Kevin Mader

Lung Opacity Overview

last run 6 days ago · IPython Notebook HTML · 8,871 views
using data from RSNA Pneumonia Detection Challenge · Public

Notebook

Overview

The notebook aims to get a better feeling for the data and more importantly the distributions of values. We take the labels and combine them with the detailed class info and try and determine what the biggest challenges of the prediction might be.

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pydicom
import pandas as pd
from glob import glob
import os
from matplotlib.patches import Rectangle
det_class_path = '../input/stage_2_detailed_class_info.csv'
bbox_path = '../input/stage_2_train_labels.csv'
dicom_dir = '../input/stage_2_train_images/'
```

Detailed Class Info

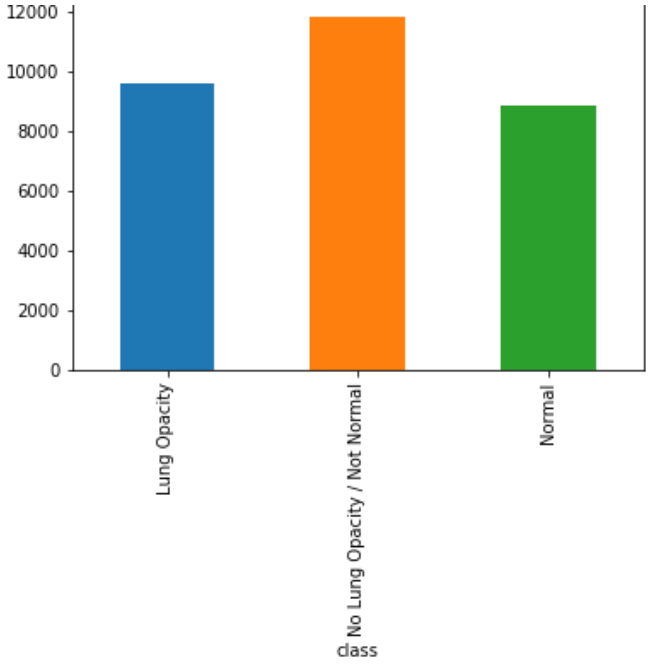
Here we show the image-level labels for the scans. The most interesting group here is the `No Lung Opacity / Not Normal` since they are cases that look like opacity but are not. So the first step might be to divide the test images into clear groups and then only perform the bounding box prediction on the suspicious images.

In [2]:
```
det_class_df = pd.read_csv(det_class_path)
print(det_class_df.shape[0], 'class infos loaded')
print(det_class_df['patientId'].value_counts().shape[0], 'patient case
s')
det_class_df.groupby('class').size().plot.bar()
det_class_df.sample(3)
```

```
30227 class infos loaded
26684 patient cases
```

Out[2]:

	patientId	class
11836	79ffd214-6464-406a-a78b-5aaa8cc1fe7e	Normal
13945	8b0cf414-03a7-499c-a080-cdc52003c84a	Lung Opacity
18257	acb0704d-dddd-4cf8-b678-ebe820443832	Normal



Load the Bounding Box Data

Here we show the bounding boxes

In [3]:
```
bbox_df = pd.read_csv(bbox_path)
print(bbox_df.shape[0], 'boxes loaded')
print(bbox_df['patientId'].value_counts().shape[0], 'patient cases')
bbox_df.sample(3)
```

```
30227 boxes loaded
26684 patient cases
```

Out[3]:

	patientId	x	y	width	height	Target
758	0a120828-703c-4273-9d62-317dc92c560e	NaN	NaN	NaN	NaN	0
29242	0f5d1591-5e8d-4328-9ba2-0d99deb8ad57	NaN	NaN	NaN	NaN	0
12206	7d0e3887-2263-4c54-8833-cb9886a62152	NaN	NaN	NaN	NaN	0

Combine Boxes and Labels

Here we bring the labels and the boxes together and now we can focus on how the boxes look on the images

In [4]:
```
# we first try a join and see that it doesn't work (we end up with too
```

```
 many boxes)
comb_bbox_df = pd.merge(bbox_df, det_class_df, how='inner', on='patien
tId')
print(comb_bbox_df.shape[0], 'combined cases')
```

