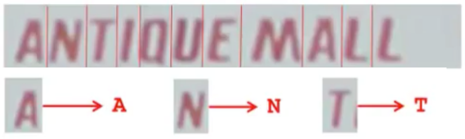
Photo OCR

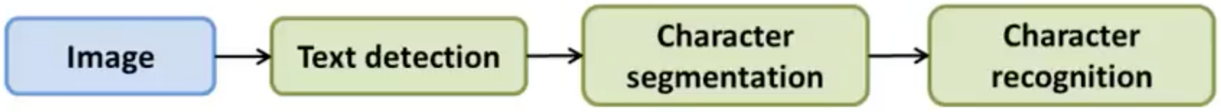
Photo OCR Pipeline

Photo OCR is stand for photo optical character recognition. How can we get the computer reads the text in image and understand the picture? For reading text in image, algorithm has go through the text regions and then reads it correctly. OCR is pretty hard machine learning problems from scanned photographer. Here is the photo OCR Pipeline:

- Text detection
- Character segmentation
- Character classification
- Spell correction, if it detects c1ean to be clean (for now, let's ignore it)



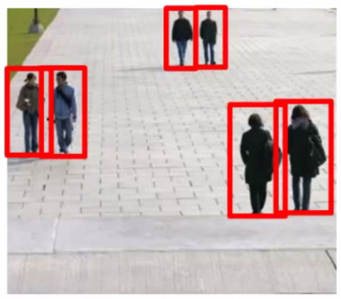
A system like this is called machine leaning, OCR pipeline. Each of this segment, may be machine learning component or may not. These set of modules needs to act one after the other.



Performance of pipeline and each module often has a big impact on the overall performance a problem. Usually around 1-5 engineers can work on each module.

Sliding Windows

Stage 1 is text detection. There are some problems in computer vision such as image angle, long or short text, slanted text and etc. Let's start with a simple example, the pedestrian detection. We want to predict the humans. The aspect ratio of height to width is similar for this case, which makes it easier than text detection. Suppose we have 82 x 36 image patches. Patches includes pedestrians are positive examples (y=1), and the rest are negative examples y=0. The training set is around 10,000. We may use neural network to detect whether the patch contains human or not (using supervised learning). For a testing dataset, we start taking a patch in the image and run the patch through classifier. Then slide the patch over the image and do the analysis again.



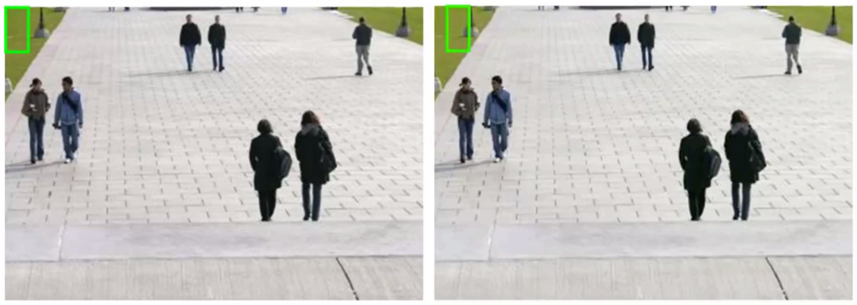
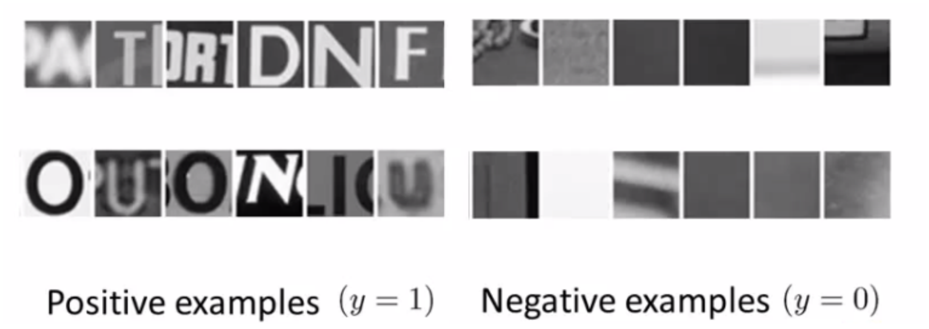


