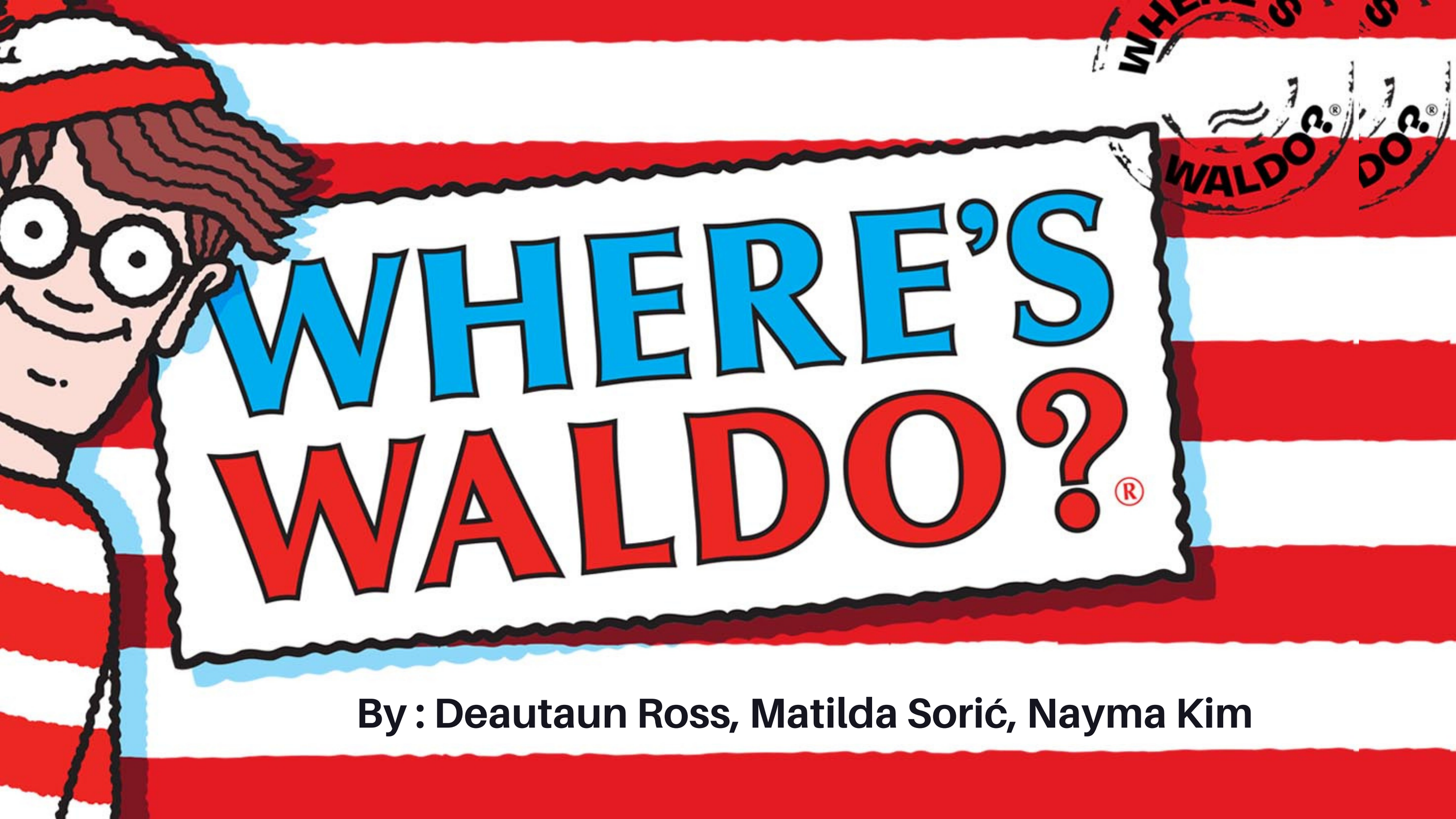


By : Deautaun Ross, Matilda Sorić, Nayma Kim

Outline

Project Goal

Data

Analyzing

Model

Conclusions

Project Goal

Find Waldo with
Image Recognition

Data

Images

- Waldo
- Not Waldo
- Original Images

From

- Found on Kaggle

Kaggle Link

File Size

376.93 MB

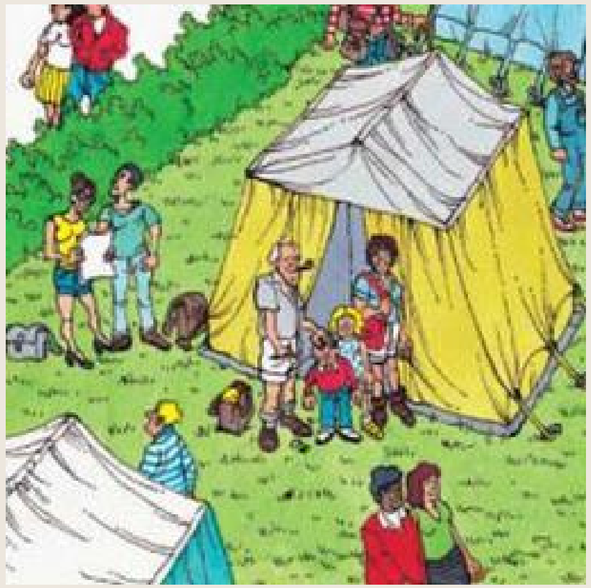
Analyzing

Waldo

39 images

Not Waldo

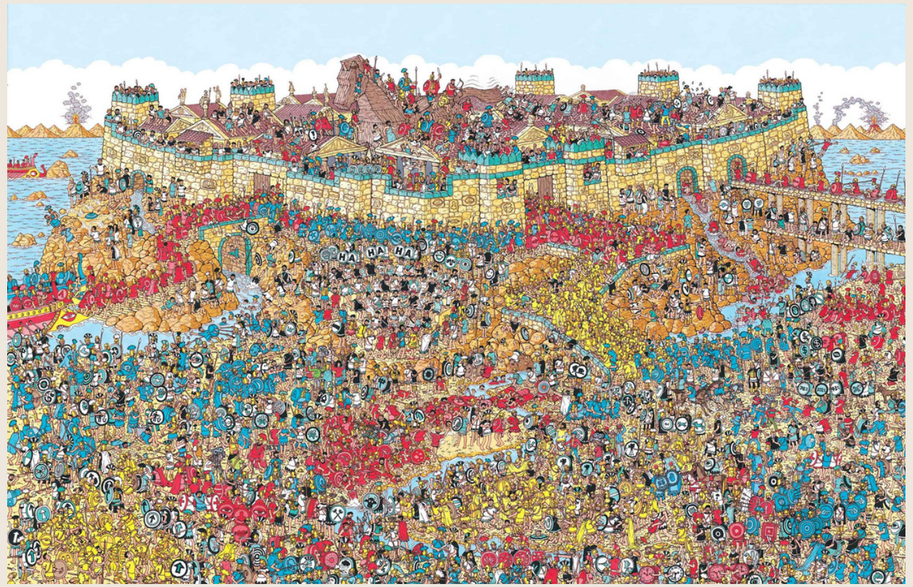
~6000 images



Original Images

19 Images

containing Waldo



Uploading and Converting Images

- cv2.imread to uploaded our images
- Converted into arrays of numbers (0-256)

```
[[[ 25  26  12]
  [ 25  26  12]
  [ 25  26  12]
  ...
  [113 106  52]
  [113 106  52]
  [114 107  53]]
```

Annotating an Image

- Find bonding values on Waldo
- Annotating box
- Saving image
- Saving array of image into a csv file

Model

Image Classification

- Classifying images (Waldo =1, Not Waldo=0)
- Train and Test split
- Normalized Data
- Weighted Binary Crossentropy

Image Classification

Model Building

```
NUM_UNITS = 64
WINDOW_SHAPE = (3, 3)

model = Sequential()

# 2 x 64 convolutional neural network
model.add(Conv2D(NUM_UNITS, WINDOW_SHAPE, input_shape=X_train.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(NUM_UNITS, WINDOW_SHAPE))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

# flatten to 1D
model.add(Flatten())
model.add(Dense(64))

# output layer
model.add(Dense(1))
model.add(Activation("sigmoid"))

# compliling model
# create_weighted_binary_crossentropy(zero_weight, one_weight)
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy", tf.keras.metrics.Recall()], loss_weights=[zero_weight, one_weight])  # "binary_crossentropy"
```