```
37629 combined cases
```

Concatenate

We have to concatenate the two datasets and then we get class and target information on each region

In [5]:

```
comb_bbox_df = pd.concat([bbox_df,
                          det_class_df.drop('patientId',1)], 1)
print(comb_bbox_df.shape[0], 'combined cases')
comb_bbox_df.sample(3)
```

```
30227 combined cases
```

Out[5]:

	patientId	x	y	width	height	Target	class
10365	6e9ba999-9a90-4d68-bbd6-1eb63b71782c	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
25798	e6a3dc3b-b72d-4615-ae59-70626761b15d	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
1766	1889dd53-607f-4897-9be6-b2a5c078d6b3	105.0	486.0	295.0	339.0	1	Lung Opacity

Distribution of Boxes and Labels

The values below show the number of boxes and the patients that have that number.

In [6]:

```
box_df = comb_bbox_df.groupby('patientId').\
    size().\
    reset_index(name='boxes')
comb_box_df = pd.merge(comb_bbox_df, box_df, on='patientId')
box_df.\
    groupby('boxes').\
    size().\
    reset_index(name='patients')
```

Out[6]:

	boxes	patients
0	1	23286

1	2	3266
2	3	119
3	4	13

How are class and target related?

I assume that all the `Target=1` values fall in the `Lung Opacity` class, but it doesn't hurt to check.

In [7]:
```
comb_bbox_df.groupby(['class', 'Target']).size().reset_index(name='Patient Count')
```

Out[7]:

	class	Target	Patient Count
0	Lung Opacity	1	9555
1	No Lung Opacity / Not Normal	0	11821
2	Normal	0	8851

Images

Now that we have the boxes and labels loaded we can examine a few images.

In [8]:
```
image_df = pd.DataFrame({'path': glob(os.path.join(dicom_dir, '*.dcm'))})
image_df['patientId'] = image_df['path'].map(lambda x: os.path.splitext(os.path.basename(x))[0])
print(image_df.shape[0], 'images found')
img_pat_ids = set(image_df['patientId'].values.tolist())
box_pat_ids = set(comb_box_df['patientId'].values.tolist())
# check to make sure there is no funny business
assert img_pat_ids.union(box_pat_ids)==img_pat_ids, "Patient IDs should be the same"
```

```
26684 images found
```

In [9]:
```
image_bbox_df = pd.merge(comb_box_df,
                         image_df,
                         on='patientId',
                        how='left').sort_values('patientId')
print(image_bbox_df.shape[0], 'image bounding boxes')
```

```
image_bbox_df.head(5)
```

```
30227 image bounding boxes
```

Out[9]:

	patientId	x	y	width	height	Target	class	boxes	path
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	1	../input/stage_ 14fd-4e...
28989	000924cf-0f8d-42bd-9158-1af53881a557	NaN	NaN	NaN	NaN	0	Normal	1	../input/stage_ 0f8d-42...
28990	000db696-cf54-4385-b10b-6b16fbb3f985	316.0	318.0	170.0	478.0	1	Lung Opacity	2	../input/stage_ cf54-43...
28991	000db696-cf54-4385-b10b-6b16fbb3f985	660.0	375.0	146.0	402.0	1	Lung Opacity	2	../input/stage_ cf54-43...
28992	000fe35a-2649-43d4-b027-e67796d412e0	570.0	282.0	269.0	409.0	1	Lung Opacity	2	../input/stage_ 2649-43...

Enrich the image fields

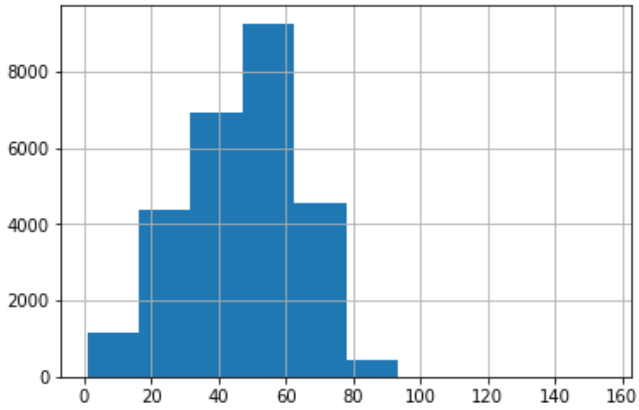
We have quite a bit of additional data in the DICOM header we can easily extract to help learn more about the patient like their age, view position and gender which can make the model much more precise

In [10]:

```
DCM_TAG_LIST = ['PatientAge', 'BodyPartExamined', 'ViewPosition', 'Pat
ientSex']
def get_tags(in_path):
    c_dicom = pydicom.read_file(in_path, stop_before_pixels=False)
    tag_dict = {c_tag: getattr(c_dicom, c_tag, '')
         for c_tag in DCM_TAG_LIST}
    tag_dict['path'] = in_path
    return pd.Series(tag_dict)
image_meta_df = image_df.apply(lambda x: get_tags(x['path']), 1)
# show the summary
image_meta_df['PatientAge'] = image_meta_df['PatientAge'].map(int)
image_meta_df['PatientAge'].hist()
image_meta_df.drop('path',1).describe(exclude=np.number)
```

Out[10]:

	BodyPartExamined	ViewPosition	PatientSex
count	26684	26684	26684
unique	1	2	2
top	CHEST	PA	M
freq	26684	14511	15166



In [11]:
```
image_full_df = pd.merge(image_df,
                         image_meta_df,
                         on='path')
```

Create Sample Data Set

We create a sample dataset covering different cases, and number of boxes

In [12]:
```
sample_df = image_bbox_df.\
    groupby(['Target','class', 'boxes']).\
    apply(lambda x: x[x['patientId']==x.sample(1)['patientId'].values[
0]]).\
    reset_index(drop=True)
sample_df
```

Out[12]:

	patientId	x	y	width	height	Target	class	boxes	path
0	700c0112-177c-4a8d-aee4-7e5a307659fd	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	1	../input/stage_2_tra 177c-4a...
1	c601b6c1-5fee-4980-8dd9-69c4fa054aeb	NaN	NaN	NaN	NaN	0	Normal	1	../input/stage_2_tra 5fee-49...
2	b93bc9fc-3863-4ac7-a097-c8977df98d4b	153.0	270.0	217.0	149.0	1	Lung Opacity	1	../input/stage_2_tra 3863-4a...
3	363a77dc-eb92-45ec-9872-aaf32a4bc920	614.0	418.0	173.0	301.0	1	Lung Opacity	2	../input/stage_2_tra eb92-45...
4	363a77dc-eb92-45ec-9872-aaf32a4bc920	219.0	272.0	261.0	309.0	1	Lung Opacity	2	../input/stage_2_tra eb92-45...

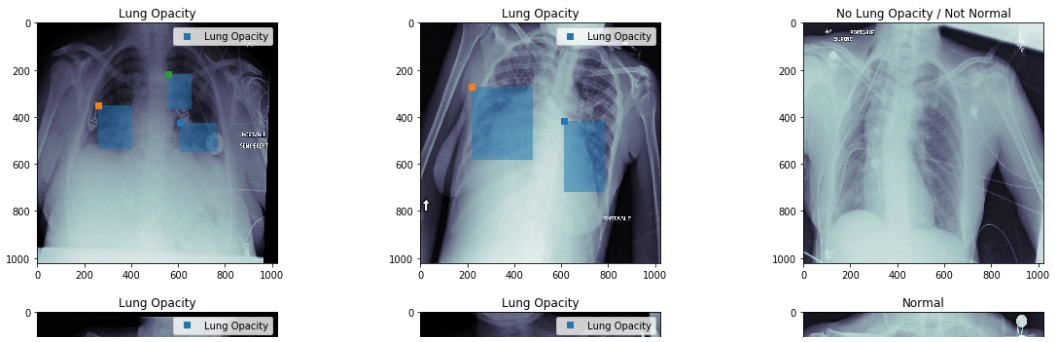
5	345e6018-89b7-41d0-bd33-2350c1c60311	607.0	427.0	157.0	121.0	1	Lung Opacity	3	../input/stage_2_tra 89b7-41...
6	345e6018-89b7-41d0-bd33-2350c1c60311	261.0	349.0	144.0	190.0	1	Lung Opacity	3	../input/stage_2_tra 89b7-41...
7	345e6018-89b7-41d0-bd33-2350c1c60311	557.0	217.0	99.0	150.0	1	Lung Opacity	3	../input/stage_2_tra 89b7-41...
8	7d674c82-5501-4730-92c5-d241fd6911e7	358.0	320.0	159.0	186.0	1	Lung Opacity	4	../input/stage_2_tra 5501-47...
9	7d674c82-5501-4730-92c5-d241fd6911e7	283.0	528.0	197.0	314.0	1	Lung Opacity	4	../input/stage_2_tra 5501-47...
10	7d674c82-5501-4730-92c5-d241fd6911e7	718.0	573.0	133.0	227.0	1	Lung Opacity	4	../input/stage_2_tra 5501-47...
11	7d674c82-5501-4730-92c5-d241fd6911e7	658.0	315.0	164.0	203.0	1	Lung Opacity	4	../input/stage_2_tra 5501-47...

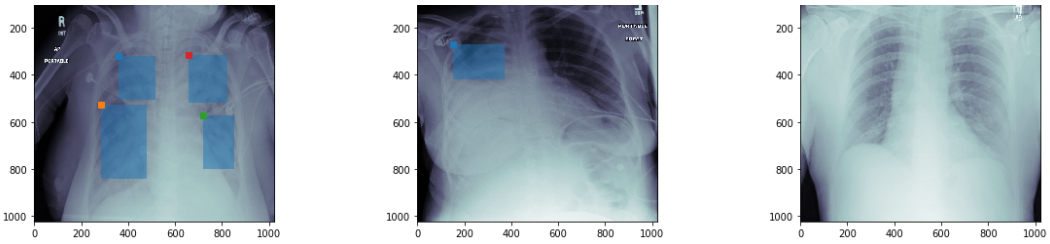
Show the position and bounding box

Here we can see the position (point) and the bounding box for each of the different image types

In [13]:

