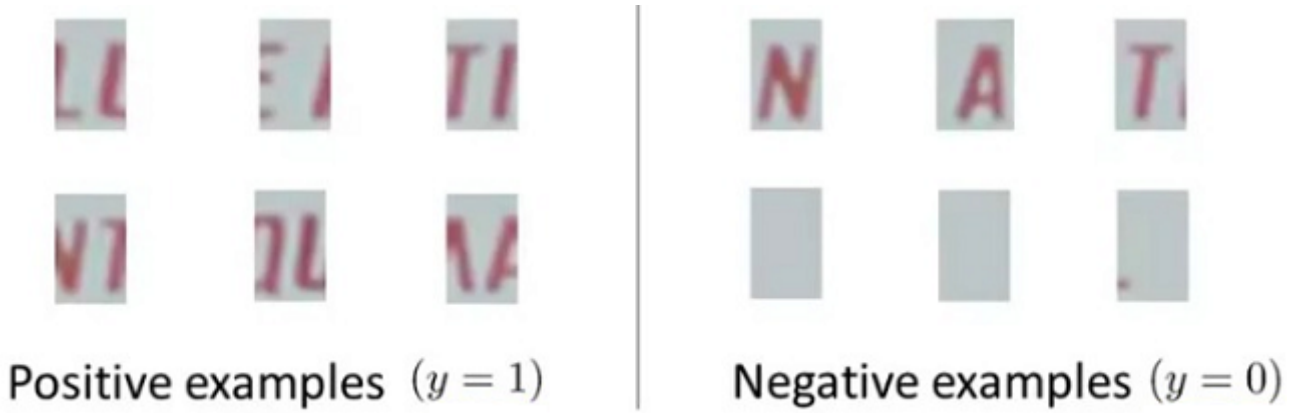
- About 100 times.
- About 400 times.

> About 2,500 times.

- About 40,000 times.

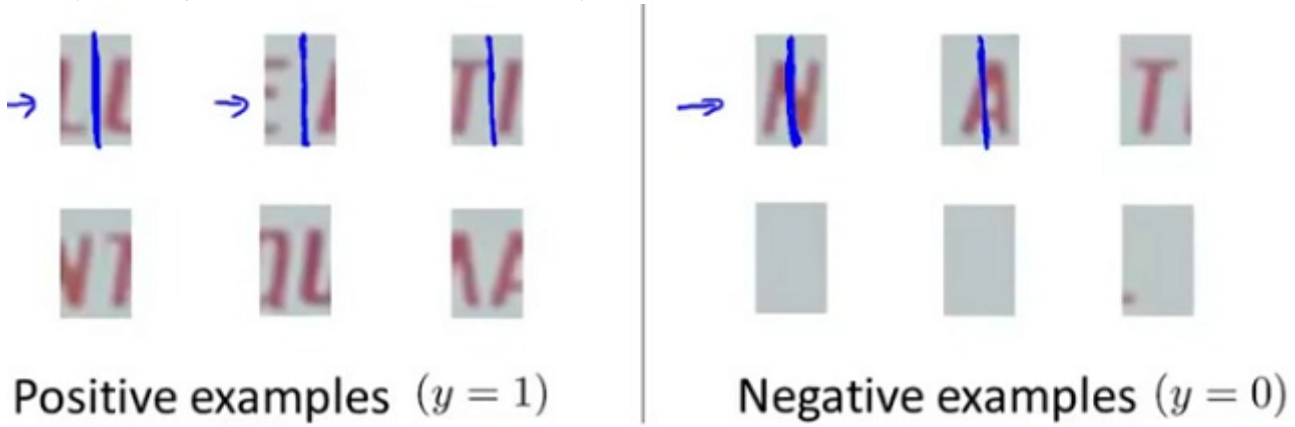
1D Sliding window for character segmentation

We recall that the second stage of pipeline was **character segmentation**, so given an image like shown on top, **how do we segment out the individual characters in this image?**. So what we will do is again **use a supervised learning algorithm** with some set of positive and some set of negative examples



Look in a defined image patch and decide, **is there a split between two characters?**

- So, for example, our first training data item below looks like there is such a split
- Similarly, the negative examples are either empty or hold a full characters



So what we will do is, **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. Having trained such a classifier, we can then **run this on the previous sort of text that our text detection system has pulled out.**

- Use a **1-dimensional sliding window to move along text regions**
  - **Does each window snapshot look like the split between two characters?**
    - If yes insert a split

    - If not move on
- **So we have something that looks like this:**