Object detection with a linear model

In this exercise we develop a linear model for object detection in images. Specifically, we will train the model to detect the area of an image in which a hand is shown to gesture a sign language sign.

We will develop a linear model that, given an image $\mathbf{x}$ (an array of pixels), produces a bounding box $\mathbf{y}$. The bounding box is a rectangle defined by its top-left and bottom-right corners. Hence $\mathbf{y}=(y_1, y_2, y_3, y_4)$. The detected object -- a hand gesturing a sign -- is to be inside this bounding box.

Data. This is a supervised learning problem, as we have a set of images with their corresponding bounding boxes. We will use a sign language dataset that includes both images and bounding box examples. The data is available as a zip file at https://github.com/yoavram/Sign-Language/raw/master/Dataset.zip. After extracting the zip file to the folder *data/sign-lang*, we have 7 folders, one for each person (user). Each folder has 10 images of that person gesturing one of 24 signs: the ABC letters, not including J and Z. This is the file structure (the hyperlinks don't lead anywhere):

(see in the next page)

```
├── user_3
│   ├── A0.jpg
│   ├── A1.jpg
│   ├── A2.jpg
│   ├── ...
│   ├── B0.jpg
│   ├── B1.jpg
│   ├── B2.jpg
│   ├── ...
│   ├── Y8.jpg
│   ├── Y9.jpg
│   └── user_3_loc.csv
├── user_4
│   ├── A0.jpg
│   ├── A1.jpg
│   ├── A2.jpg
│   ├── ...
│   ├── Y8.jpg
│   ├── Y9.jpg
│   └── user_3_loc.csv
...
└── user_10
    ├── A0.jpg
    ├── A1.jpg
    ├── A2.jpg
    ├── ...
    ├── Y8.jpg
    ├── Y9.jpg
    └── user_10_loc.csv
```

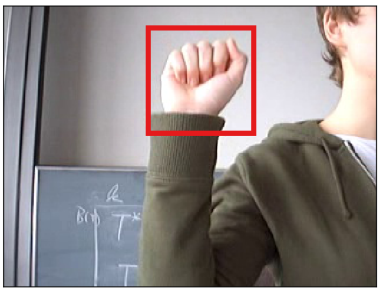
The *jpg* files are images, for example these are *user_3/A0.jpg* and *user_9/K7.jpg*:


The image name has the sign (A and K in these examples) and the repetition number.

In addition, each folder contains a metadata file; for example, the first 5 rows in *user_3/user_3_loc.csv* are

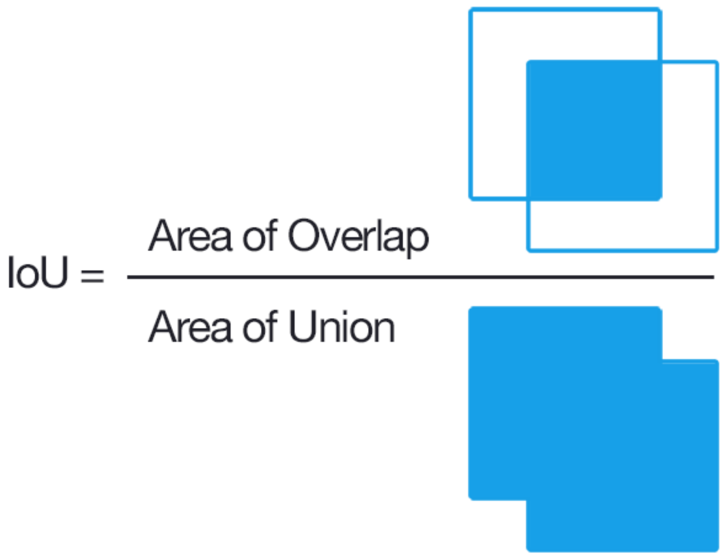
	image	top_left_x	top_left_y	bottom_right_x	bottom_right_y
0	user_3/A0.jpg	124	18	214	108
1	user_3/A1.jpg	124	18	214	108
2	user_3/A2.jpg	123	19	213	109
3	user_3/A3.jpg	122	21	212	111
4	user_3/A4.jpg	122	20	212	110

This table has 5 columns. The first column **image** provides the name of the image. The other columns provide the **bounding box** *y* for that image. For example, the bounding box for *user_3/A0.jpg* is a box with corners at (124, 18) (214, 108):



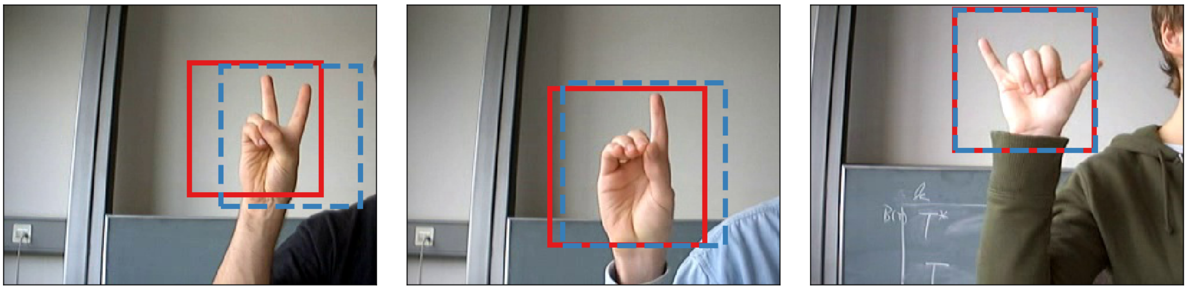
Goal. We want to collect all images into an array *X* and all bounding boxes into an array *Y*. Then, we want to train a model that, given an example image *x* produces the bounding box *y* for that image, such that the bounding box will contain a gesturing hand.
We would like to have a simple, rather than complex model: if we can produce good results with a linear model, than we prefer them over neural networks.
We also need to evaluate and visualize our model. Visualization is easy enough – we plot the image, together with the true bounding box and the predicted bounding box. To evaluate the model, however, we need to define a metric called IoU (intersection over union): the ratio of the intersection and the union of the true and predicted bounding boxes. The intuition is given by this illustration:

The definition is illustrated here:



You can find more details on IoU, including an implementation, [here](). However, you should implement IoU using NumPy such that it works on arrays of images, rather than just using the implementation in the link. Hint: use *np.minimum* and *np.maximum*.

Results. First, **train a generalized linear model** on a training set.
Second, **Visualize the model performance** by plotting the image, the original bounding box, and the predicted bounding box, for example:



Plot the predictions for 9 random images from the test set and print the real and predicted bounding boxes, together with the IoU.
Third, **summarize model performance** by computing the IoU over all test images, plotting the IoU histogram, together with the mean, the 5th percentile, and the 95th percentile. Also **print** the mean and the percentiles.