```
fig, m_axs = plt.subplots(2, 3, figsize = (20, 10))
for c_ax, (c_path, c_rows) in zip(m_axs.flatten(),
                    sample_df.groupby(['path'])):
    c_dicom = pydicom.read_file(c_path)
    c_ax.imshow(c_dicom.pixel_array, cmap='bone')
    c_ax.set_title('{class}'.format(**c_rows.iloc[0,:]))
    for i, (_, c_row) in enumerate(c_rows.dropna().iterrows()):
        c_ax.plot(c_row['x'], c_row['y'], 's', label='{class}'.format(
**c_row))
        c_ax.add_patch(Rectangle(xy=(c_row['x'], c_row['y']),
                                 width=c_row['width'],
                                 height=c_row['height'],
                                  alpha = 0.5))
        if i==0: c_ax.legend()
```



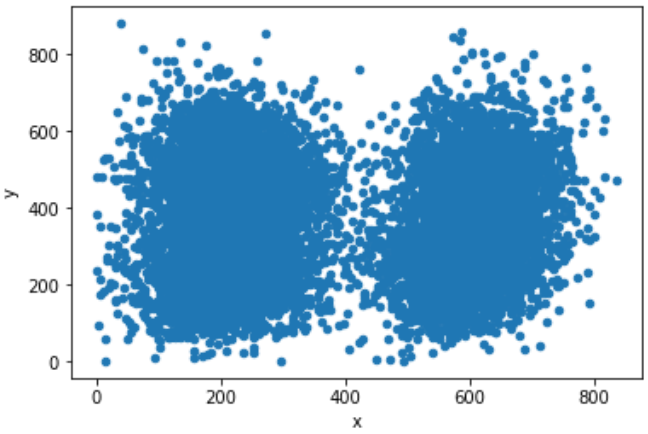


Bounding Box Distribution

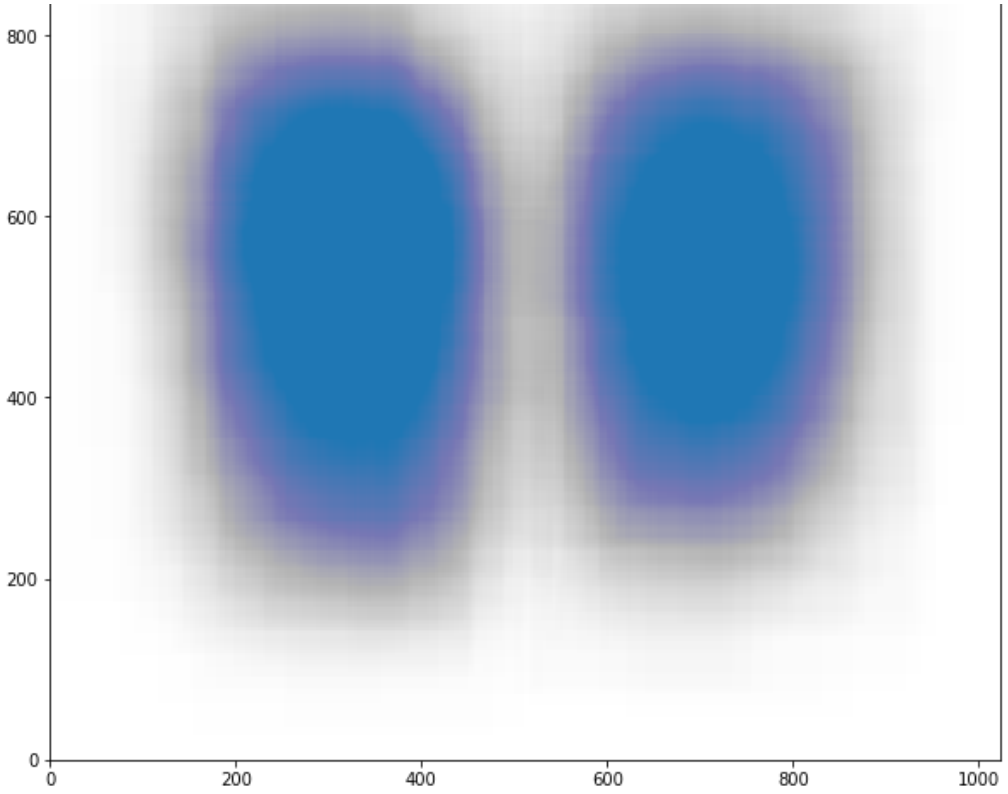
