10.02 - Application Example: Photo OCR

Figure 1: Photo OCR pipeline

1 Problem Description and Pipeline

Photo OCR = Photo Optical Character Recognizion. This problem focuses on how to get computers to read text in pictures that we take.

1.1 Pipeline

1. Text detection - need to find where text is located in the photo.
2. Character segmentation - divide the photo rectangle into different letters.
3. Character classification - see which letter is which.

We can see the Machine Learning pipeline in fig. 1. One of the principal steps when building a ML system, it's very important to split the model into modules so different people can work on different things.

2 Sliding Windows

2.1 Text Detection

First step of the pipeline is *text detection*.

We can first explain how to build a pedestrian detection system (easier) and then incorporate the ideas into the text detection module.

What we do is take some standarized (same size) images where there are pedestrians and some where there are not (fig. 2) and train with that. What we can do is then take that data and classify it as a supervised learning problem.

When we are predicting from an image, we simply take a rectangle and shift it around the image, testing each time whether there is a pedestrian there or not, as seen in fig. 3. We start with a small window size, move it around the image and then try larger image patches (we do this by actually making the img smaller, not the window larger).

When doing text detection, we do a similar thing. We also use positive examples and negative examples (fig. 4) to build up a supervised learning environment.

We then apply the classifier using the sliding window technique and collect the results. We then use computer vision techniques: use expansion on the collected results image. Afterwards, we take advantage of knowing that text boxes should have width much higher than height and discard those that don't satisfy this characteristic. We can then just place bounding boxes around the rectangles that are left, and finally cut those image regions. We can see an example of this process in fig. 5.

2.2 Character Segmentation

For separating the individual characters inside an image, we can use a similar technique than when we were doing text detection.