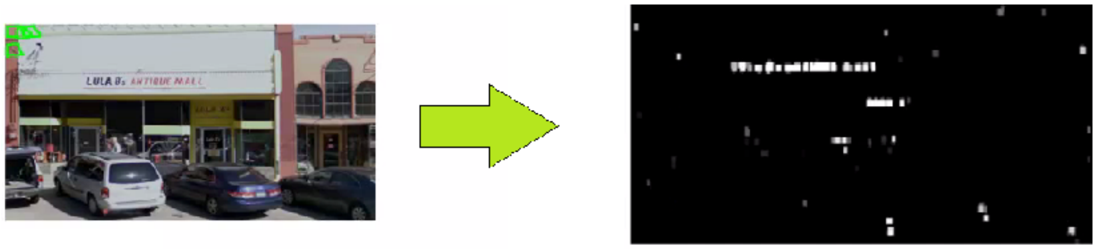
Photo OCR

The amount the window is shift is called the step size or stride parameter. If step size is 1 pixel, it's great but computationally it's very expensive. Next step is increasing the patch size and resize it to smaller image (82 by 36 in our example) till we detect the pedestrians in image.

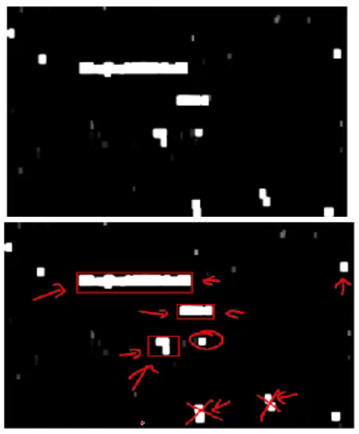
Now, let's get back to text detection, where we have positive examples, text (y=1), and negative examples (y=0).



Next, we can apply it to the test image. We use sliding windows on the image, and we end up with image on the right, showing the white dot, where we found the text. The axis of these image below is same.



We want to draw rectangular all around the regions where there's a text. We can apply *expansion algorithm* after the text classifier. It will expand each of the white regions. For every pixel, if it's within certain region of the white pixel, it colors it white.



Then we draw a rectangular region around the connected areas. For text, we know the width is bigger than height, therefore we discard the rectangular that has high aspect ratio of height to width. This algorithm, misses a small piece written in a door of the store, since we discard it due to aspect ratio.

Next stage is character segmentation. We can use supervised learning. We look if there is any split between two patches of character.

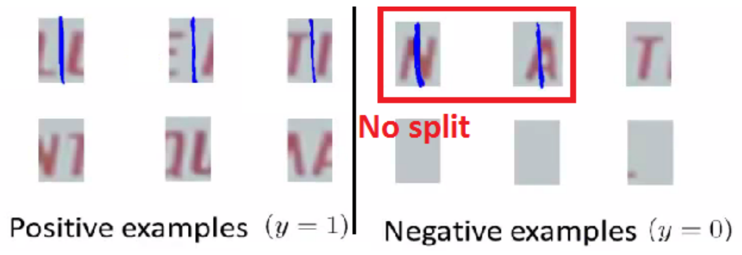
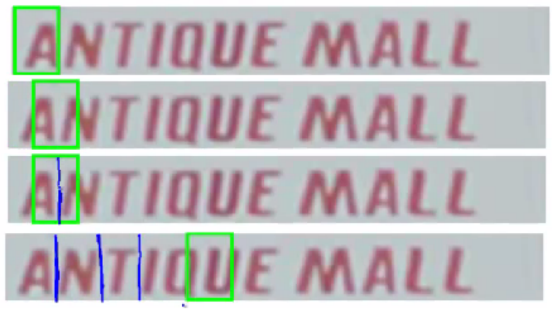


Photo OCR

We may use neural network to classify between the character segmentation. This is a one-dimensional sliding window, since we only move it in x direction because we only have text here. In classifier, we want it split it right in neural network algorithm.