Here we just look at the bounding box distribution to get a better idea how this looks over the whole dataset

In [14]:
```
pos_bbox = image_bbox_df.query('Target==1')
pos_bbox.plot.scatter(x='x', y='y')
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9229df32b0>
```



In [15]:
```
fig, ax1 = plt.subplots(1, 1, figsize = (10, 10))
ax1.set_xlim(0, 1024)
ax1.set_ylim(0, 1024)
for _, c_row in pos_bbox.sample(1000).iterrows():
    ax1.add_patch(Rectangle(xy=(c_row['x'], c_row['y']),
                  width=c_row['width'],
                  height=c_row['height'],
                            alpha=5e-3))
```



Show the boxes as segmentation

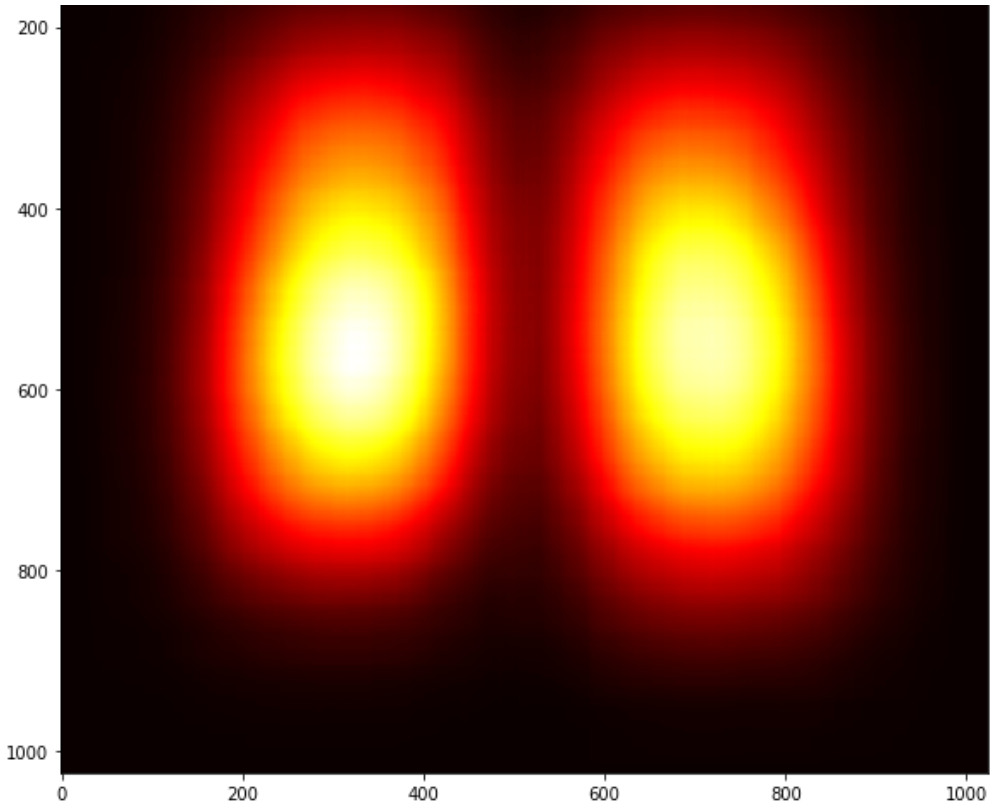
By showing them as segmentations we can get a better probability map for where the opacity regions are most likely to occur

In [16]:

```
# Show the boxes themselves
X_STEPS, Y_STEPS = 1024, 1024
xx, yy = np.meshgrid(np.linspace(0, 1024, X_STEPS),
           np.linspace(0, 1024, Y_STEPS),
           indexing='xy')
prob_image = np.zeros_like(xx)
for _, c_row in pos_bbox.sample(5000).iterrows():
    c_mask = (xx>=c_row['x']) & (xx<=(c_row['x']+c_row['width']))
    c_mask &= (yy>=c_row['y']) & (yy<=c_row['y']+c_row['height'])
    prob_image += c_mask
fig, ax1 = plt.subplots(1, 1, figsize = (10, 10))
ax1.imshow(prob_image, cmap='hot')
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x7f922a4447b8>
```



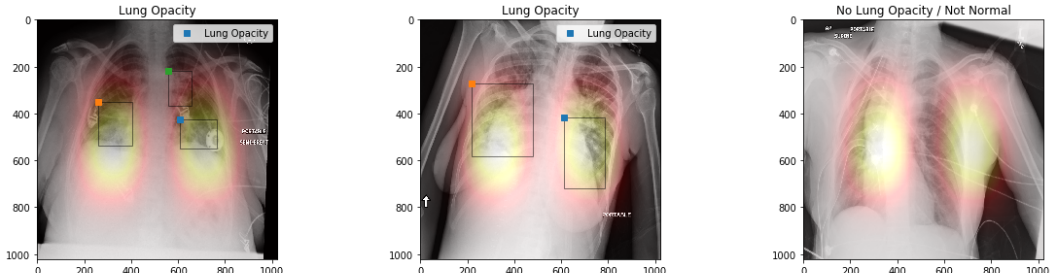
Overlay the Probability on a few images

Does the probability we calculate seem to make sense? or have we flipped something somewhere?

In [17]:

```
fig, m_axs = plt.subplots(2, 3, figsize = (20, 10))
for c_ax, (c_path, c_rows) in zip(m_axs.flatten(),
                    sample_df.groupby(['path'])):
    c_img_arr = pydicom.read_file(c_path).pixel_array
    # overlay
    c_img = plt.cm.gray(c_img_arr)
    c_img += 0.25*plt.cm.hot(prob_image/prob_image.max())
    c_img = np.clip(c_img, 0, 1)
    c_ax.imshow(c_img)

    c_ax.set_title('{class}'.format(**c_rows.iloc[0,:]))
    for i, (_, c_row) in enumerate(c_rows.dropna().iterrows()):
        c_ax.plot(c_row['x'], c_row['y'], 's', label='{class}'.format(
**c_row))
        c_ax.add_patch(Rectangle(xy=(c_row['x'], c_row['y']),
                                 width=c_row['width'],
                                 height=c_row['height'],
                                  alpha = 0.5,
                                 fill=False))
        if i==0: c_ax.legend()
fig.savefig('overview.png', figdpi = 600)
```



Did you find this Kernel useful?
Show your appreciation with an upvote

206

Comments (32)

All Comments | Sort by Hotness

Click here to enter a comment...

Xavier • Posted on Version 3 • 2 months ago • Options • Reply 0

Great work and visualizations, thanks for sharing Kevin!!

CHANS • Posted on Version 3 • 2 months ago • Options • Reply 0

Great help for insight , thx

SterlingRamroach • Posted on Version 3 • 2 months ago • Options • Reply 0

What an amazing tutorial!

Tian Xia • Posted on Version 3 • 2 months ago • Options • Reply 0

This is a terrific kernel!

EvaTC • Posted on Version 3 • 2 months ago • Options • Reply 0

Really good overview, thanks. I'm new to this site and this was incredibly helpful.

jongyun jung • Posted on Version 3 • 2 months ago • Options • Reply 0

Great visualization!

GSD • Posted on Version 3 • 2 months ago • Options • Reply

0

Excellent start Kevin ..Great work..

Faizunnabi • Posted on Version 3 • 2 months ago • Options • Reply

0

Fantastic visuals Kevin...grt effort !

JuanOI • Posted on Version 3 • 2 months ago • Options • Reply

0

Really nice job!

Vishy • Posted on Version 3 • 2 months ago • Options • Reply

0

Nice one, please checkout a list of useful references here and suggest

TSHLNIHAO • Posted on Version 3 • 2 months ago • Options • Reply

0

good ! especially the start point in the picture

Boadi Samson • Posted on Version 3 • 2 months ago • Options • Reply

0

Great kenel. Helpful

Morgan • Posted on Version 3 • 2 months ago • Options • Reply

0

Great work !

ChungyehWang • Posted on Version 3 • 2 months ago • Options • Reply

0

Thanks for sharing Kevin!!

Hamza Abdullah • Posted on Version 3 • 2 months ago • Options • Reply

0

Great approach!

Zeeshan Hayat • Posted on Version 3 • 2 months ago • Options • Reply

0

Great work

boosting_75 • Posted on Version 3 • 2 months ago • Options • Reply

0

Clear and helping tutorial:-)

Harm Buisman • Posted on Version 3 • 2 months ago • Options • Reply

0

Very nice notebook, thanks for sharing!

For the Distribution of Boxes and Labels you may want to split between the box 0 and 1 counts, they are now combined

Samet Maraşlı • Posted on Version 3 • 2 months ago • Options • Reply

0

I guess there is an error at Distribution of Boxes and Labels table. This would correct the mistake:

```
box_df = comb_bbox_df.groupby('patientId')['x'].count().reset_index(name='boxes')
```

Gabriel Preda • Posted on Version 3 • 2 months ago • Options • Reply

0

Excellent visualizations and great tutorial. I would like to reuse the method to show overlapping windows over the radiology images.

Tianyu Lan • Posted on Version 3 • 2 months ago • Options • Reply

0

great kernal, thanks for sharing!

sunnysingh97g • Posted on Version 3 • 2 months ago • Options • Reply

0

Great Visualization

Brian Smith • Posted on Version 3 • 2 months ago • Options • Reply

0

Thanks for sharing Kevin - excellent kernel!

Zhang Likang • Posted on Version 3 • 2 months ago • Options • Reply

0

Awesome

Alpamys Tolegen • Posted on Version 3 • 2 months ago • Options • Reply

0

Great work! But I cannot get why so many people have 1 bounding box cuz I am getting different values?

kartik • Posted on Version 3 • a month ago • Options • Reply

0

Awesome Kernel

Prashanth Kurum... • Posted on Latest Version • 5 days ago • Options • Reply

0

Great kernel, thank you for sharing!

2 months ago

This Comment was deleted.

2 months ago

This Comment was deleted.

2 months ago

This Comment was deleted.

2 months ago

This Comment was deleted.

2 months ago

This Comment was deleted.