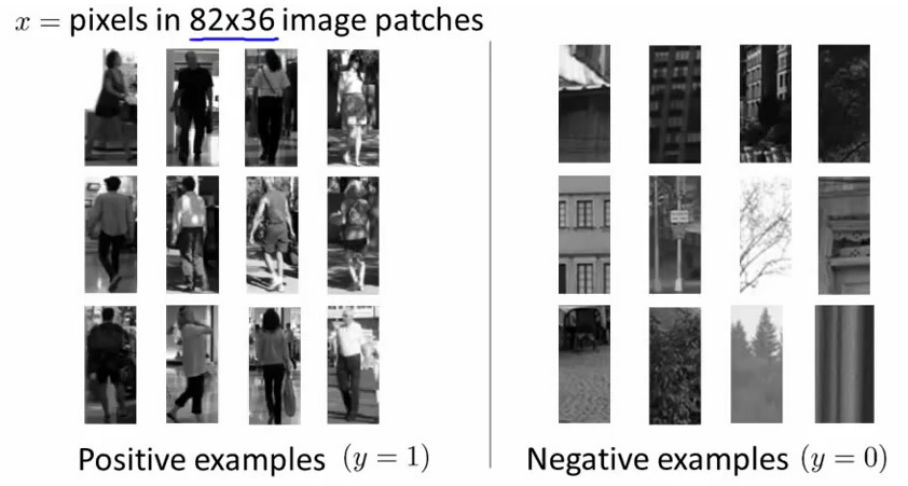

Figure 2: Data used for pedestrian detection

Figure 3: Example of how to apply a sliding window when recognizing pedestrians

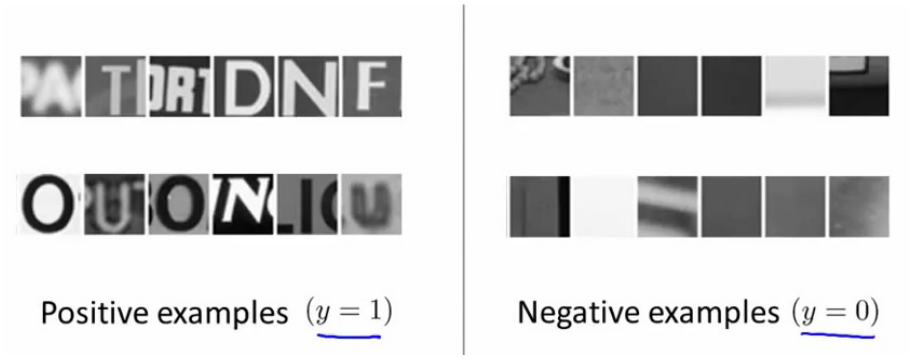

Figure 4: Data used for text detection

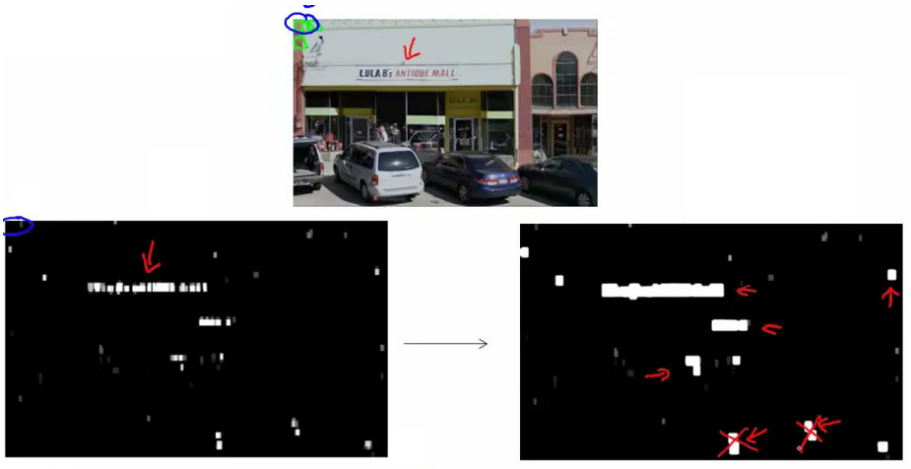

Figure 5: Text detection steps

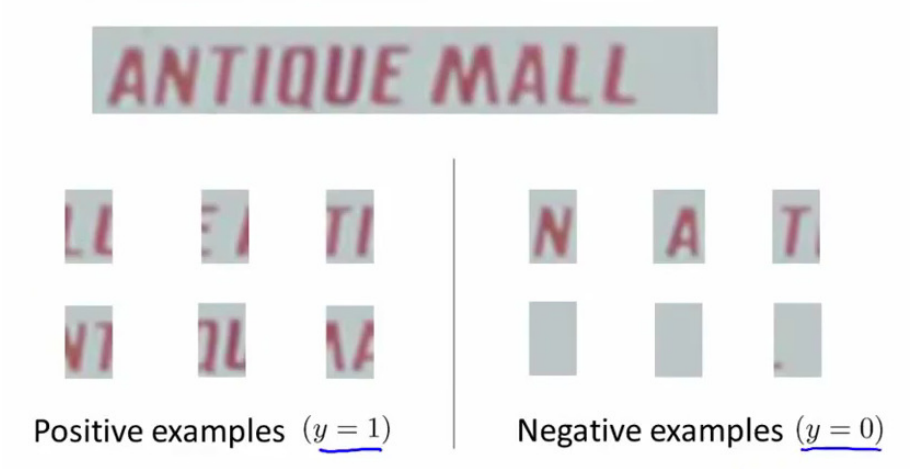

Figure 6: Data used for training a character classifier

Figure 7: Results of applying 1D sliding window for character segmentation

We can just take positive and negative examples and train a classifier with that, as seen in fig. 6. The only difference is that now we try to find whether there is a separation between two characters as the true label. After training the classifier, we use a 1D sliding window (only from left to right) and collect the results (which will be the splits), as we can see in fig. 7.

2.3 Character Classification

The last step of the pipeline is just to use the previously done classification task so we know which letter corresponds to each of the split symbols.

3 Getting Lots of Data and Artificial Data

Artificial Data Synthesis = either create data from scratch or amplify an already existing dataset.

For example, on the character recognition task, we can synthesize data to help us learn better. One way to create data from scratch would be to simply take existing fonts and throw them to a random background similar to the ones from real data. We can also rotate or blur them to make them more realistic. This is a very creative job, as we can see. An example to amplify the existing dataset is to introduce small distortions on the existing letter examples (fig. 8).

It usually does not help to add purely random/meaningless noise to the data. We need to only add noise that is expected on the test set.

If we had a speech recognition task we could add for example crowd on the background to the clean data examples.

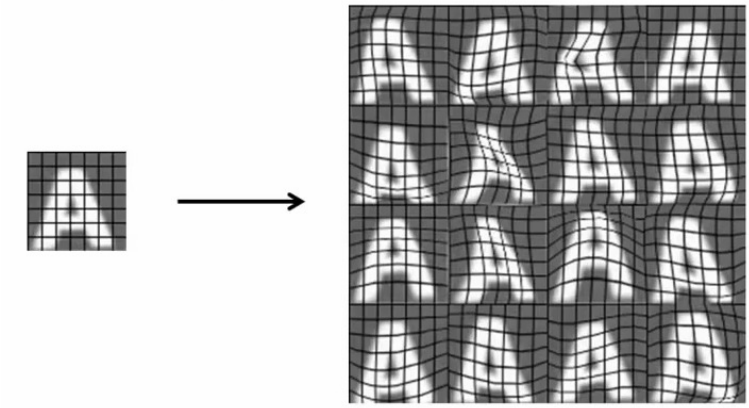

Figure 8: Synthesizing data via real data amplification using distortions

3.1 Discussion on getting more data

1. Make sure you have a low bias classifier (plot learning curves to know that). For example: keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier. If we don't do this, getting more data isn't likely to help.

2. Ask yourself: "How much work would it be to get 10x as much data as we currently have?" If the answer is reasonable, consider doing that.

3. Thing about crowd sourcing the data labeling process. Buy hours from people, basically. Example: Amazon Mechanical Turk.

4 Ceiling Analysis: What Part of the Pipeline to Work on Next

When developing on a ML system, one of the most valuable resources is the time of the developers. We need to know what to prioritize when working on a pipeline.

We will continue taking a look on the OCR pipeline (fig. 1). Where to alocate resources (time)? The first step is to get a number to evaluate the whole system (accuracy, recall, etc.). We can then evaluate every single module using the same score, using as test data manually selected ground truth. This is basically giving the correct answers to the algorithm, as if the other parts of the pipeline had a 100% score. Doing this consecutively we might end up with the results seen in fig. 9. In this example, we see that the part that needs more work is text detection, as the drop is a 17%, much higher than the other parts.

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

Figure 9: Photo OCR results