Image Classification Model Summary

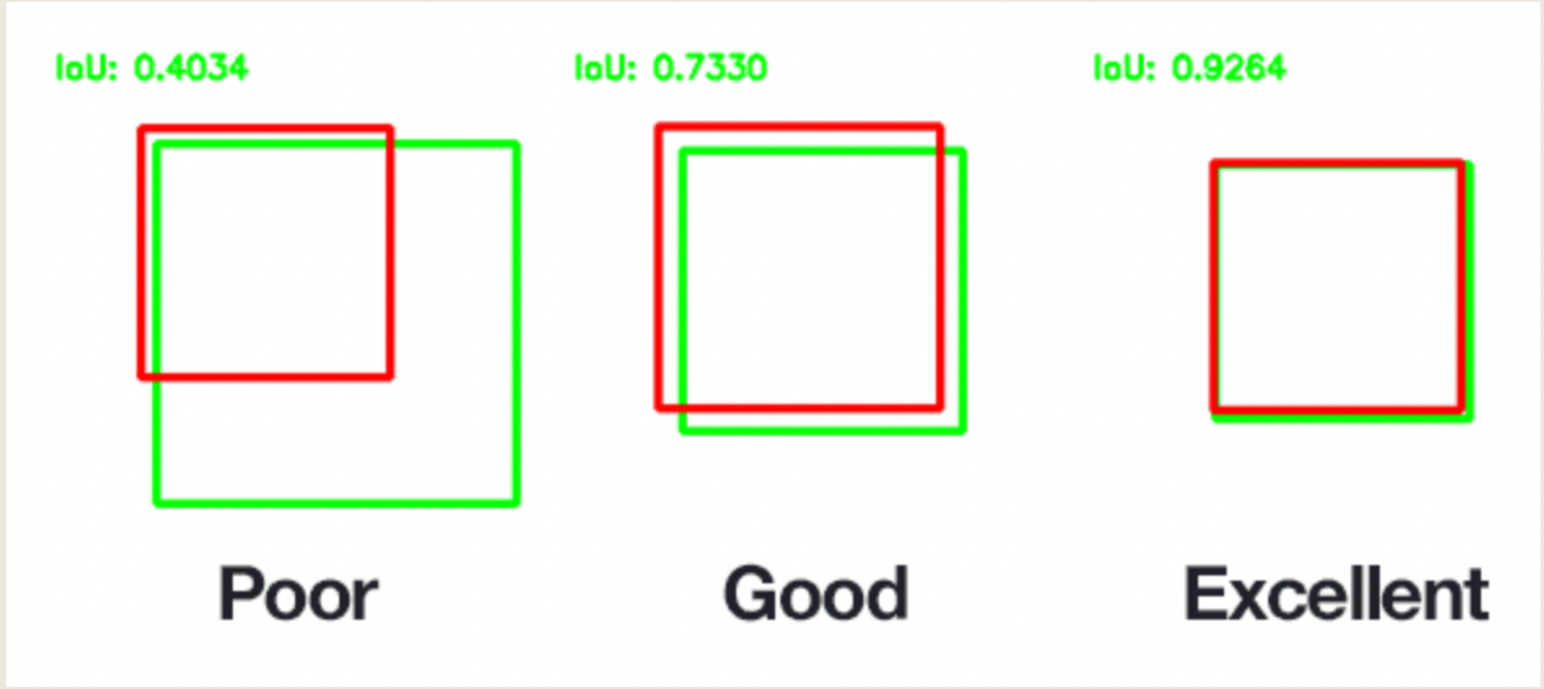
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 62, 62, 64)        1792

 activation_3 (Activation)   (None, 62, 62, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 29, 29, 64)        36928