Character classification also can be done using neural network, similar to digit recognition method discussed in previous lesson.

Artificial Data Analysis

One of the most reliable ways to get a high-performance machine learning system is to take a low bias algorithm and train on a massive data set. We can create artificial dataset either from scratch or amplify the small training set into a larger training set. Suppose, we are doing character recognition in OCR in grey scale image patches. So, one way in order to create more training set, we can use different fonts in random ways in various of background. Now, we have a massive training images. Here, we made artificial training set from scratch.

Abcdefg
Abcdefg
Abcdefg
Abcdefg
Abcdefg

Synthetic data

Another approach is to synthesize data using our small dataset by adding distortions. One image of character can give like 16 images of patches and new dataset. Another example for speech recognition, we can add audio on bad connections, add noisy background to create new training dataset. Be careful after applying those changes, the result is making sense and you might see in training dataset.

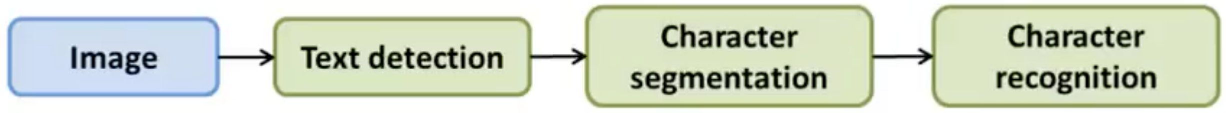
Before getting more data:

- Make sure we have low bias system before expanding the dataset.
    - Plot learning curve
- If not low bias system, add more features
    - Then we can use artificial training dataset.
- Answer this question before starting: How much work would it be to get 10X training dataset?
    - Artificial data synthesis
    - Collect/label yourself
    - Crowd sourcing: Like amazon mechanical Turk

Photo OCR

Ceiling analysis

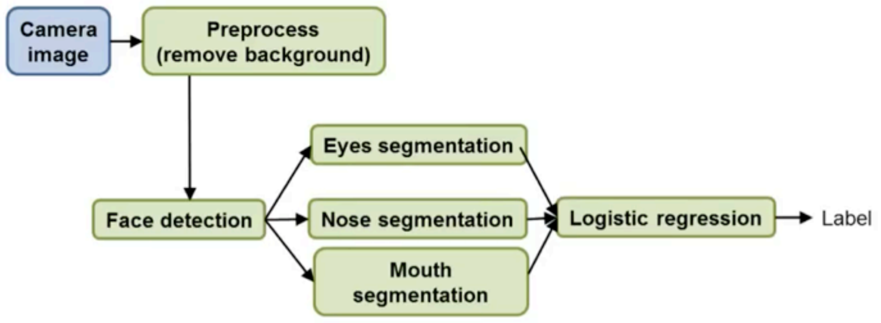
What part of pipeline is best to spend more time to work in order to improve the system? Single number for evaluation metric is good to have for our decision. Imagine our initial pipeline in a course, the overall accuracy to be 72%. In each of the test examples in pipeline, we should estimate the accuracy. For example, accuracy of text detection is 89%.



Then we are going to give 100% correct values (manually correct the wrong predictions) for test set in text detection input to the next stage of character segmentation (we are feeding correct data to the next stage) and so on. Final output from character recognition is the accuracy of the system. Correcting the values in feeding it to the next stage increases the overall accuracy of the system. Based on the accuracy we get in a way we described, accuracy increases more from initial overall curacy to text detection (17%). This analysis shows the upside potential of improving each side of components. We can learn how much we can gain by improving each of pipeline segments.

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

Another example is face detection. We need to do the process shown below:



We can investigate the accuracy same as above and see how each part effects the overall performance of the system.