 activation_4 (Activation)   (None, 29, 29, 64)        0

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 12544)             0

 dense_2 (Dense)             (None, 64)                802880

 dense_3 (Dense)             (None, 1)                 65

 activation_5 (Activation)   (None, 1)                 0

=================================================================
Total params: 841,665
Trainable params: 841,665
Non-trainable params: 0
_________________________________________________________________
```

R-CNN

- Building a Box
- Identifying the Bounds
- Used to define the location of the target object
- Trining Model on Bounds
- IoU

1 Better

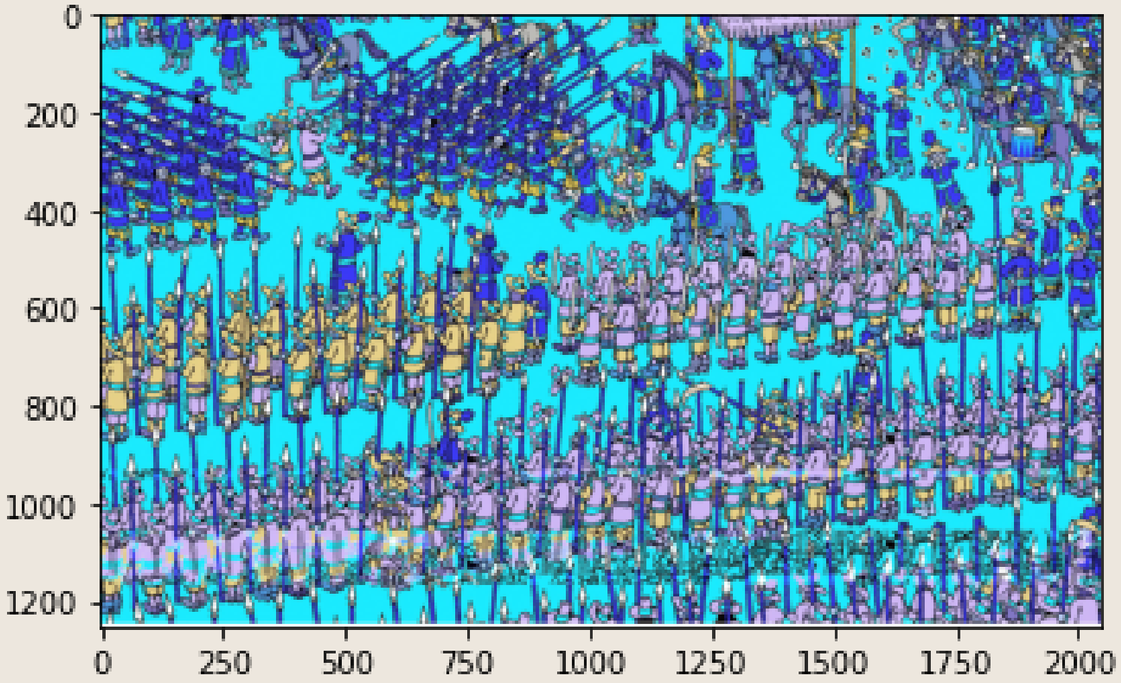
0 Worse



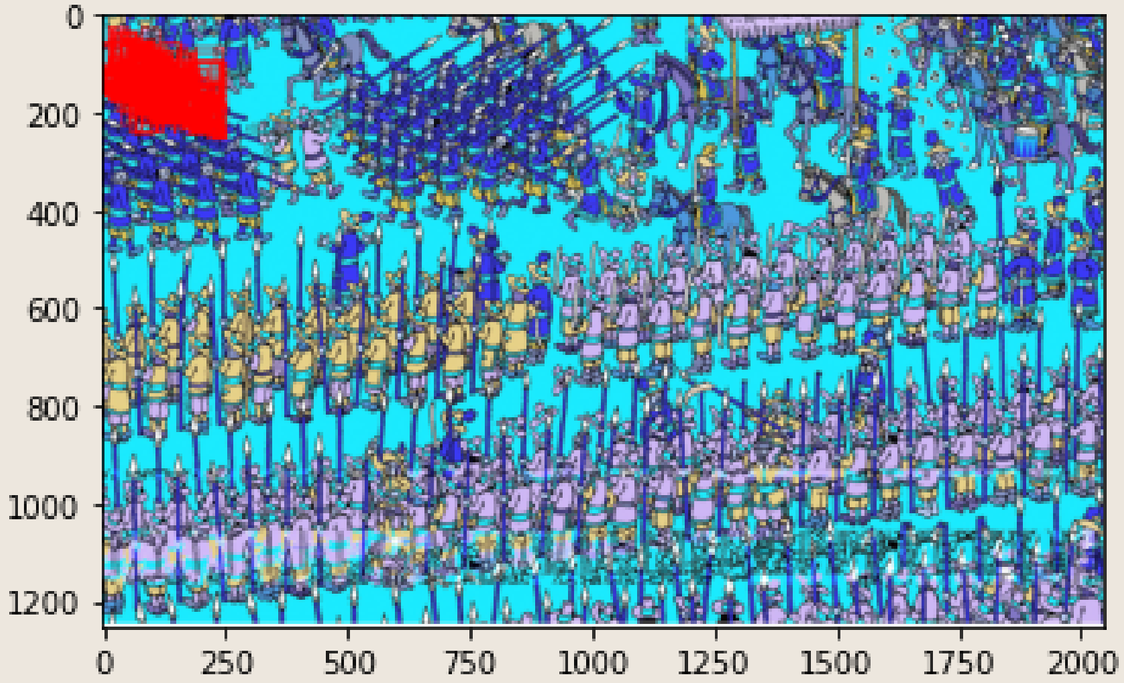
Labelled Image

```
for e,i in enumerate(os.listdir(original_img)):
    if e < 10:
        filename = i.split(".")[0]+".jpg"
        Index = 19
        filename2 = "waldo_"+str(Index)+".png.csv"
        img = cv2.imread(os.path.join(original_img,filename))
        #print(img)
        df = pd.read_csv(os.path.join(annot,filename2))
        plt.imshow(img)
        for row in df.iterrows():
            x1 = int(''.join(filter(str.isdigit, row[1][0].split(" ")[0])))
            y1 = int(''.join(filter(str.isdigit, row[1][0].split(" ")[1])))
            x2 = int(''.join(filter(str.isdigit, row[1][0].split(" ")[2])))
            y2 = int(''.join(filter(str.isdigit, row[1][2].split(" ")[1])))
            cv2.rectangle(img,(x1,y1),(x2,y2),(255,0,0), 2)
        plt.figure()
        plt.imshow(img)
        break
```

Original Image



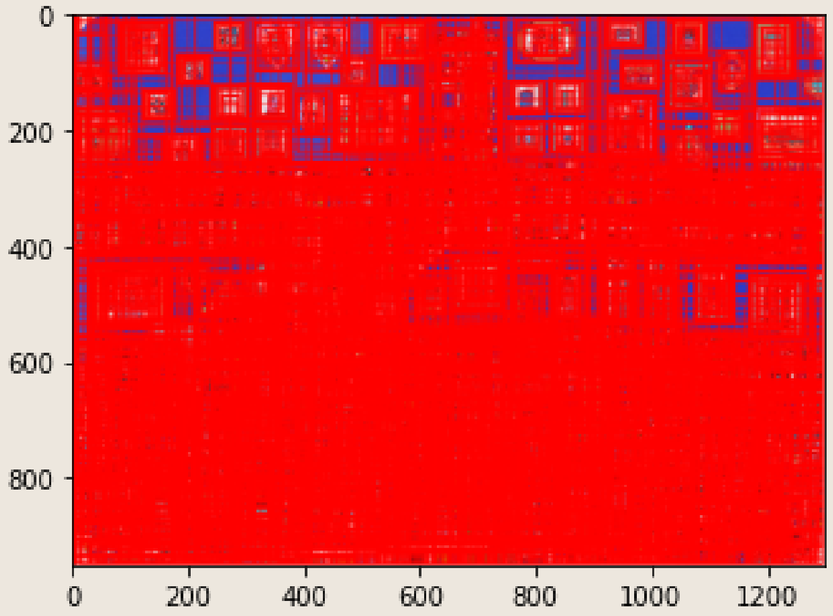
Annotated (not correct)



Selective Search

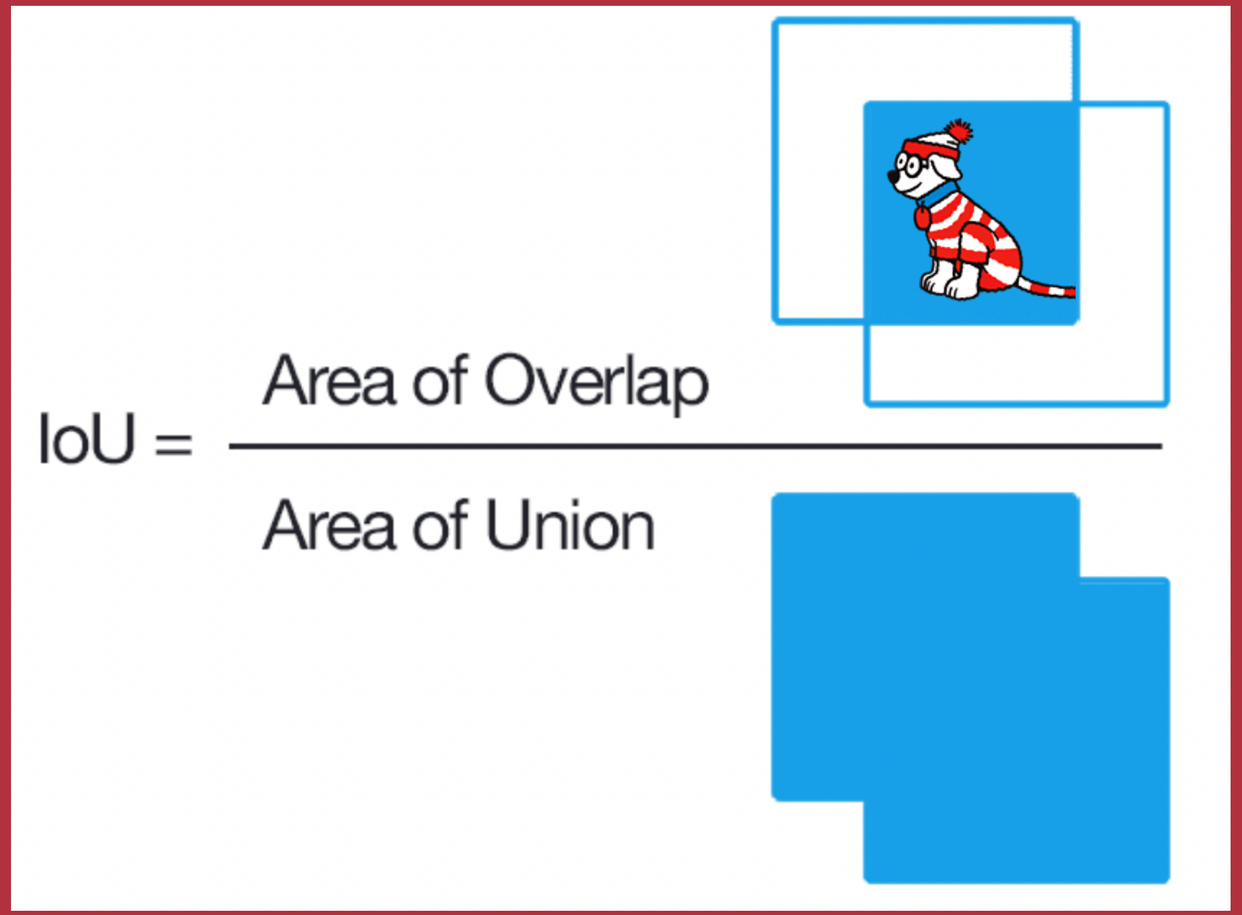
- Gets region proposals from selective search algorithm
- One single image
- Image on what the algorithm is doing

```
im = cv2.imread(os.path.join(original_img,'9.jpg'))
#cv2.imread(os.path.join(annot,"9.jpg"))
ss.setBaseImage(im)
ss.switchToSelectiveSearchFast()
rects = ss.process()
imOut = im.copy()
for i, rect in (enumerate(rects)):
    x, y, w, h = rect
    cv2.rectangle(imOut, (x, y), (x+w, y+h), (255, 0, 0), 1, cv2.LINE_AA)
    print(imOut)
plt.imshow(imOut)
```



Intersection Over Union

- Predicted bounding boxes from model.
- Boxes IoU greater than 0.7 (original paper it's 0.5) are considered positive example.
- Boxes with relative low IoU 0.3 negative examples.



Conclusions

Classifying Waldo

- Our model can classify if there is or isn't Waldo even with unbalanced data
- Accuracy: ~80 %
- Results are inconsistent

Finding Waldo

- Annotation was a success.
- Since our index was out of bounds IoU could not identify the bounding boxes.
- Did not find Waldo.

Thank you!